

# **Software Documentation Report**

Training and Development  
Reporting System

Marya Belanger

For the Royal Commission Medical Center

24/1/2016 to 5/5/2016

## **SAMPLE NOTE**

I have cut down this report from its original version to highlight chapters relevant to software documentation technical writing. I also modified the original cover page used for publishing for this portfolio. I created this report to document a software system I designed and implemented during an internship.

# TABLE OF CONTENTS

[Chapters 1, 2, 4, 5 and appendices B, C redacted]

CHAPTER 3 .....	1
3.2 Training and Development Reporting System .....	1
3.2.1 Project Timeline .....	1
3.2.2 System Analysis .....	2
3.2.3 Requirements .....	4
3.2.4 Design .....	8
3.2.5 User Interface .....	19
3.2.6 Implementation .....	51
3.2.7 Testing .....	61
3.2.8 Documentation .....	73
GLOSSARY .....	VI
APPENDIX A .....	VII

## TABLE OF FIGURES

Figure 3.2: Project Timeline .....	2
Figure 3.3: Start Page.....	20
Figure 3.4: Secretary View – Home Tab .....	20
Figure 3.5: New User .....	21
Figure 3.6: Change Password .....	22
Figure 3.7: New Year.....	22
Figure 3.8: New Year Confirmation Dialog .....	23
Figure 3.9: Adding COOP Trainee .....	24
Figure 3.10: Dropdown Lists .....	25
Figure 3.11: New University Dialog.....	25
Figure 3.12: More Dropdown Lists .....	26
Figure 3.13: COOP View.....	27
Figure 3.14: Remaining Columns in COOP View.....	27
Figure 3.15: Update Dialog.....	28
Figure 3.16: Training Activities Tab .....	29
Figure 3.17: Training Activities Dropdown List .....	29
Figure 3.18: Training Activities View .....	30
Figure 3.19: Remaining Columns in Training Activities View .....	30
Figure 3.20: Update Dialog for Training Activities.....	30
Figure 3.21: Symposium/Conference Tab .....	31
Figure 3.22: Symposium/Conference View.....	31
Figure 3.23: Remaining Columns in Symposium/Conference View.....	32
Figure 3.24: CME Tab .....	32
Figure 3.25: CME View.....	33
Figure 3.26: Remaining Columns in CME View.....	33
Figure 3.27: Medical Course Tab .....	34
Figure 3.28: Dropdown List for CPR Courses.....	34
Figure 3.29: Medical Course View .....	35
Figure 3.30: Remaining Columns in Medical Course View .....	35
Figure 3.31: Non Medical Course Tab .....	36
Figure 3.32: Activity Dropdown List for Non Medical Course.....	36
Figure 3.33: Non Medical Course View .....	37
Figure 3.34: Remaining Columns of Non Medical Course View .....	37
Figure 3.35: Reports Tab .....	38
Figure 3.36: Saving the Annual Report .....	38
Figure 3.37: Report Success .....	39
Figure 3.38: Sample of the Annual Report .....	39
Figure 3.39: Report Chart Success.....	40

Figure 3.40: Sample of the Report Charts .....	40
Figure 3.41: CPR Activities Tab.....	41
Figure 3.42: CPR Activities Dropdown List.....	41
Figure 3.43: Adding New CPR Course .....	42
Figure 3.44: CPR Activities View .....	43
Figure 3.45: Updated Attendees Amount in Medical Course View .....	43
Figure 3.46: CPR Activities Update .....	44
Figure 3.47: Certified Staff Tab.....	44
Figure 3.48: ID and Course Dropdown Lists.....	45
Figure 3.49: New Certified Staff .....	45
Figure 3.50: Certified Staff View .....	46
Figure 3.51: Remaining Columns of Certified Staff View .....	46
Figure 3.52: Update Staff.....	47
Figure 3.53: Update Certification Row.....	47
Figure 3.54: CPR Reports Tab.....	48
Figure 3.55: CPR Activities Report Sample .....	48
Figure 3.56: Trainees Tab .....	49
Figure 3.57: Trainees View.....	49
Figure 3.58: Remaining Columns in Trainees View .....	50
Figure 3.59: Training Reports.....	50
Figure 3.60: Trainee Report Sample.....	51
Figure 3.61: Test Case 1 .....	61
Figure 3.62: Test Case 2 .....	62
Figure 3.63: Test Case 3 .....	62
Figure 3.64: Test Case 4 .....	63
Figure 3.65: Test Case 5 .....	64
Figure 3.66: Test Case 6 .....	64
Figure 3.67: Test Case 7 .....	65
Figure 3.68: Test Case 8 .....	65
Figure 3.69: Test Case 9 .....	66
Figure 3.70: Test Case 10 .....	67
Figure 3.71: Test Case 11 .....	67
Figure 3.72: Test Case 12 .....	68
Figure 3.73: Test Case 13 .....	69
Figure 3.74: Test Case 14 .....	70
Figure 3.75: Test Case 15 - 1 .....	71
Figure 3.76: Test Case 15 - 2.....	71
Figure 3.77: Test Case 16 .....	72
Figure 3.78: Test Case 17 .....	72

## TABLE OF DIAGRAMS

Diagram 3.1: Database Schema Diagram .....	15
Diagram 3.2: Entity Relationship Diagram .....	16
Diagram 3.3: Use Case Diagram .....	17
Diagram 3.4: Navigational Diagram.....	18
Diagram 3.5: Activity Diagram .....	19

## **CHAPTER 3**

### **3.2 Training and Development Reporting System**







The TDS is a desktop application for users in various sections of the Training and Development department to consistently share data logs while maintaining their integrity, and produce reports of that data automatically. This project was chosen because of its clearly defined goal (to replace a manual workflow) and its obvious potential benefit to the hospital (improving efficiency and data integrity).

The TDS was developed in the .NET environment in Microsoft Visual Studio Professional 2015, and Microsoft Blend 2015 IDEs. It is implemented using the C# programming language, the XAML mark-up language, and the SQL data language. It is a Windows Presentation Foundation application (WPF). It contains 16 classes of functionality code (C#), 9 classes of design code (XAML), and 9 classes of “behind” or interaction-logic code (XAML.CS).

#### **3.2.1 Project Timeline**

- Identifying an issue within a RCMC department with realistic potential to be mediated by a software project – 2 weeks.
- Elicited requirements from Training Department users by interview and questionnaire methods, and analyzed system through system modeling with use case, activity, and entity relationship diagrams – 2 weeks.
- Designed database, populated with mock-data and tested constraints and relationships with various SQL queries – 1 week
- Programmed application, beginning with design of the system (2 weeks), and followed by functionality – 7 weeks.
- Tested and released system to users, presented to hospital departments for approval of release, and creation of user manual – 2 weeks.

The following is the timeline as it was diagrammed using Microsoft Project 2007:

		Task Name	Duration	Start	Finish
1		Identify Problem	10 days?	Sun 1/24/16	Thu 2/4/16
2		Elicit Requirements/System Analysis	10 days?	Sun 2/7/16	Thu 2/18/16
3		Design Database	5 days?	Sun 2/21/16	Thu 2/25/16
4		Programming	41 days?	Sun 2/28/16	Thu 4/21/16
5		Testing and Release	10 days?	Sun 4/24/16	Thu 5/5/16

**Figure 1.2: Project Timeline**

### **3.2.2 System Analysis**

The system analysis is carried out at the onset of the project lifecycle. After interviewing the potential users (see Appendix A), it was necessary to create a clear problem statement as a base to design a proposed solution. With those in mind, next came listing out solid objectives for the system to accomplish. Lastly, the scope of the system was defined. The importance of these stages is to remove any ambiguity from the future design and development stages.

#### **3.2.2.1 Problem**

The Training and Development department at the RCMC is facing some difficulty culminating data throughout the year. The secretary of the department is responsible for maintaining records of all the trainees, conferences, classes, and continuous education going on at the hospital. At the end of each year, the department head requires an annual report, with the data from all these records summarized. Currently, the secretary manually records this information in Word or Excel files as she receives it, and produces the summary a significant amount of time after the year ends, because of the large amount of data involved. It is not appropriate for this type of work flow to be maintained in such a way.

Of the previously mentioned topics, the trainee records and medical class records are also the responsibility of the Training department, just that of different sectors. The Training unit maintains data on each individual trainee, and the CPR unit maintains data on medical classes. Despite being in the same department, there is no efficient practice in place for these units to provide updates of their records to the secretary. Both units also maintain their records by manually storing them in Word or Excel files. This creates issues of inconsistency between the units and the secretary. For example, if the secretary



needs to search the Training unit's data for trainees of a certain training type, but the Training unit has misspelled the type, or if there is some miscommunication of which trainees fall under what types, the consistency of the shared data is lost.

Additionally, the Training unit and CPR unit have their own issues with maintaining data. The Training unit handles records of hundreds of trainees coming and going through the hospital every year. When entering these records, they will often have to repeat the same fields for many different trainees, such as university names, statuses of their training, and department names. However, doing so using Excel causes the issue that if a field is entered with even a slightly different spelling, a trailing whitespace, or difference in capitalization, Excel will not filter through the data as expected. Considering the high volume of records entered, this proves problematic.

The CPR unit conducts different medical training classes for staff in the hospital several times a week. As expected, each class will have a different number of attendees each time. This poses a problem, as the CPR unit must submit the total annual attendees for each individual course type to the secretary at the end of the year for the annual report. Currently, this is carried out manually, which results in errors and loss of data. The CPR also keeps records of which staff are certified in each of the courses offered. Considering the large amount of medical personnel in the hospital, managing this data manually is cause for many issues.

#### **3.2.2.2 Proposed Solution**

As a solution, the Training & Development Reporting System (TDS) was suggested. The TDS is a means to organize and automate the entering, sharing and maintenance of data, as well as automatic report creations customized to each user of the system. It intends to eliminate the need for excessive communication between users, and to eliminate inconsistencies that occur in frequently used data. In most cases, manually entering a value will only have to occur once, and thereafter it will be saved in the system to be used again. Additionally, the users will never have to manage an overwhelming Word or Excel file again.

The secretary will use the TDS for storing all of the data necessary for the report in one secure place. She will be able to receive the data required from other sections of the department automatically as it is updates. The CPR unit can easily send class attendee amounts to the secretary without ever having to leave his office, as well as maintain the data for all the certified staff members in one place. The Training unit will use the TDS to organize and sort through the multitudes of trainee records at the click of a button, while simultaneously being able to update the secretary to the number of trainees in each type.

### **3.2.2.3 Objectives**

The objectives of the Training and Development Reporting System are as follows:

- Store data important to each users' workflow
- Communicate data between users when appropriate
- Create reports automatically for each user based on their needs
- Eliminate manual, error-prone functionality of the department

### **3.2.2.4 Scope of System**

The TDS is meant to be deployed in the Training and Development department in the Royal Commission Medical Center. Its users should be the secretary of the department, the Training unit, the CPR unit, and any assistants or trainees working under them. The system should be easily be expanded to other users in the Training department who want to maintain and share their data with each other, but should not be expanded outside of the Training department, to prevent convolution of the data involved.

### **3.2.3 Requirements**

This section details the requirements for the system. The software requirements outline the software programs, development environments, operating system and database server needed to develop and run the application. The functional requirements explain the users and their needs for the functionality of the program. The non functional requirements are the non technical expectations for the application. The secretary provided a sample of previous reports on the topic of the system, which the TDS was largely based off of (see Appendix B).

### **3.2.3.1 Hardware Requirements**

The hardware requirements of the system are as follows:

- 64/86 bit operating system, x64/x86 based processor
- 4.00 GB RAM
- At least 30 MB available on Hard Disk for installation of system
- 1.70 GHz CPU, within reasonable range for optimal performance

### **3.2.3.1 Software Requirements**

The software requirements of the system are as follows:

- Windows OS between Windows 2007 and Windows 10
- Microsoft Word 2007 and later
- Microsoft Excel 2007 and later
- Oracle 10g server
- SQLPlus command line
- Microsoft Visual Studio Professional 2015
- Microsoft Blend for Visual Studio Professional 2015
- .NET 4.5 Package

### **3.2.3.2 Functional Requirements**

The functional requirements for each of the three users, Secretary, Training Unit, and CPR Unit, are as follows:

- Secretary
  - The system shall store yearly data on the following topics: Symposiums and Conferences, Training activities, Continuous Medical Education, Medical Courses, Non-Medical Courses, and COOP Trainees.
  - The system shall produce reports in the form of Microsoft Word tables on all the previously mentioned topics.
  - The system shall produce summaries in the form of Microsoft Word tables on all the previously mentioned topics.
  - The system shall produce graphs and charts in the form of Microsoft Excel charts on all the previously mentioned topics.

- The topic of “Training Activities” shall automatically be updated when either the Training unit or Secretary adds new trainees in the system.
- The topic of “Medical Courses” shall be automatically updated when the CPR unit adds new CPR activities in the system.
- Entering data into the system shall be maintained in a way where consistency and integrity are maintained in topics shared between users, like Medical Courses, Training Activities, hospital departments, university names, etc.
- Training Unit
  - The system shall store data on all trainees in the hospital.
  - The trainee data shall be searchable by type of training, paid or not paid, status, date started, and every other piece of data stored on trainees.
  - The system shall automatically send the count (amount) of trainees in each training group to the Secretary, each time the amount changes.
  - The system shall store yearly data on the topic of Non-Medical courses, shared with the Secretary
  - The system shall produce reports and summaries in the form of Microsoft Word tables on the topic of Non-Medical courses.
  - The system shall produce reports in the form of Microsoft Excel worksheets on the trainee data.
- CPR Unit
  - The system shall store CPR activities, the dates they occur, and the number of attendees at each course.
  - The system shall automatically send the sum of attendees for each CPR activity type to the secretary, each time the amount changes.
  - The system shall store staff member details, and their corresponding certifications to each of the various CPR activities.
  - The system shall produce reports on the CPR activities and staff certifications in the form of Microsoft Excel worksheets.

### **3.2.3.3 Non Functional Requirements**

The non-functional requirements of the system are as follows:

- Availability:
  - The application should be available during office hours, 7 AM until 5 PM weekdays. The availability of the system relies on the hospital's servers, which experience only one or two minutes of downtime every few months.
- Portability
  - The application should run on any CPU, 32, 64 or 86-bit systems. The application is centered on a Windows OS, and will not maintain its capabilities on other OSs.
- Performance
  - Response time is partly reliant on the hospital's servers. Response time of the application itself, due to efficiency of code written, should be only several seconds, and not exceeding 30 seconds, depending on the complexity of the action being attempted by the application (i.e., creating Word documents takes the application approximately 28 seconds).
- Reliability
  - The system will maintain its reliability in the form of its data's integrity. This is maintained by restricting users' ability to format data to an unnecessary extent, and using frequent validation checks both on server and end-user side.
- Maintainability
  - The application has been designed in such a way that maintenance can be easily achieved through simple reusable methods and clear division of user, data and connection logic.

#### **3.2.3.4 Data Requirements**

The data requirements of the application include the username, password and permission of the three original users. With that data, the original users can continue to add new users under their permission, but one of each user type must be in the system before deployment so the system can be accessed. Additionally, the system requires that all hospital departments are already listed by name in the system's data stores.

### **3.2.4 Design**

This section is dedicated to the design phase of the creation of the TDS. This phase involves designing the database for the system, and the system design itself, which is a collection of diagrams created to aid in the clarity and direction of the system.

The general description of the application is as follows: The secretary user will log in to the system and access the various topics under her jurisdiction, which includes: symposiums and conferences, training activities, COOP trainees, medical and non-medical courses, and continuous medical education. For these topics, she can add, delete, update and view their data. She can also create full reports in Word and Excel on these topics with the click of a button. The Training unit and CPR unit are provided the same capabilities (including report creation), but their topics are limited to: all hospital trainees and non-medical courses, and CPR activities and certified staff members, respectively.

#### **3.2.4.1 Database Design**

The tables in this database were created specifically for the use of the TDS. This section details each table and its description, the fields of each table and their type, and schema and entity relationship diagrams.

##### **3.2.4.1.1 Description of Database**

One of the main objectives for the TDS is to maintain up-to-date data sharing and integrity between users in the Training department. Because of this, the relationships between some tables and the constraints on them are important. There are eleven tables in the database. The following is a description of each relation:

**MASTER\_TABLE:** For maintaining important data to be safe from user tampering. Data that needs to maintain its consistency throughout the program (data that is shared between two or more users) is saved here. This includes university names, CPR training activities, hospital departments, and training activities. Each different type is a foreign key in another table. The fields are: NAME – the name visible to users in the program, (i.e. King Fahd University), and TYPE – what the name represents (i.e. Private-University)

**USERS:** For authentication of users and identification of which view they are allowed to access. The fields are USERNAME and PASSWORD, for signing in, and PERMISSION, which can be CPR, Training, or Secretary.

**SYMP\_CONF:** “Symposiums and Conferences”. This is one of the subjects the secretary is required to report on at the end of each year. The fields (the data the report requires) are SN, ACTIVITY – the title of the event (i.e. 4<sup>th</sup> Annual Infection Control Conference), DATE\_ – the date or period of time the event takes place (i.e. From June 24<sup>th</sup> to July 1<sup>st</sup> 2016) , VENUE – where the conference or symposium is held (i.e. Movenpick, Yanbu), SCFHS\_CREDIT\_HOURS – the credits that attendees will be rewarded for attending the event, part of a doctor-training system, NO\_OF\_ATTENDEES – how many people attended, and STAMP – a timestamp of when the data is actually inserted in the database.

**CME:** “Continuous Medical Education”. Another subject of the annual report, it covers basic information about the continuous classes doctors and nurses are required to take throughout the year. It has similar fields as SYMP\_CONF: SN, ACTIVITY, VENUE, SCFHS\_CREDIT\_HOURS and STAMP. It also has the fields TOTAL – how many classes were conducted over the whole year, and FREQUENCY – which day of the week the course occurs.

**NON\_MEDICAL\_COURSE:** Another subject for the annual report, this table keeps track of all non-medical related courses carried out at the RCMC. It contains the same previously mentioned fields, SN, ACTIVITY, DATE\_, VENUE, NO\_OF\_ATTENDEES and STAMP, as well as the additional field TARGET\_GROUP – which department or group in the hospital the course is meant for (i.e. Doctors and nurses, security personnel, all staff, etc.).

**MEDICAL\_COURSE:** Another subject for the annual report, this table keeps track of all medical related courses carried out at the RCMC, including CPR courses. It contains all the same fields as NON\_MEDICAL\_COURSE.

**TRAINING\_ACTIVITIES:** The records in this table also serve as a summary for the annual training department report. It has the same fields SN, ACTIVITY, DATE\_, and STAMP. It also has ENTITY – usually the university(s) or program that provides the

trainees for each training activity, TYPE – whether the training is specific to medical fields, administrative, etc., and NO\_OF\_TRAINEES – for each training activity, how many trainees were trained at the hospital this year.

**TRAINEE:** For the record of all trainees in the hospital. This includes (fields) a trainee's NAME, DEPARTMENT they're assigned to, UNIVERSITY, START\_DATE, DURATION – how long they'll be training for, TRAINING\_TYPE – from the TRAINING\_ACTIVITIES table, EMAIL, PHONE, STATUS – whether they're active, completed, new etc., NOTE and STAMP.

The next few tables are used specifically by the CPR unit of the Training Department.

**CPR:** This table keeps records of the oft-occurring CPR courses facilitated by the Training department. This includes the ACTIVITY name, the DATE\_ it occurs, and the number of ATTENDEES that day.

**STAFF:** The CPR unit needs the ability to keep records of staff and their corresponding certifications upon completing the various CPR courses. This table is for maintaining records of the staff members themselves. The fields are ID – the doctor's official employee ID number, NAME, TITLE – the doctor's position (consultant, physician, etc), and DEPARTMENT – which department they work in.

**CERTIFIED\_STAFF:** This table maintains the certification records of staff members. The fields are SN, ID, COURSE – which CPR activity they are certified in (the employee can be certified in multiple activities), STATUS, ISSUE\_DATE, EXPIRY\_DATE, NOTE, and STAMP.

#### **3.2.4.1.2 Data Dictionary**

The following is the definition of types of data in every table used by the TDS:

##### **MASTER\_TABLE**

```
( Name VARCHAR2(100) NOT NULL,  
  
Type VARCHAR2(100) NOT NULL);
```



## **USERS**

```
( Username VARCHAR2(15) NOT NULL,  
Password VARCHAR2(15) NOT NULL,  
Permission VARCHAR2(15) NOT NULL);
```

## **SYMP\_CONF**

```
( SN NUMBER(3) NOT NULL ,  
Activity VARCHAR2(100) NOT NULL,  
Date_ VARCHAR2(50) ,  
Venue VARCHAR2(50) ,  
SCFHS_Credit_Hours NUMBER(3) ,  
NO_of_Attendees NUMBER(5) ,  
Stamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

## **CME**

```
( SN NUMBER(3) NOT NULL ,  
Activity VARCHAR(100) NOT NULL ,  
Frequency VARCHAR(11) ,  
Total NUMBER(5) ,  
Venue VARCHAR(50) ,  
SCFHS_Credit_Hours VARCHAR(200) ,  
Stamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

## **NON\_MEDICAL\_COURSE**

```
(  SN NUMBER NOT NULL ,

Activity VARCHAR2(100) NOT NULL,

Date_ VARCHAR2(50) ,

Venue VARCHAR2(40) ,

Target_Group VARCHAR2(200) ,

NO_of_Attendees NUMBER ,

Stamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

#### **MEDICAL\_COURSE**

```
(  SN NUMBER NOT NULL ,

Activity VARCHAR2(100) NOT NULL,

Date_ VARCHAR2(50) ,

Venue VARCHAR2(40) ,

Target_Group VARCHAR2(200) ,

NO_of_Attendees NUMBER ,

Stamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

#### **TRAINING\_ACTIVITIES**

```
(  SN NUMBER NOT NULL ,

Activity VARCHAR2(100) NOT NULL,

Date_ VARCHAR2(50) ,

Entity VARCHAR2(200) ,

Type VARCHAR2(50),

NO_of_Trainees NUMBER,
```

Stamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP);

#### **TRAINEE**

( SN NUMBER NOT NULL ,  
Name VARCHAR2(100) NOT NULL,  
Department VARCHAR2(100) ,  
University VARCHAR2(100) NOT NULL,  
Start\_Date DATE NOT NULL,  
Duration VARCHAR2(50) ,  
Training\_Type VARCHAR(100) NOT NULL,  
Email VARCHAR2(50),  
Phone VARCHAR2(20),  
Status VARCHAR2 (20) ,  
Note VARCHAR2(300) ,  
Stamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP);

#### **CPR**

( SN NUMBER NOT NULL ,  
Activity VARCHAR2(100) NOT NULL,  
Date\_ DATE NOT NULL,  
Attendees NUMBER NOT NULL ,  
Stamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP);

#### **STAFF**

```
( ID VARCHAR2(10) NOT NULL,  
  
Name VARCHAR2(100) NOT NULL,  
  
Title VARCHAR2(50) ,  
  
Department VARCHAR2(100));
```

#### **CERTIFIED\_STAFF**

```
( SN NUMBER NOT NULL ,  
  
ID VARCHAR2(10) NOT NULL,  
  
Course VARCHAR2(100) NOT NULL,  
  
Status VARCHAR2(20) ,  
  
Issue_Date DATE ,  
  
Expiry_Date DATE ,  
  
Note VARCHAR2(300),  
  
Stamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

#### **3.2.4.1.3 Database Schema**

The purpose of the database schema is to show the associations and constraints between fields of different tables within the database. The diagram identifies the following constraints: Training\_Activities' Activity field references Master\_Table, Trainee's Department, University, and Training\_Type fields reference Master\_Table, Cpr's Activity field references Master\_Table, Staff's Department field references Master\_Table, and Certified\_Staff's ID field references Staff, and its Course field references Master\_Table.

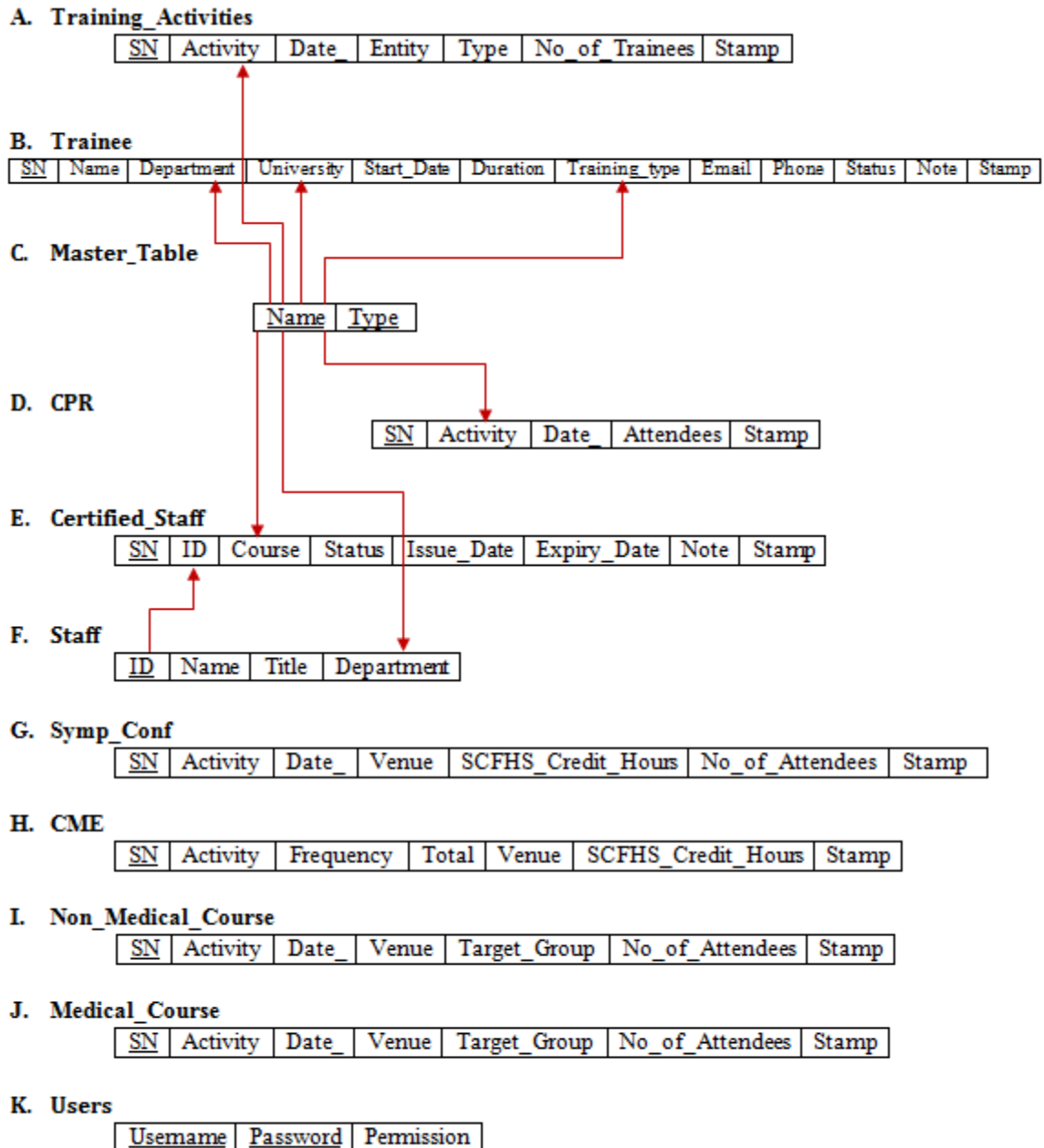
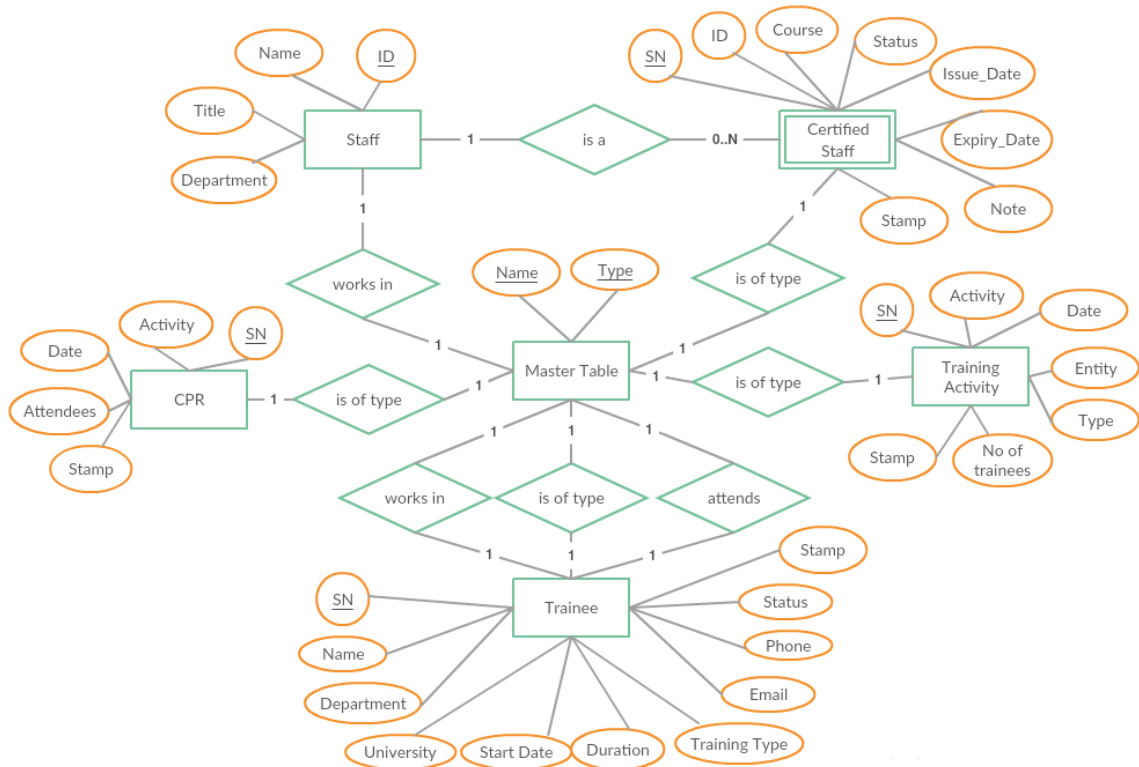


Diagram 3.1: Database Schema Diagram

#### 3.2.4.1.4 Entity Relationship Diagram

The entity relationship diagram is a visualization of the relationships and interactions between the tables in the database. The rectangles in the diagram represent entities, or tables, and the diamonds describe the relation between the entities. Attributes of an entity are depicted as ovals or circles.



**Diagram 3.2: Entity Relationship Diagram**

Aside from the relation between the Staff and Certified Staff entities, where a staff can be a certified staff member, all other relations in the diagram focus around the Master Table entity. This is because the purpose of the Master Table entity is to hold all data that signifies a relationship between tables. It has a compound primary key, Name and Type. The Type field consists of four different values throughout the table. The values are “Department”, “CPR”, “University” and “Training”. The relations throughout the database were designed in this way as a suggestion from one of the senior database administrators in the RCMC IT department.

When analyzing the diagram, the relation “Staff > works in > Master Table” is not immediately clear. However, when considering that one of the types in the Master Table is “Department”, the relation becomes clear. Next, “Certified Staff > is of type > Master Table” refers to the “CPR” type, meaning that a certification can be one of the predefined CPR types held in the table. The CPR entity’s relation with the Master Table entity is also referencing any “CPR” type. “Training Activity > is of type > Master Table” refers to the “Training” type, which includes all types of training categories held at the hospital.

The last, “Trainee > works in AND is of type AND attends > Master Table” is simultaneously referring to the “Department”, “Training”, and “University” types held in the Master Table entity.

### 3.2.4.2 System Design

Designing the TDS involved first developing diagrams that illustrate the responsibilities of users and the actions the system is intended to go through. Because of the various different user types involved, it was important to document a use case diagram. Additionally, a navigational chart was documented in order to clarify the steps involved in moving around the application, as well as an activity diagram to show what the system is capable of.

#### 3.2.4.2.1 Use Case Diagram

The following use case diagram details the basic capabilities of each user in the system.



**Diagram 3.3: Use Case Diagram**

### 3.2.4.2.2 Navigational Diagram

The navigational diagram allows one to visualize the structure of the application, abstracted from graphical complications.

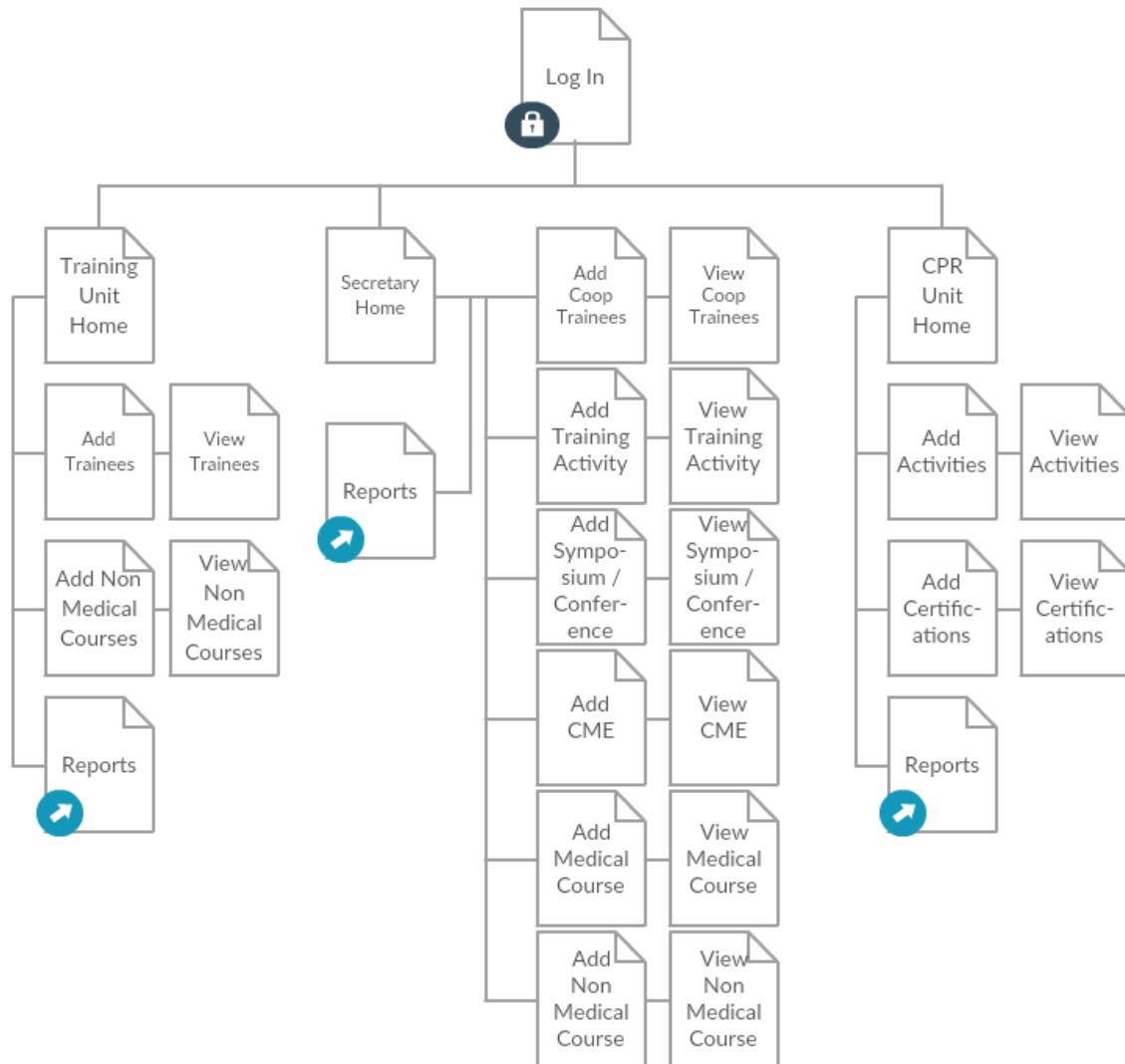
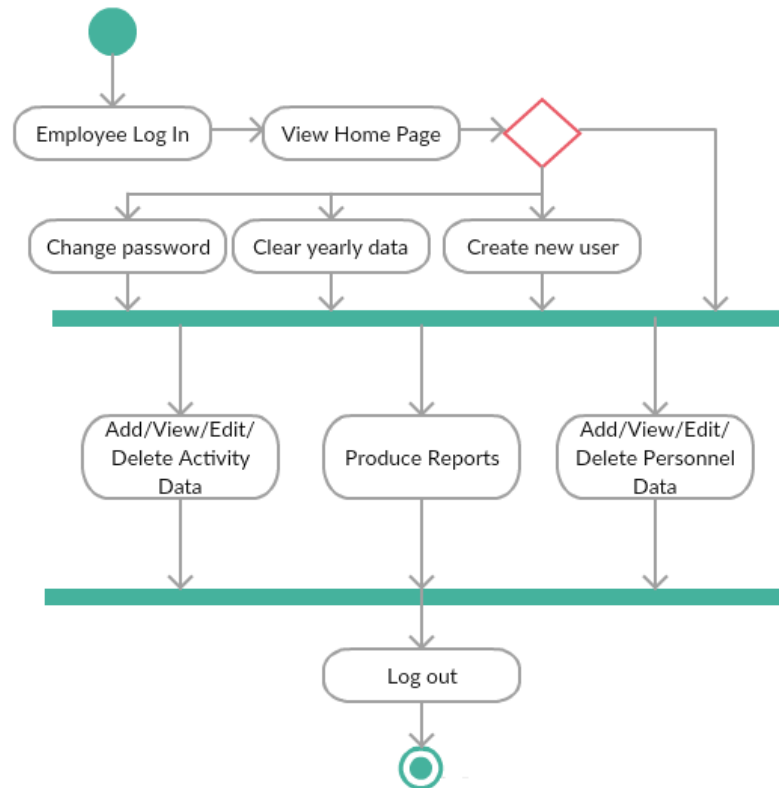


Diagram 3.4: Navigational Diagram

### 3.2.4.2.3 Activity Diagram

The following diagram is a super-simplified activity diagram, which serves the purposed of showing the flow of capabilities available in the system. It is super-simplified in order to highlight what the system intends for its users, without distracting away from this point by including each high-level detail.





**Diagram 3.5: Activity Diagram**

The diagram begins with employees logging in, which is necessary to access the system. The next logical step is viewing the homepage, where the user has the decision to change their password, create a new user, or clear their data. From there, the directions include the neat summarization of adding, viewing, deleting or editing activity related data. For the training unit user, this means the non medical activities, and for CPR, it means CPR classes. For the secretary, it includes the five activities under her responsibility. The direction can also be to add, view, delete or edit personnel data. For the training unit, personnel means all trainees. For the secretary, just coop trainees, and for CPR, personnel means staff. In this way, the diagram is abstracted outside of the constraints of each user to display the functionality of the system.

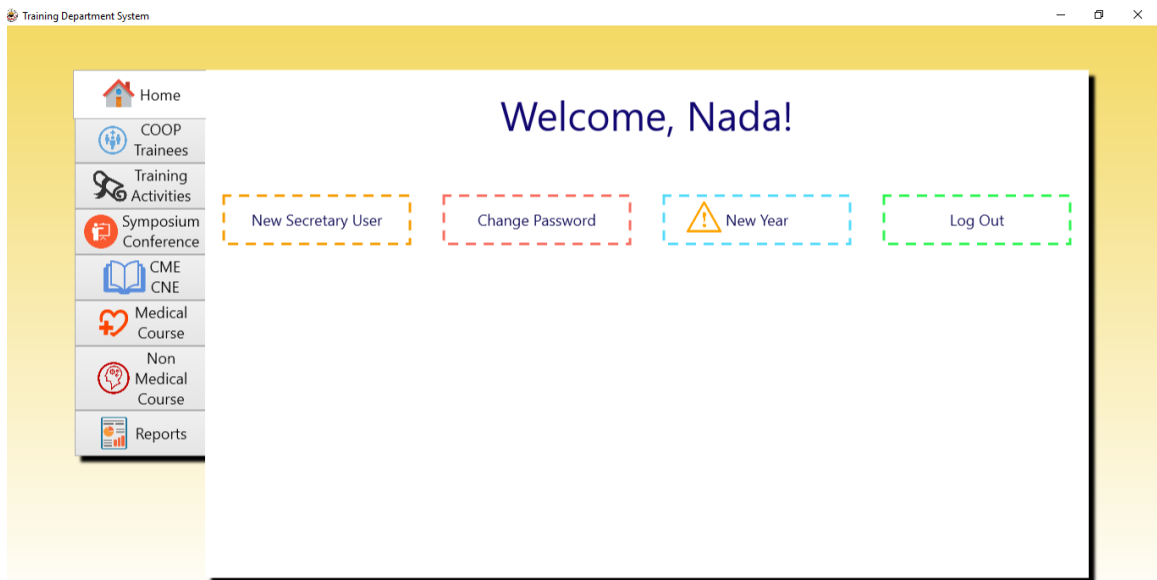
### 3.2.5 User Interface

The following figures are screenshots from the application as it is run and used.



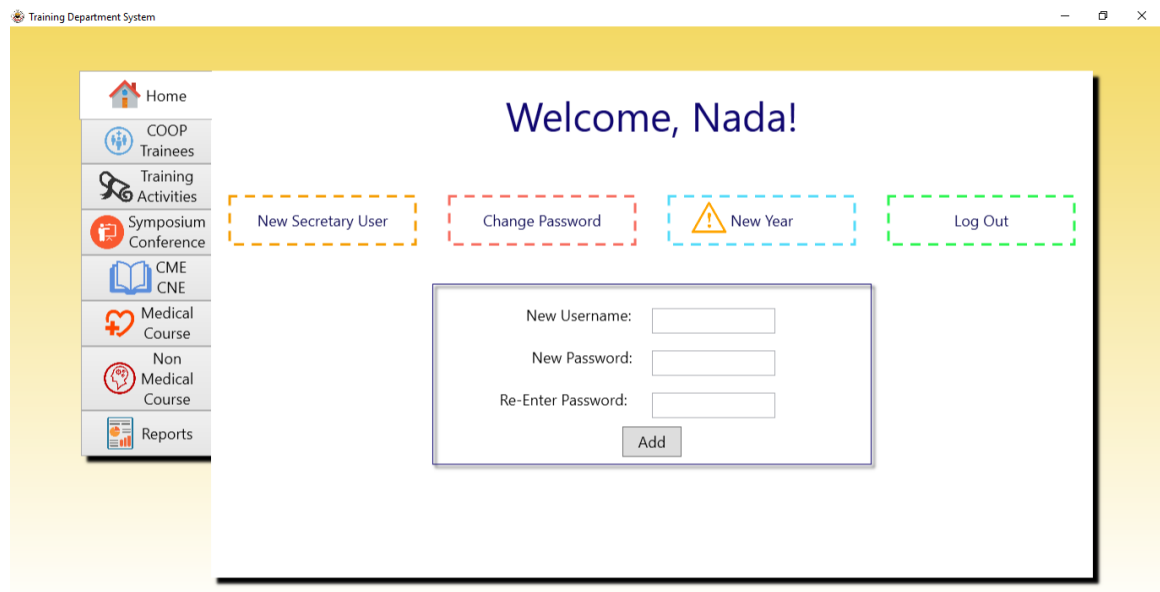
**Figure 3.3: Start Page**

Figure 3.3 is the first page seen when starting the program. The username and password boxes are programmed to immediately have the text in them highlighted when the box gains focus. That way, the user can immediately start typing over the existing text without having to use the mouse or delete what's in the boxes. All text boxes in the application have this feature.



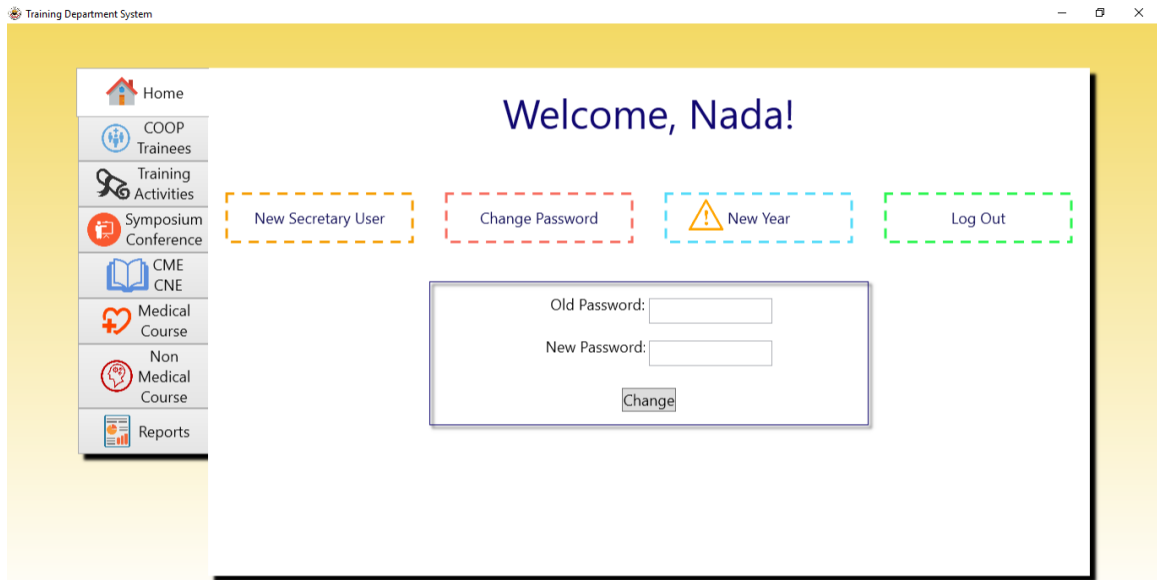
**Figure 3.4: Secretary View – Home Tab**

After logging in, each user will see a variation of Figure 3.4. The application uses Tab Controls for navigation. Figure 3.4 is the secretary view, so all the tabs on the left are specific to secretary needs. However, all users will see the same home tab.



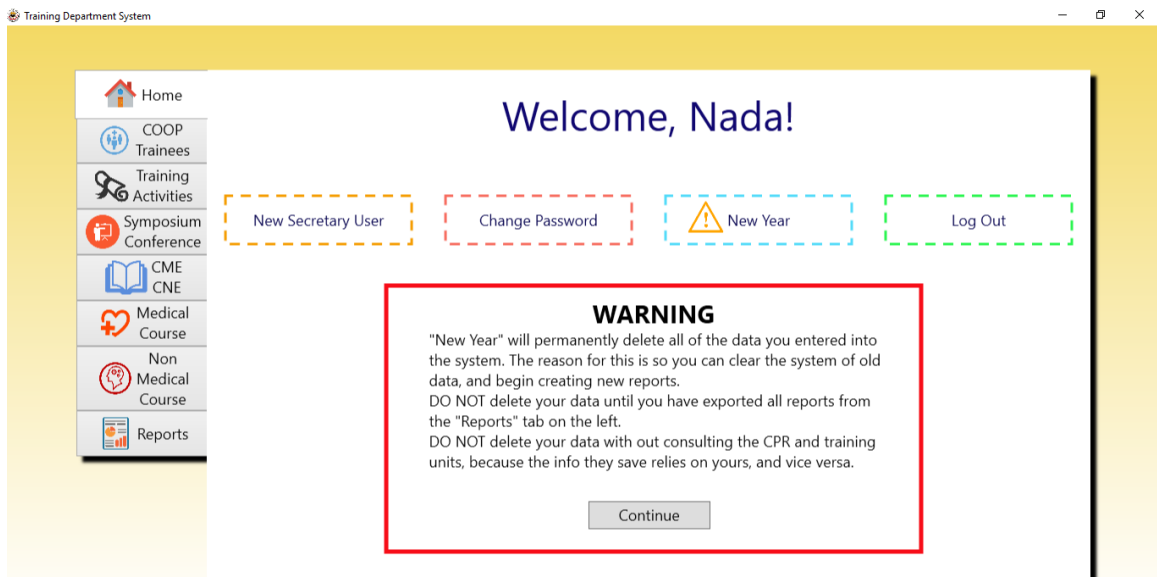
**Figure 3.5: New User**

Each user has the capability to create a new user of their own type. For example, if a new user is created from the Secretary view, that user will only have access to the Secretary view, and not the CPR or Training unit views. Figure 3.5 illustrates that to create a new user, you must give them a username and a password, and then confirm the password.



**Figure 3.6: Change Password**

Figure 3.6 shows that users are allowed to change their passwords.



**Figure 3.7: New Year**

The “New Year” button shown in Figure 3.7 is meant to only be used once a year, after the secretary has produced her reports. It deletes all the data for each topic (the Training unit and CPR unit both have the same button for their data, but the CPR unit’s button does not delete staff certifications, as those persist beyond a year). The reason for this is because the reports are year-sensitive, and if the secretary wants accurate reports for a

year, all previous data should be eliminated so it doesn't show up in the sums and counts of the annual report.

For example, if the secretary finishes her report for 2016, and wants to start reporting on 2017, but the CPR unit hasn't erased all their data from 2016, the secretary's 2016 report will be convoluted with the old values, and won't show the accurate CPR course data for that year. As one can see, this is a delicate procedure that requires coordination beyond the software system, and between users in the real world.

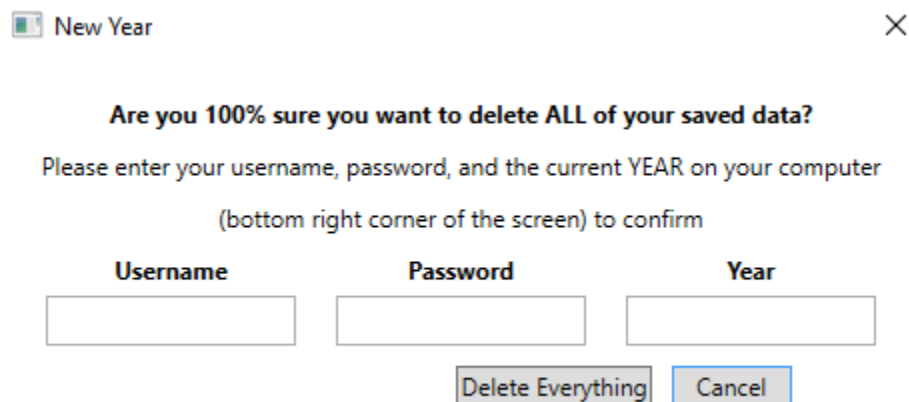


Figure 3.8: New Year Confirmation Dialog

Figure 3.8 is the dialog box that will show up if a user presses the "Continue" button in the previous image. To prevent users from accidentally deleting all their data, this step requires some arbitrary fields to be filled in to prove that the user is paying attention and actually wants to delete everything from the system. As one can see, the focus is set on the cancel button by default, so if the Enter key is pressed by accident, the user won't lose everything (another precaution).

Training Department System

## New COOP Trainee

Name\*  Dept\*  University\*

Start Date  Duration  Training Type\*  [+ View](#)

Email  Phone  Status

Note

**Figure 3.9: Adding COOP Trainee**

Moving down the tabs on the left of the screen, we reach COOP Trainees, Figure 3.9. There are many types of trainees, but the secretary is only responsible for COOP trainees. The user enters the trainee's name, department, university, start date, duration of training period, training type, email address, phone number, status and a note about the trainee if they want. Name, department, university and training type are all required fields. Name is required because trainees need to be distinguished from one another. Department is required because a trainee cannot be enrolled to train without a department. University is required because the Training unit needs to know if the trainee is from a public or private university and ultimately whether they should be paid or not. And training type is necessary to count the number of each type of trainee. The other fields are helpful but not necessary, and can be added at a later time.

**Figure 3.10: Dropdown Lists**

Figure 3.10 shows the drop down lists for the important fields previously mentioned. Because the data from these fields is shared throughout different topics in the application, and even between users, it is necessary to maintain its integrity by only allowing predefined names to be used. This is because even a slight difference in spaces or capitalization could affect the integrity of a string. All of the values shown here exist in the MASTER\_TABLE of the database (see section 3.2.3). They cannot be deleted once added because other tables rely on these values as foreign key constraints. The boxes are searchable.

**Figure 3.11: New University Dialog**

The only one of these combo boxes that can have a new value added to it is University. However, if the user enters a new university name that has not been used before, the

system will prompt them to specify whether it is public or private. It will then be saved to use again and again without having to see the dialog box after the first time.

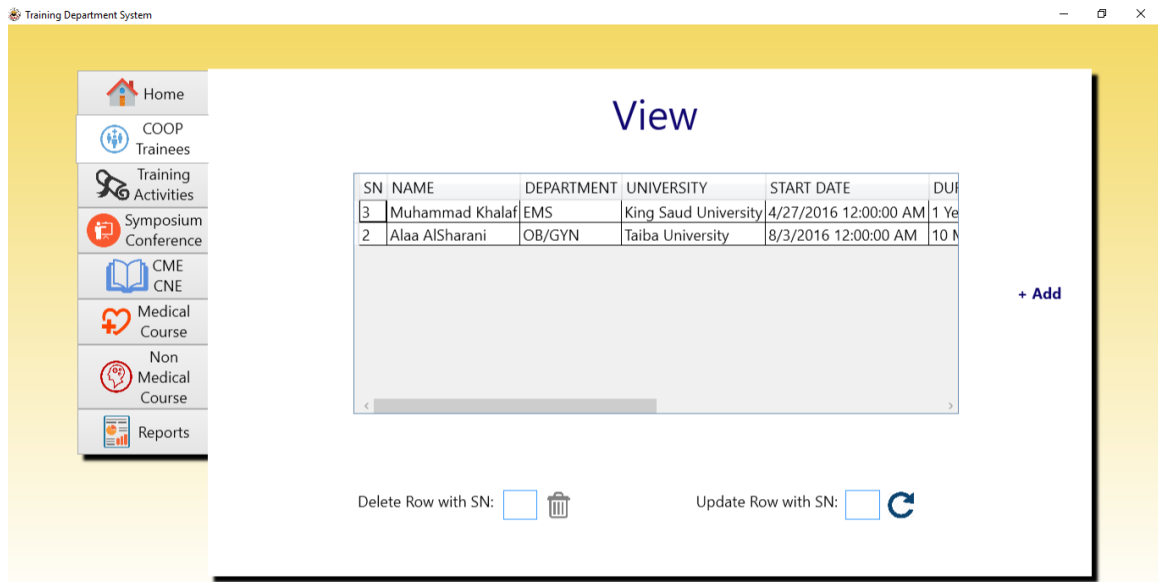
The image shows a portion of a web form. On the left, there are labels 'Start Date' and 'End Date' (partially visible). The 'Start Date' field has a dropdown menu open, showing a calendar for April 2016. The date 15 is selected in the header, and the date 24 is highlighted in the calendar grid. To the right of the calendar is a 'Status' dropdown menu. The dropdown is open, showing three options: 'Active', 'Terminated', and 'Complete'.

Su	Mo	Tu	We	Th	Fr	Sa
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

**Figure 3.12: More Dropdown Lists**

These dropdown lists in Figure 3.12 are not as sensitive as the previously mentioned ones. The first is the drop down calendar, so users can enter dates in a predefined format and the system doesn't have to have any trouble fitting it to its own date format expectations. The second is the status drop down. It is not vital information, but it is important that the user be able to organize trainees by this field, so the statuses available should be consistent. The system does this by collecting all the statuses already written in the trainee table, and listing them in this box. If the user wants to enter a new status, they are free to do so, but the system will recommend them an existing one if it is similar to the new entrance.





**Figure 3.13: COOP View**

The view shown in figure 3.13 is the same for every tab and every topic in the secretary application. It simply shows the contents of that topic's table in the database for the user to review. The user can copy parts-of or all-of the table and paste it in Excel if they want. They can also click the column names at the top row, and the table will be organized by that column. For example, if one were to click "Department", the trainees would be listed and grouped by their departments. Clicking on "Start Date" would organize the trainees from newest to oldest; clicking again, from oldest to newest. Imagine hundreds of entries in this table; the ability to sort and copy and paste would be very helpful.

DURATION	TRAINING TYPE	EMAIL	PHONE	STATUS	NOTE
1 Year	Cooperative Training	muhammad141@hotmail.com	5806779011	Complete	
10 Months	Cooperative Training	a.sharani@gmail.com	0557832234	Active	

**Figure 3.14: Remaining Columns in COOP View**

Notice a horizontal scroll bar in Figure 3.13; scrolling right reveals the remaining columns of the table shown in 3.14.

Update

Name: Muhammad Khalaf    Dept: EMS    University: King Saud Univer

Start Date: 4/27/2016    Duration: 1 Year    Training Type: Cooperative Train

Email: muhammad141@hotmail.com    Phone: 5806779011    Status: Complete

Note:

Ok    Cancel

SN    NAME    DEPT    UNIVERSITY    START DATE    DURATION

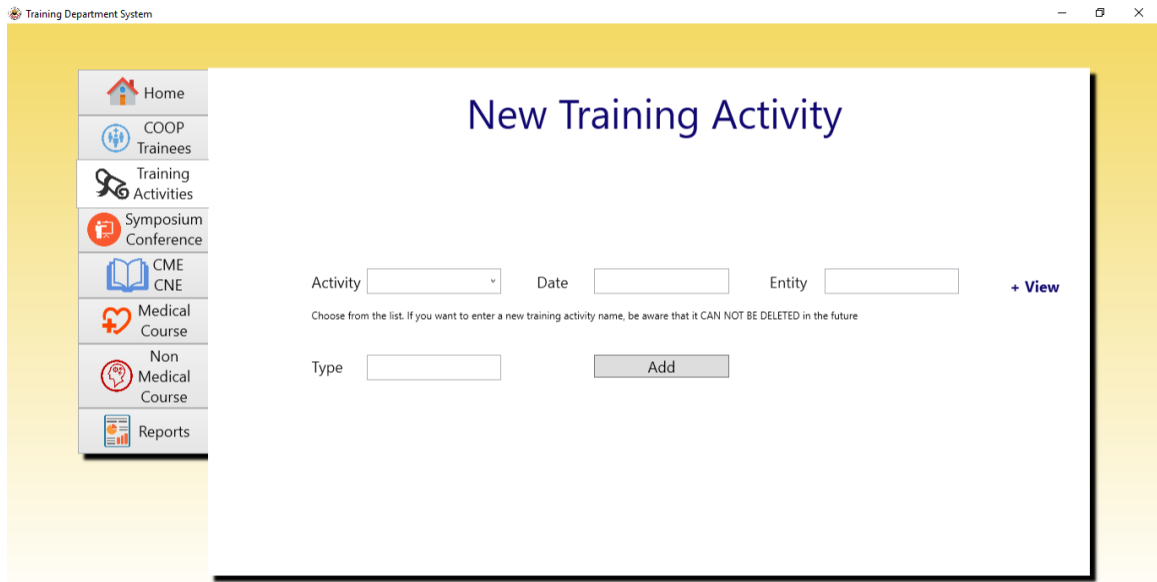
3	M	EMS	King Saud Univer	4/27/2016 12:00:00 AM	1 Year
2	A	EMS	King Saud Univer	8/3/2016 12:00:00 AM	10 Months

Delete Row with SN:

Update Row with SN:

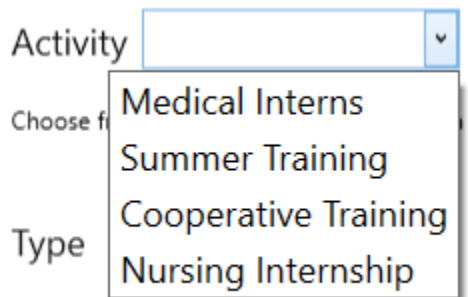
**Figure 3.15: Update Dialog**

Figure 3.15 is the result of entering a valid SN number from the table and clicking the update button (entering a valid SN in the delete box will simply prompt a message box stating “Deleted Successful”). This dialog is the same for every table, besides obviously containing the fields for that table. As we can see, all the fields of the COOP Trainee are shown in the dialog, already filled with the data of the SN the user selected. This is helpful so the user doesn’t have to re-enter every bit of information; they can just edit the bits that need editing. It is also helpful if the user didn’t have all of the trainee’s info at the time of adding them to the list. If the user didn’t know the trainee’s email but later found it, they could add it using the update dialog.



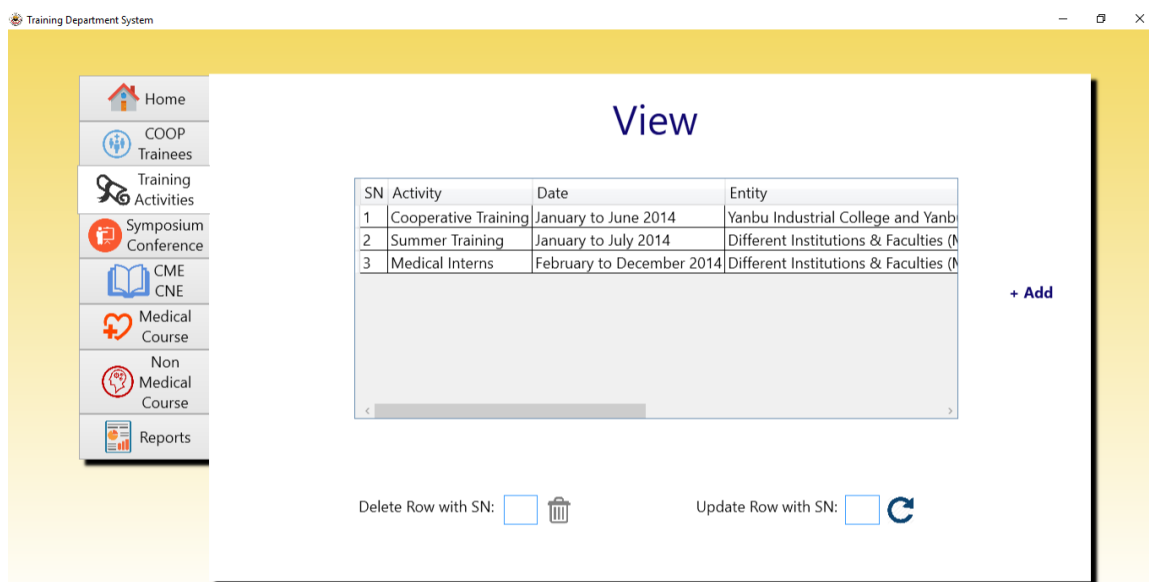
**Figure 3.16: Training Activities Tab**

Figure 3.16 is the tab for adding training activities. This is where the secretary can add a new activity to the MASTER\_TABLE, so that it will show up in the “Training Type” list when adding trainees. This is also where the secretary enters data about each training type to be mentioned in the report. In the “View” for training activities, there is a column in the table where one can see the number of trainees in the system for each training type.



**Figure 3.17: Training Activities Dropdown List**

Training activities are stored in the MASTER\_TABLE, so they appear in this drop down list in Figure 3.17 to be added to the training activities table. The note beneath the field states: “Choose from the list. If you want to enter a new training activity name be aware that it CAN NOT BE DELETED in the future”. If the user enters a new activity in the dropdown list, it will be saved to the MASTER\_TABLE for future use.

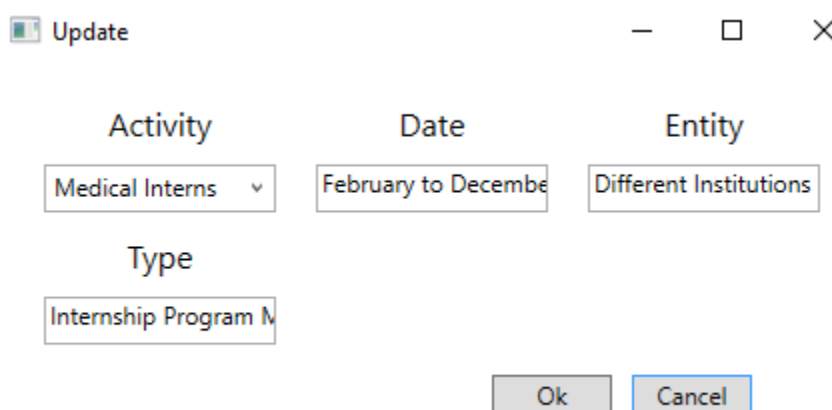


**Figure 3.18: Training Activities View**

	Type	Trainees
of Technology	Engineering	2
Non-Medical) Private & Government	Medical & Non-Medical Departments	2
Non-Medical) Private & Government	Internship Program Medical Departments	1

**Figure 3.19: Remaining Columns in Training Activities View**

In the “View” for training activities (Figure 3.18), there is a column in the table where one can see the number of trainees in the system for each training type (Figure 3.19).



**Figure 3.20: Update Dialog for Training Activities**

Figure 3.20 shows the update dialog for rows in the training activities view. As one can see, the fields are specific to the data present in the table. Also notice that the “Trainees”

column (Figure 3.19) is not updateable. This is because that data is automatically managed by the system and does not require user tampering.

Training Department System

Home  
COOP Trainees  
Training Activities  
Symposium Conference  
CME CNE  
Medical Course  
Non Medical Course  
Reports

## New Symposium/Conference

Activity:  Date:

Venue:  Attendees:  [+ View](#)

SCFHS Credit Hours:

**Figure 3.21: Symposium/Conference Tab**

Training Department System

Home  
COOP Trainees  
Training Activities  
Symposium Conference  
CME CNE  
Medical Course  
Non Medical Course  
Reports

## View

SN	Activity	Date	Venue
1	1st Basic Laparoscopic Surgery Updates Symposium	23-24 May	Movenpick Hote
2	Workplace Based Assessment Symposium	28-29 May	Basement Confe
3	2nd Clinical Nutrition Conference	29-30 April	Al-Murjan Recre
4	Guidelines to Smoking Cessation Symposium	18 June	Basement Confe
5	4th RCMC Infection Control Conference	12-22 October	King Fahed Cultu

[+ Add](#)

Delete Row with SN:

Update Row with SN:

**Figure 3.22: Symposium/Conference View**

Venue	SCFHS Credit Hours	Attendees
Movenpick Hotel - Yanbu	15	38
Basement Conference Room	10	28
Al-Murjan Recreation Center - Yanbu	15	500
Basement Conference Room	5	95
King Fahed Cultural Center	15	581

**Figure 3.23: Remaining Columns in Symposium/Conference View**

Figures 3.21, 3.22 and 3.23 show the Symposium/Conference tab, where new events can be added, the Symposium/Conference view, and the remaining columns that can be seen when scrolling right in the view. Nothing of particular interest merits discussion on this topic, because it exists solely for the secretary's report and nothing else. It also has the same kind of update dialog as the other previously mentioned ones.

Training Department System

Home

COOP Trainees

Training Activities

Symposium Conference

CME CNE

Medical Course

Non Medical Course

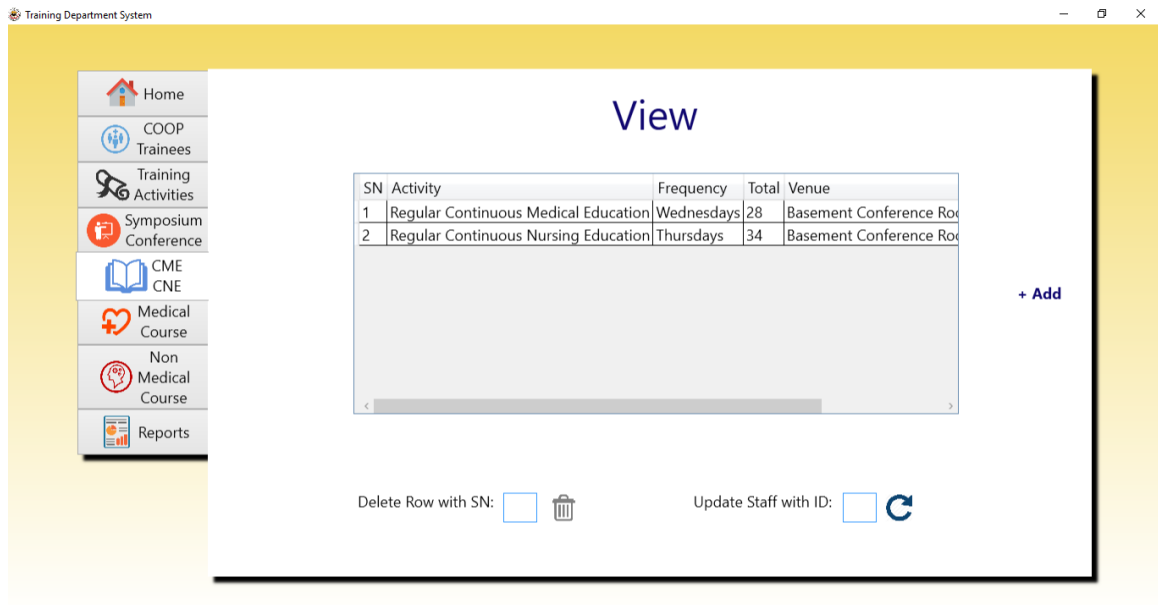
Reports

## New CME/CNE

Activity  Frequency  Total  [+ View](#)

Venue  SCFHS Credit Hours

**Figure 3.24: CME Tab**



**Figure 3.25: CME View**

Venue	SCFHS Credit Hours
Basement Conference Room	12
Basement Conference Room	8

**Figure 3.26: Remaining Columns in CME View**

Figures 3.24-3.26 detail the Continuous Medical Education Tab. It also does not interact with other data and only exists for the sake of the annual report.

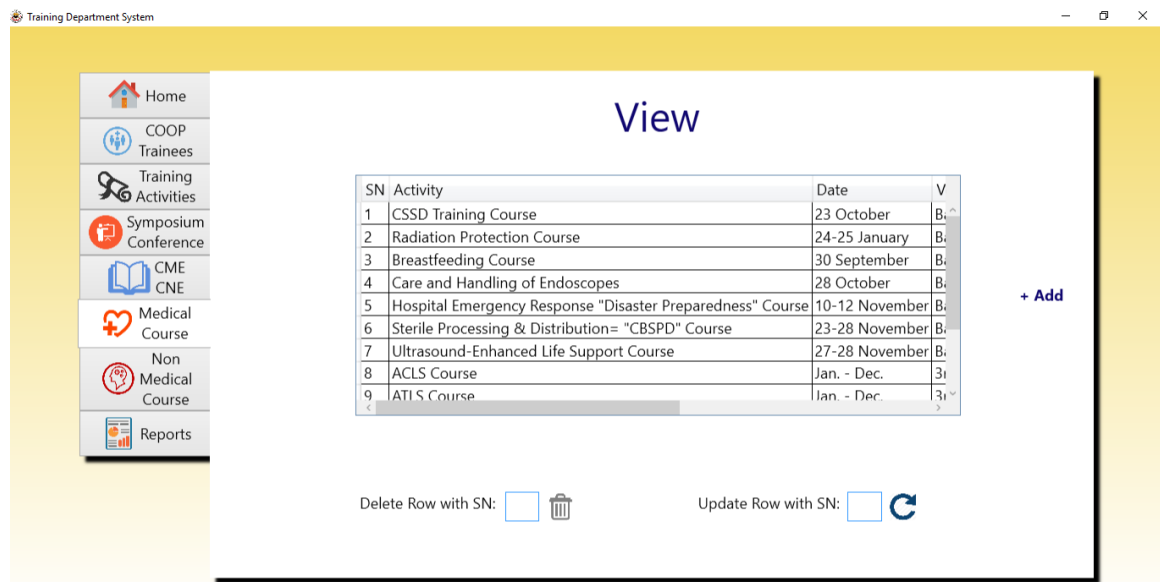
**Figure 3.27: Medical Course Tab**

The Medical Course tab is for the secretary to add all of the medical type courses delivered at the hospital throughout the year. This includes CPR courses. Because CPR course names are being shared between users and across tables, their titles exist in the MASTER\_TABLE. But the Medical Course topic includes other, non-CPR courses as well. So the user has the option to choose a CPR course from the list (Figure 3.28), maintaining the integrity of its title, or entering a new, non-CPR course. The note below “Activity” reads: “Choose a CPR course from the list, or add a new (non-CPR) course”. The second note, under “Attendees” reads: “NOTE: Do not fill in ‘Attendees’ field for CPR courses, they will update automatically”. This is because the “Attendees” field for a CPR course is the sum of all attendees for that course in the CPR user’s data.

**Figure 3.28: Dropdown List for CPR Courses**



The list of CPR courses available to the secretary user. If the secretary does not choose a CPR course from this list and instead writes their own CPR course name, the attendees will never be automatically updated for that course.

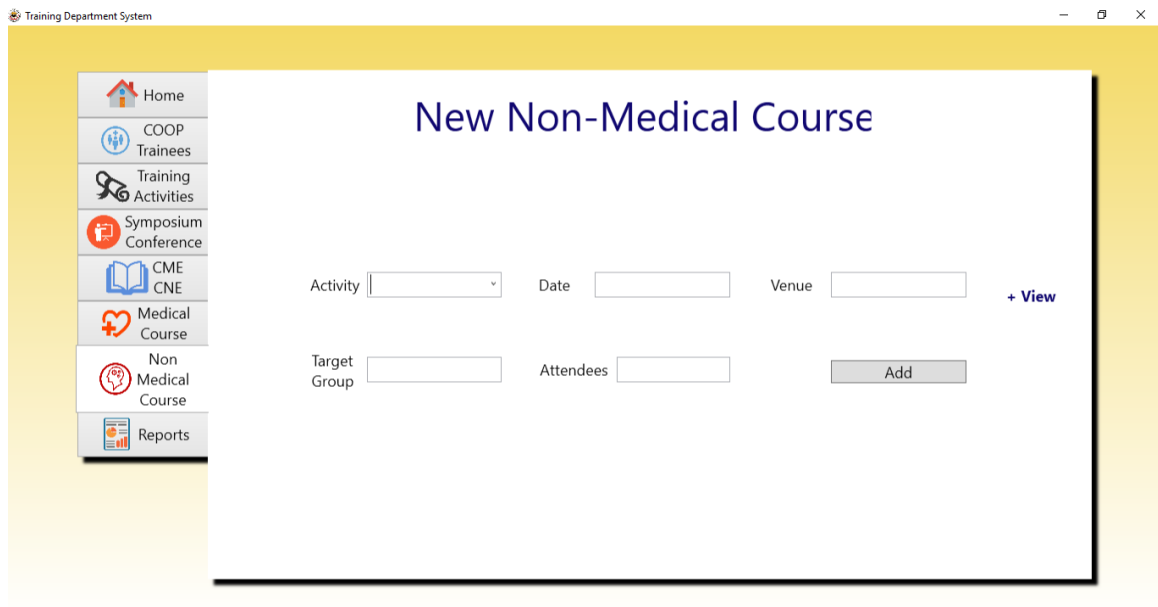


**Figure 3.29: Medical Course View**

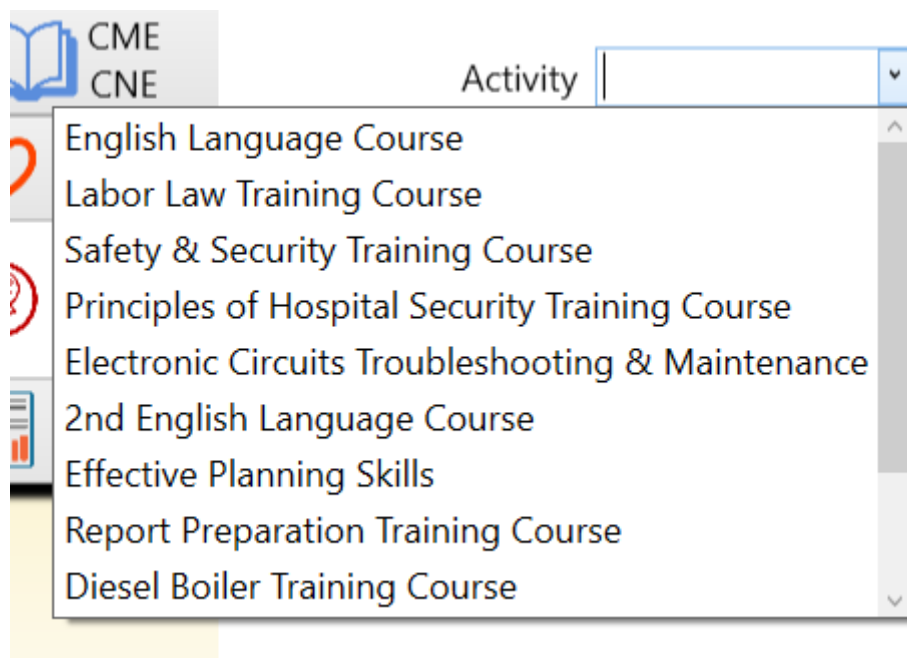
Venue	Target Group	Attendees
Basement Conference Room	CSSD Staff, Nurses & Technicians	128
Basement Conference Room	Doctors, Nurses, Technicians	200
Basement Conference Room	Doctors, Nurses	258
Basement Conference Room	Nurses	24

**Figure 3.30: Remaining Columns in Medical Course View**

Figure 3.29 is the view for the Medical Course tab, and Figure 3.30 shows the remaining columns not shown without scrolling right. It has the same style update dialog as other tables.



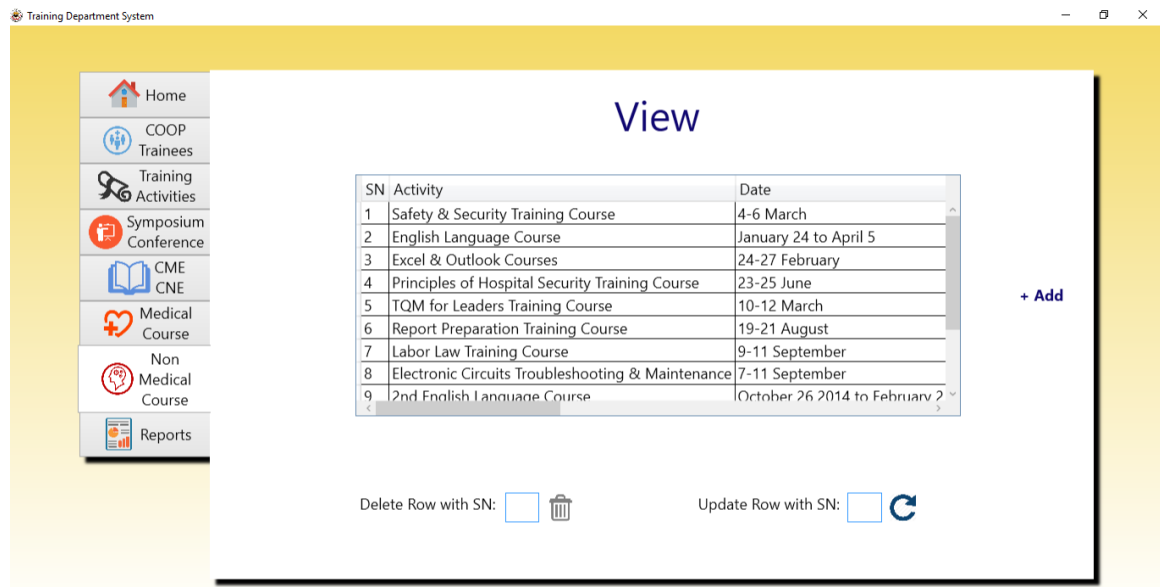
**Figure 3.31: Non Medical Course Tab**



**Figure 3.32: Activity Dropdown List for Non Medical Course**

Figure 3.31 shows the tab for adding new non-medical courses to the report. The list in figure 3.32 shows all the course names already in the table. This is so, when multiples of the same courses occur, the user can select the name without retyping it. It also serves the purpose of keeping multiples of the same class in sync with the same name, so when the

report is created, it can count how many times the class occurred and how many attendees attended all the classes combined.



**Figure 3.33: Non Medical Course View**

Venue	Target Group	Attendees
Classroom 2	Safety & Security	26
Classroom 1	RCMC Staff	71
Computer Lab	RCMC Staff	50
Classroom 2	Safety & Security	25

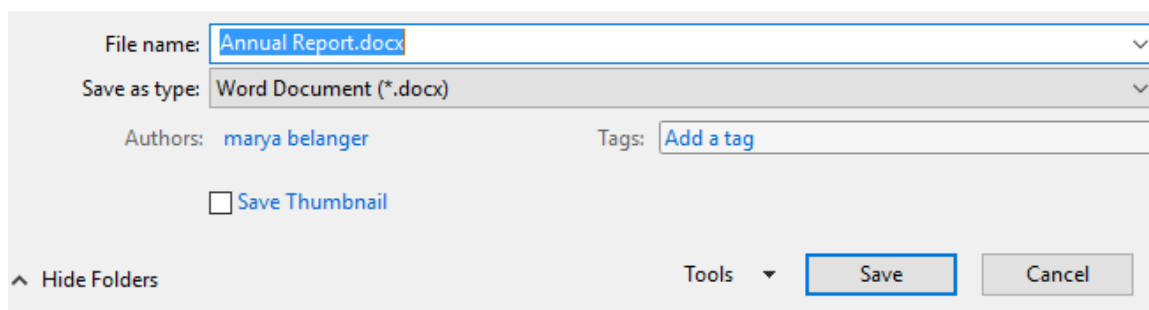
**Figure 3.34: Remaining Columns of Non Medical Course View**

Figures 3.33 and 3.34 show the view for non-medical course and the remaining columns to the right of the visible table. It has the same style update dialog as the other tabs.



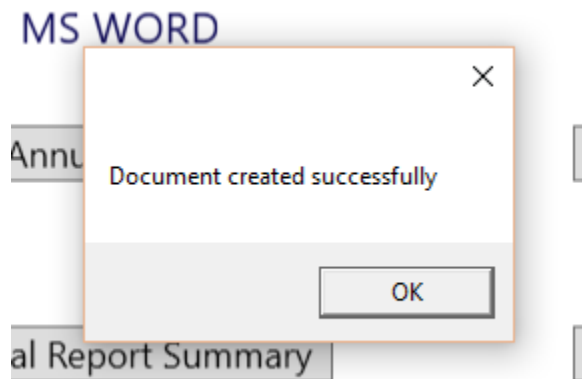
**Figure 3.35: Reports Tab**

The final tab is the reports tab, where all the data from the previous tabs is compiled into custom reports for the secretary. There are four report options: Annual Report, Annual Report Summary, Annual Report Charts, and Coop Trainee Report. The note at the bottom reads “Reports will be saved in the ‘Documents’ folder” to save users the trouble of finding the report after it has been created. The Annual Report displays all the data of the previous tabs (except the COOP Trainees tab) and styles it in easy to read tables. The Annual Report is also an aesthetically pleasing collection of tables, but contains the meta-data of the Annual Report, and mostly totals all the data for each table. The Annual Report Charts file transfers all the data from Annual Report to Excel, and creates graphs based on any numeric information in the tables. And lastly, the Coop Trainee report is a simple Excel file of all the trainee data.



**Figure 3.36: Saving the Annual Report**

If the user clicks the “Annual Report” button, a save-document dialog will open in the documents folder and allow the user to change the name of the file if they want, or leave it as “Annual Report.docx”. They can also change the location of the file from this dialog.



**Figure 3.37: Report Success**

If the user saves the report and there are no other errors, then the message in Figure 3.37 will pop into the screen.

SN	Activity	Date	Year	SCFEB Credit Hours	Attendees
1	1st Basic Laparoscopic Surgery Options Conference	23-24 May	Microscopic Mini-Lapda	15	38
2	Workplace Based Assessment Symposium	28-29 May	Simulation Conference Room	10	28
3	2nd Central Venous Catheter Conference	29-30 April	AC Medical Simulation Center - Lapda	15	100
4	Workshop to Simulating Cerebral Symptom	18 June	Simulation Conference Room	5	95
5	4th SCFEB Conference Control Conference	12-21 October	King Fahd Cultural Center	15	581
<b>Grand Total</b>				<b>60</b>	<b>1242</b>

SN	Activity	Date	Faculty	Type	Trainers
1	Cooperative Training	January to June 2014	Yasda Industrial College and Yasda College of Technology	Engineering	2
2	Summer Training	January to July 2014	Yasda Industrial College and Yasda College of Technology	Medical & Non-Medical Personnel	2
3	Medical Internship	February to December 2014	Yasda Industrial College and Yasda College of Technology	Medical & Non-Medical Personnel	1
<b>Grand Total</b>					<b>5</b>

SN	Activity	Frequency	Total	Year	SCFEB Credit Hours
1	Regular Continuous Medical Education	Tuesday	25	Simulation Conference Room	12
2	Regular Continuous Medical Education	Thursday	34	Simulation Conference Room	8

**Figure 3.38: Sample of the Annual Report**

Figure 3.38 shows the first three pages of the Annual Report opened in Microsoft Word. These pages contain the tables for Symposium/Conference, Training Activities, and Continuous Medical Education.

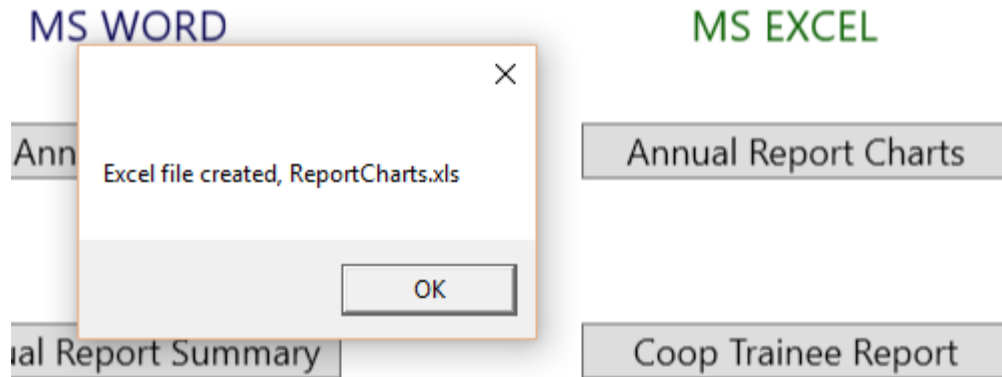


Figure 3.39: Report Chart Success

If the user clicks the “Annual Report Charts” button, and there are no errors, the message in Figure 3.39 will appear.

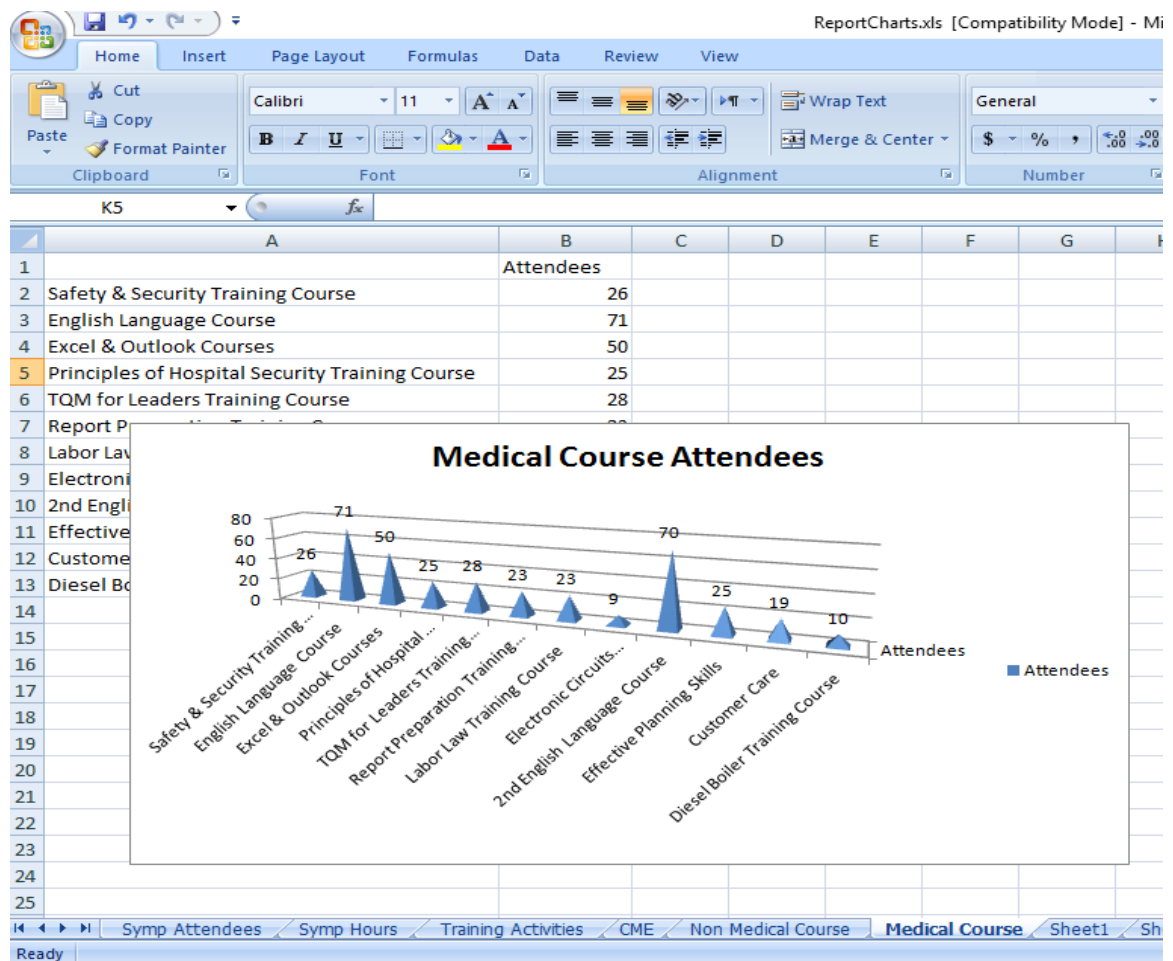
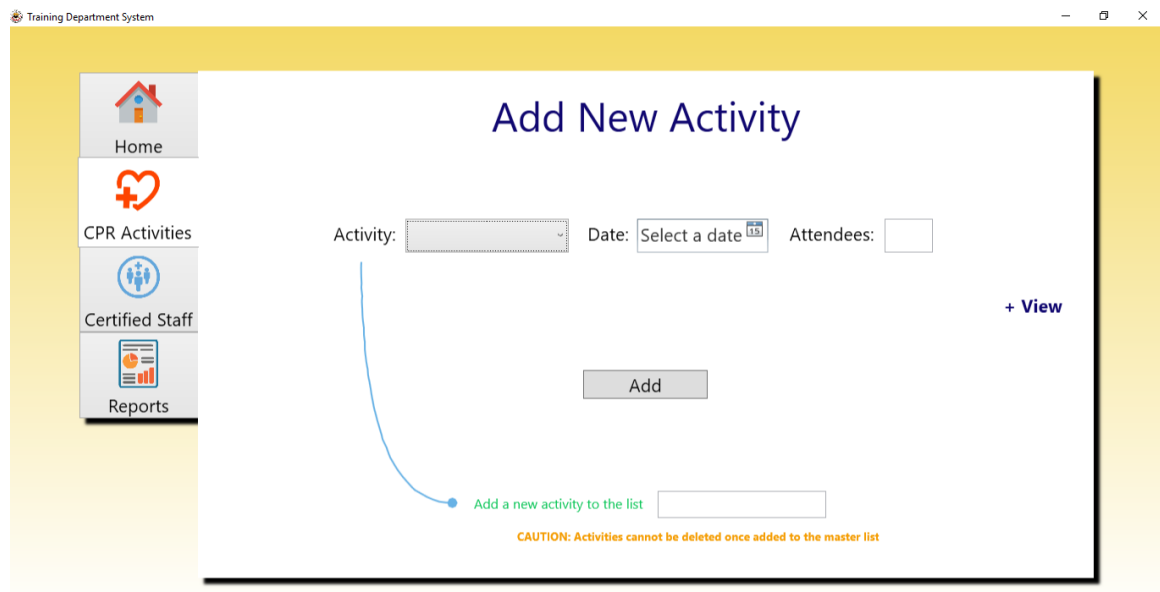


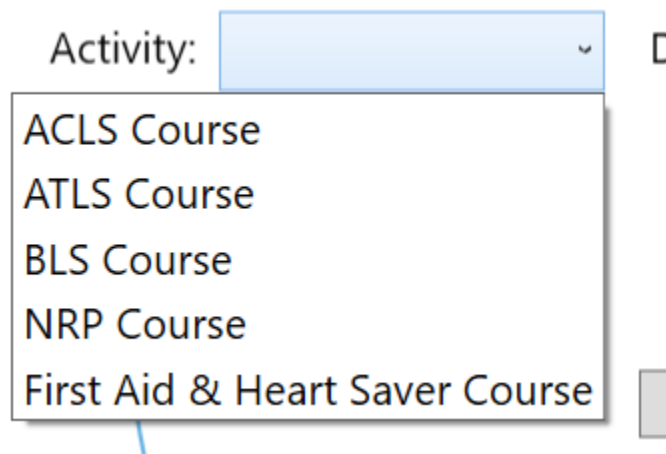
Figure 3.40: Sample of the Report Charts

Figure 3.40 is a sample of all the charts created for the Annual Report. The graph is automatically produced by the system, and is a product of the numeric data in each table. The charts are meant to be used by the secretary in the final draft of the Annual Report.



**Figure 3.41: CPR Activities Tab**

As previously mentioned, the Home tab is the same for all users. The differences between user views are the other tabs on the left of the screen. In figure 3.41, we examine the CPR user's CPR Activities tab. This is where the CPR unit adds data on the oft-occurring CPR courses each time they occur. The user chooses which CPR course occurred, one what date, and how many attendees.



**Figure 3.42: CPR Activities Dropdown List**

Figure 3.42 is the list of activities available to choose from. As mentioned before, their titles are saved in the MASTER\_TABLE to maintain their spelling, punctuation, etc.

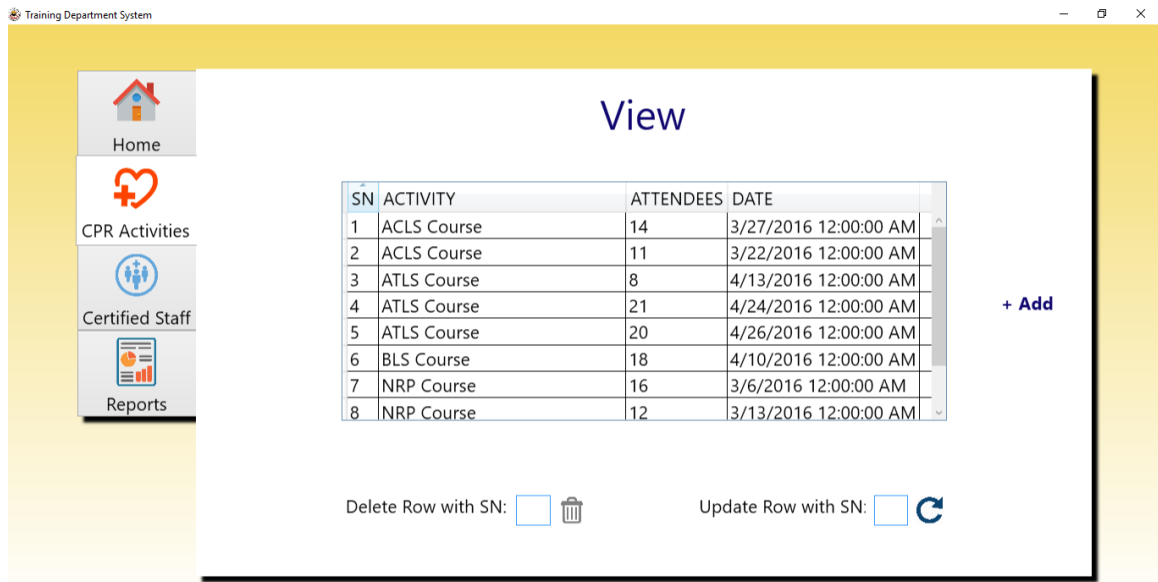
The screenshot shows a web application interface. On the left, there is a vertical sidebar with a button labeled "Add a new activity to the list" in green text. A blue curved arrow points from this button to a modal dialog box titled "Are You Sure?". The dialog box contains a warning icon (a yellow triangle with an exclamation mark) and the following text: "A new activity cannot be deleted once it is created", "Please re-enter the new activity to verify its name", and "Re-enter new activity name" (which is a text input field). Below this, it says "Enter additional information about the course (Optional)" and lists three fields: "Venue", "Date", and "Group", each with a corresponding text input box. At the bottom of the dialog are "Ok" and "Cancel" buttons. To the right of the dialog, there is a label "Attendees:" followed by a large empty rectangular box. Below the "Add a new activity to the list" button, there is a text input field containing the text "PPR Course".

**CAUTION: Activities cannot be deleted once added to the master list**

Figure 3.43: Adding New CPR Course

If the user wants to add a new CPR course title to the list (to MASTER\_TABLE), Figure 3.43 shows what will happen. The user can enter the new name at the bottom of the screen, and will then be prompted to confirm the spelling of the title. This is important since it cannot be deleted later. The CPR user can also fill in the Venue, Date, and Group for the CPR course, to be sent to the Medical Course table in the secretary's view. This is optional, because the secretary can also just fill in that information herself.





**Figure 3.44: CPR Activities View**

Figure 3.44 is the view of the CPR Activities table. Like all other tables, the view can be sorted by any of the columns listed. Each time an activity is added, deleted or updated in the CPR view, the Medical Course table is updated in the secretary's application, as we see in Figure 3.45.

8	ACLS Course	Ja	3r	Ph	25
9	ATLS Course	Ja	3r	Ph	49
10	BLS Course	Ja	3r	Ph	18
11	NRP Course	Ja	3r	Ph	28
12	First Aid & Heart Saver Course	Ja	3r	Ph	100

**Figure 3.45: Updated Attendees Amount in Medical Course View**

SN	ACTIVITY	ATTENDEES	DATE
1	ACLS Course	14	3/27/2016 12:00:00 AM
2	ACLS		22/2016 12:00:00 AM
3	ATLS		13/2016 12:00:00 AM
4	ATLS		24/2016 12:00:00 AM
5	ATLS		26/2016 12:00:00 AM
6	BLS		10/2016 12:00:00 AM
7	NRP Course	16	3/6/2016 12:00:00 AM
8	NRP Course	12	3/13/2016 12:00:00 AM

Update CPR Activity

Activity

NRP Course

Date


3/13/2016


Attendees

12

Ok

Cancel

Delete Row with SN:  

Update Row with SN:  

**Figure 3.46: CPR Activities Update**

As previously mentioned, the Medical Course table updates when any row in the CPR Activities table is updated, because it is possible to change the number of attendees from the update dialog, seen in Figure 3.46.

Training Department System

Home

CPR Activities

Certified Staff

Reports

## New Certified Staff

ID

Click here to add new staff

→

New

+ View

Course

Status

Issue

Select a date

Expiry

Select a date

Note

Add

**Figure 3.47: Certified Staff Tab**

The Certified Staff tab (Figure 3.47) is meant for storing details on which staff members of the hospital are certified in any of the CPR Activities. If the staff member has already been certified previously, their information is most likely still in the system, so their ID number can be chosen without rewriting their name, department and title again.

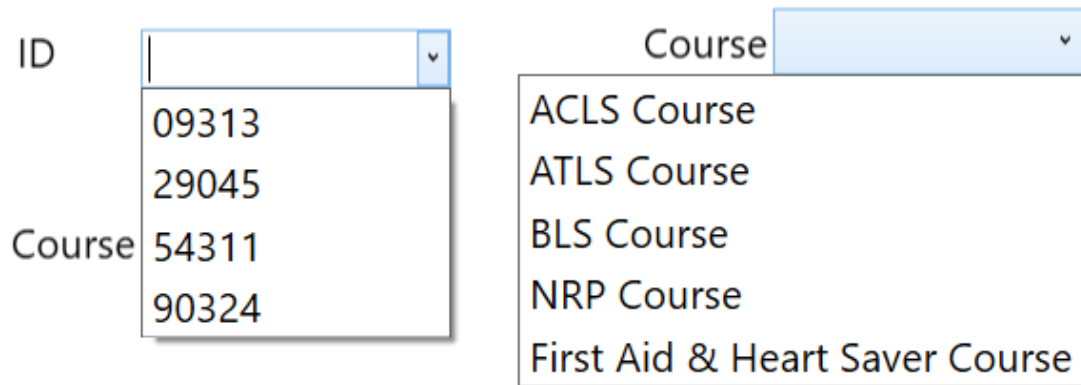


Figure 3.48 displays two dropdown lists. The first list, labeled 'ID', contains the following options: 09313, 29045, 54311, and 90324. The second list, labeled 'Course', contains the following options: ACLS Course, ATLS Course, BLS Course, NRP Course, and First Aid & Heart Saver Course.

**Figure 3.48: ID and Course Dropdown Lists**

All staff members who have already been mentioned in the system will have their IDs in the ID list. Then the user must specify which class they're being certified in with the course dropdown list.

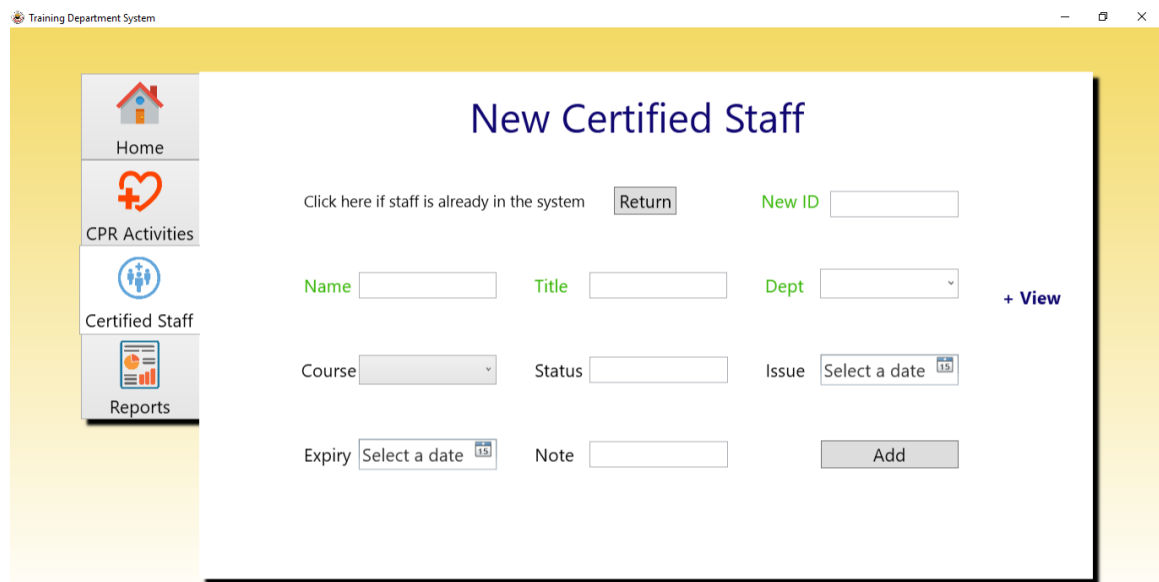


Figure 3.49 shows the 'New Certified Staff' form. The form includes a sidebar with navigation links: Home, CPR Activities, Certified Staff, and Reports. The main form area contains the following fields and controls:

- Click here if staff is already in the system [Return](#) New ID
- Name  Title  Dept  [+ View](#)
- Course  Status  Issue
- Expiry  Note  [Add](#)

**Figure 3.49: New Certified Staff**

If the staff member's ID is not found in the list, the user can add the new staff member by clicking the "New" button seen in Figure 3.47. This will reveal more boxes for the user to enter the name, title and department of the user (same department dropdown mentioned in previously). After adding a new staff, in the future, just the staff's ID number can be used to make a record of their certification.

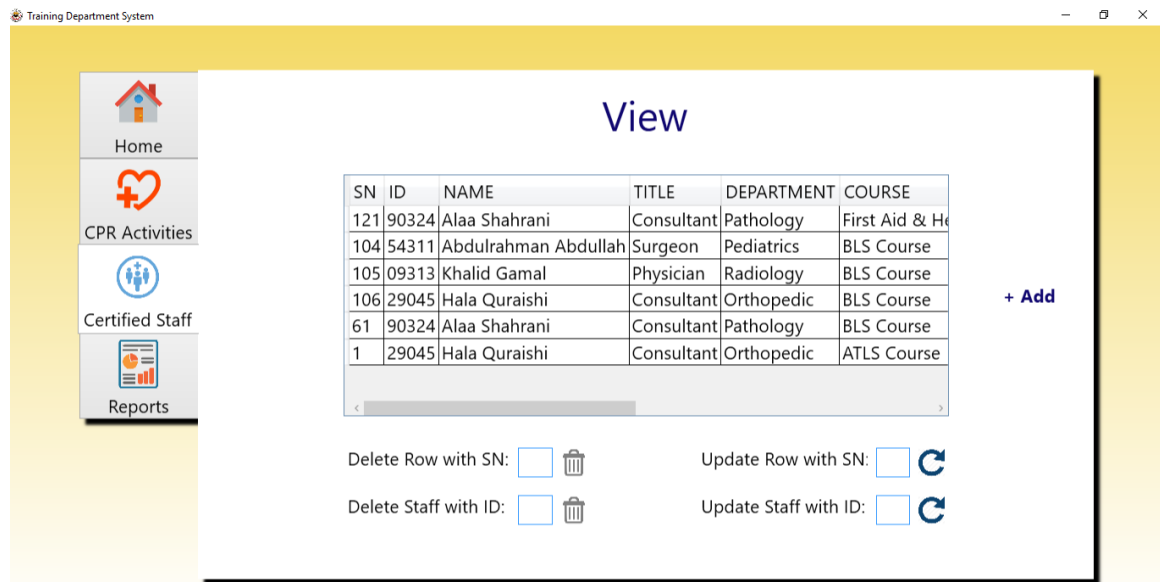


Figure 3.50: Certified Staff View

STATUS	ISSUE DATE	EXPIRY DATE	NOTE
Pending	4/4/2016 12:00:00 AM	4/4/2017 12:00:00 AM	
Complete	4/6/2016 12:00:00 AM	4/24/2018 12:00:00 AM	
Peding	3/30/2016 12:00:00 AM	1/30/2018 12:00:00 AM	

Figure 3.51: Remaining Columns of Certified Staff View

The Certified Staff view gives the option to delete both certification rows (shown in the table) and staff member details (so they won't be available in the list anymore). If the user deletes a staff member, all certification info regarding them will also be deleted.

Update Staff

Name	Title	Department
Hala Quraishi	Consultant	Orthopedic

Ok Cancel

**Figure 3.52: Update Staff**

Choosing to update a staff member based on their ID will produce the dialog box in Figure 3.52. This will change any certification row in the table referencing the staff member.

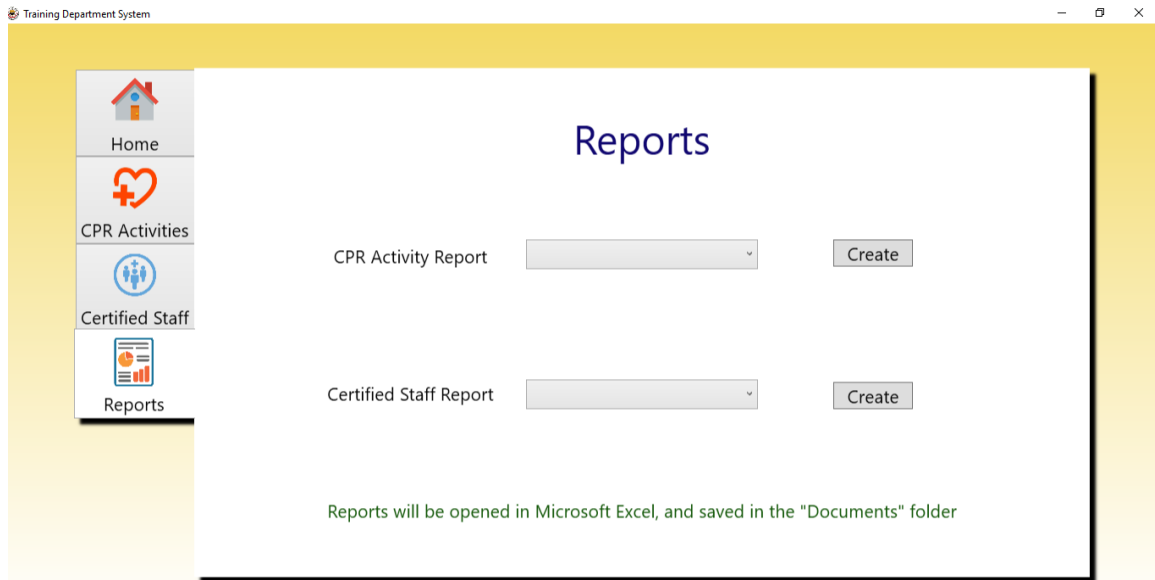
Update Staff

ID	Course	Status	Issue	Expiry	Note
90324	First Aid & Heart	Pending	4/4/2016	4/4/2017	

Ok Cancel

**Figure 3.53: Update Certification Row**

Choosing to update a certification row based on the SN of the row will let users edit the remaining information relating to the certification itself, not the staff member.



**Figure 3.54: CPR Reports Tab**

The CPR unit has a Reports tab as well, shown in Figure 3.54. The note at the bottom of the screen reads: “Reports will be opened in Microsoft Excel, and saved in the ‘Documents’ folder”. Both report types are Excel files. One allows the user to select a specific kind of CPR activity and create a file of the records of all the occurrences of that activity. The other allows users to choose a CPR activity and create a file of all the staff that are certified under that activity.

Clipboard		Font		
A1		fx		
	SN	ACTIVITY	ATTENDEES	DATE
1	1	ATLS Course	8	4/13/2016
2	2	ATLS Course	21	4/24/2016
3	3	ATLS Course	20	4/26/2016
4	Total Attendees		49	
5				
6				

**Figure 3.55: CPR Activities Report Sample**

Figure 3.55 is a sample of what the report would look like if one were to choose ATLS Course from the list, and create the first type of report.

**New Trainee**

Name\*  Dept\*  University\*

Start Date  Duration  Training Type\*  [+ View](#)

Choose from the list. If you want to enter a new training activity name, be aware that it CAN NOT BE DELETED in the future

Email  Phone  Status

Note

**Figure 3.56: Trainees Tab**

Figure 3.56 is the Trainees tab in the Training unit view. It is the same as the Secretary’s Coop Trainees tab, except that the Training unit is responsible for all types of trainees, not just Coop students. The note under “Training Type” has the same message in the secretary view.

**View**

SN	NAME	DEPARTMENT	UNIVERSITY	START DATE
1	Rawan AlMarhomi	Dentistry	King AbdulAziz University	1/1/2016 12:00:00 AM
2	Alaa AlSharani	OB/GYN	Taiba University	8/3/2016 12:00:00 AM
3	Muhammad Khalaf	EMS	King Saud University	4/27/2016 12:00:00 AM
4	Abdullah Sindi	Medical	King AbdulAziz University	11/16/2015 12:00:00 AM
5	Ali Khan Sikander	Radiology	Yanbu Industrial College	10/20/2015 12:00:00 AM

[View All](#) [View Paid](#) [View Not Paid](#) [View by Type](#) [+ Add](#)

Delete Row with SN:  Update Row with SN:

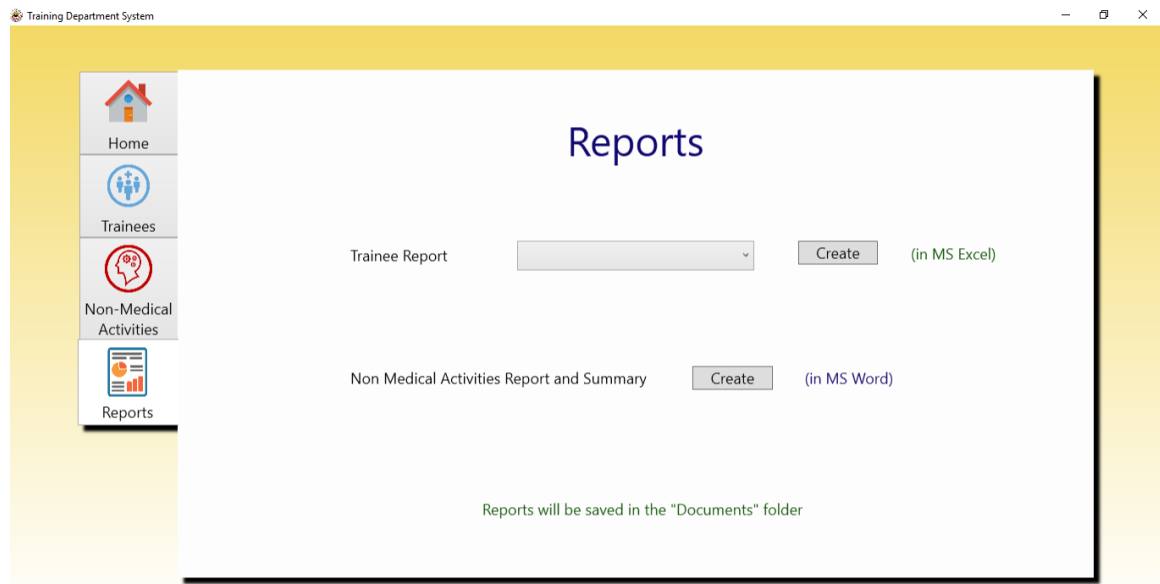
**Figure 3.57: Trainees View**

DURATION	TRAINING TYPE	EMAIL	PHONE	STATUS	NOTE
2 Months	Summer Training	rawan_almarhoumi@hotmail.com	509380514	Terminated	Canceled training
10 Months	Cooperative Training	a.sharani@gmail.com	0557832234	Active	
1 Year	Cooperative Training	muhammad141@hotmail.com	5806779011	Complete	

**Figure 3.58: Remaining Columns in Trainees View**

The Trainees view shown in Figure 3.57 is slightly different than the other tab views throughout the program, as it has more searching and organizing options. This is because there will be potentially hundreds of trainees in the hospital at any given time. It is important for the Training unit to know which trainees receive a salary and which don't, hence the "View Paid" and "View Not Paid" buttons. There is also the "View by Type" button, where the user can choose a training type from the MASTER\_TABLE and see trainees only categorized under that type.

It is worth mentioning that the Trainee tab Non-Medical Courses is exactly the same as the secretary's tab, mentioned in Figures 3.31-3.34.



**Figure 3.59: Training Reports**

The Training unit has its own reports tab shown in Figure 3.59. The Non Medical Activities Report and Summary are the same as those created in the secretary's Annual Report and Annual Report Summary. The Trainee report prompts the user to select a training type, and then produces an Excel report of all the trainees under that type.



Employee Training Schedule													
	A1		SN										
	A	B	C	D	E	F	G	H	I	J	K	L	
1	SN	NAME	DEPARTMENT	UNIVERSITY	START DATE	DURATION	TRAINING TYPE	EMAIL	PHONE	STATUS	NOTE		
2		1 Abdullah Sindi	Medical	King AbdulAziz University	11/16/2015	1 Year	Summer Training	sindim1234@hotmail.c	503841231	Active			
3		2 Rawan AlMarhoumi	Dentistry	King AbdulAziz University	1/1/2016	2 Months	Summer Training	rawan_almarhoumi@h	509380514	Terminate	Canceled training		
4													

**Figure 3.60: Trainee Report Sample**

Figure 3.60 is a sample of the kind of report that would be created if the user selected “Summer Training” from the list.

### 3.2.6 Implementation

The project has over 3,500 lines of functions, not including the design code. The functions that are shown were chosen to highlight the main points of the application.

The following function interacts with the database to retrieve data to fill the datagrids in every “View”.

```

1. public static DataView viewGrid(string query)
2. {
3.     DBConnection db = new DBConnection();
4.     db.Connect();
5.     DataView source = new DataView();
6.     DataSet result = new DataSet();
7.
8.     if (db.GetConnectionState())
9.     {
10.         db.SetSql(query);
11.         result = db.RetrieveRecords();
12.         source = new DataView(result.Tables[0]);
13.     }
14.
15.     db.Dispose();
16.
17.     return source;
18. }

```

This function is reused for every different function passing through it. For example, another function simply has to create a query string and pass it to the function from its own class, as shown:

```

1. public static DataView sympView()
2. {
3.     string query = "select sn, activity AS \"Activity\", date_ AS \"Date\", venu
e AS \"Venue\", scfhs_credit_hours AS \"SCFHS Credit Hours\", attendees AS \"Att
endees\" from symp_conf";
4.
5.     return SecViewModel.viewGrid(query);
6. }

```

The above function fills the Symposium/Conference datagrid. The next function uses the same viewGrid function to fill the CPR Activities view, in another class and for a completely different user:

```
1. public static DataView actView()
2. {
3.     string query = "select sn, activity, attendees, date_ AS \"DATE\" from cpr";
4.
5.     return CPRViewModel.viewGrid(query);
6. }
```

The next function interacts with the database to process any transaction, meaning any insertion, deletion, or update in the data base.

```
1. public static bool executeQuery(string query, Dictionary<string, object> parameters)
2. {
3.     bool success = false;
4.     DBConnection db = new DBConnection();
5.     db.Connect();
6.
7.     if (db.GetConnectionState())
8.     {
9.         db.SetSql(query);
10.        foreach (var param in parameters) db.AddParameter(param.Key, param.Value);
11.
12.        if (db.ExecuteTransactions())
13.            success = true;
14.        else
15.            success = false;
16.    }
17.    else
18.        success = false;
19.
20.    return success;
21. }
```

Parameters are used to prevent SQL injection, a security threat. The next function is an overwritten version of the previous, except that it doesn't accept a parameter list (for transactions like "delete from table\_name", solely used in the application for the "New Year" functionality):

```
1. public static bool executeQuery(string query)
2. {
3.     bool success = false;
4.     DBConnection db = new DBConnection();
5.     db.Connect();
6.
7.     if (db.GetConnectionState())
```

```

8.     {
9.         db.SetSql(query);
10.
11.         if (db.ExecuteTransactions())
12.             success = true;
13.         else
14.             success = false;
15.     }
16.     else
17.         success = false;
18.
19.     return success;
20. }

```

The following are some examples of the executeQuery function being used (deleting and inserting into the symp\_conf table):

```

1. public static bool sympDelete(int sn)
2. {
3.     string query = "delete from symp_conf where sn = :SN";
4.
5.     paramDict = new Dictionary<string, object> { { "SN", sn } };
6.
7.     return SecViewModel.executeQuery(query, paramDict);
8. }
9.
10. public static bool sympAdd(string activity, string date, string venue, int attendees, int hours)
11. {
12.     string query = "insert into symp_conf(activity, date_, venue, scfhs_credit_hours, attendees) values(:act, :dat, :ven, :scf, :att)";
13.
14.     paramDict = new Dictionary<string, object> { { "act", activity }, { "dat", date }, { "ven", venue }, { "scf", hours }, { "att", attendees } };
15.
16.     return SecViewModel.executeQuery(query, paramDict);
17. }

```

The executeCommand function, shown next, is used only in the newYear functionality for executing the non-query commands of “drop sequence” and “create sequence”, necessary for restarting the count of SN values after everything gets deleted.

```

1. public static bool executeCommand(string command)
2. {
3.     OracleConnection con = new OracleConnection("User Id=marya;Password=marya2016;Data Source=(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (COMMUNITY = tcp.world)(PROTOCOL = TCP)(Host = rcymcdb2)(Port = 1521)))(CONNECT_DATA = (SID = orcl)))");
4.     if (con.State != ConnectionState.Open)
5.     {
6.         con.Open();
7.     }
8.
9.     if (con.State == ConnectionState.Open)
10.    {

```

```

11.         OracleCommand cmd = con.CreateCommand();
12.         cmd.CommandText = command;
13.         cmd.ExecuteNonQuery();
14.     }
15.
16.     if (con.State == ConnectionState.Open)
17.     {
18.         con.Close();
19.     }
20.     con.Dispose();
21.
22.     return true;
23. }

```

The following shows the complete newYear function for the secretary user, using both executeQuery and executeCommand methods

```

1. public static bool newYear()
2. {
3.     string[] queries = new string[6];
4.     string[] drops = new string[5];
5.     string[] creates = new string[5];
6.     bool[] success = new bool[16];
7.     bool result = true;
8.
9.     queries[0] = "delete from symp_conf";
10.    queries[1] = "delete from cme";
11.    queries[2] = "delete from non_medical_course";
12.    queries[3] = "delete from medical_course";
13.    queries[4] = "delete from training_activities";
14.    queries[5] = "delete from trainee where training_type = 'Cooperative Training'";
15.
16.    drops[0] = "drop sequence symp_conf_seq";
17.    drops[1] = "drop sequence cme_seq";
18.    drops[2] = "drop sequence non_med_seq";
19.    drops[3] = "drop sequence med_seq";
20.    drops[4] = "drop sequence t_act_seq";
21.
22.    creates[0] = "create sequence symp_conf_seq";
23.    creates[1] = "create sequence cme_seq";
24.    creates[2] = "create sequence non_med_seq";
25.    creates[3] = "create sequence med_seq";
26.    creates[4] = "create sequence t_act_seq";
27.
28.    int count = 0;
29.    foreach (string item in queries)
30.    {
31.        success[count] = SecViewModel.executeQuery(item);
32.        count++;
33.    }
34.    foreach (string item in drops)
35.    {
36.        success[count] = SecViewModel.executeCommand(item);
37.        count++;
38.    }
39.
40.    foreach (string item in creates)

```

```

41.     {
42.         success[count] = SecViewModel.executeCommand(item);
43.         count++;
44.     }
45.     ///
46.     foreach (bool item in success)
47.     {
48.         if (item == false)
49.             result = false;
50.     }
51.
52.     return result;
53. }

```

The next method interacts with the database to retrieve lists of strings to populate any dropdown list (called ComboBox in WPF) in the application.

```

1.  public static List<string> comboFill(string query)
2.  {
3.      DBConnection db = new DBConnection();
4.      db.Connect();
5.      List<string> names = new List<string>();
6.      DataSet result = new DataSet();
7.      DataTable table = new DataTable();
8.
9.      if (db.GetConnectionState())
10.     {
11.         db.SetSql(query);
12.         result = db.RetrieveRecords();
13.         table = result.Tables[0];
14.         for (int i = 0; i < table.Rows.Count; i++)
15.         {
16.             String Name = Convert.ToString(table.Rows[i][0]);
17.             names.Add(Name);
18.         }
19.     }
20.     db.Dispose();
21.
22.     return names;
23. }

```

The following four methods all use the comboFill method to retrieve data from the master table (though comboFill can be used to make lists out of any table data in the database, not just the master table).

```

1.  public static List<string> cpr()
2.  {
3.      string query = "select name from master_table where type='CPR'";
4.
5.      return (MasterTableViewModel.comboFill(query));
6.  }
7.
8.  public static List<string> departments()
9.  {
10.     string query = "select * from master_table where type = 'Department'";

```

```

11.
12.     return (MasterTableViewModel.comboFill(query));
13. }
14.
15. public static List<string> university()
16. {
17.     string query = "select name from master_table where type = 'Private-
Uni' OR type = 'Public-Uni'";
18.
19.     return (MasterTableViewModel.comboFill(query));
20. }
21.
22. public static List<string> training()
23. {
24.     string query = "select name from master_table where type = 'Training'";
25.
26.     return (MasterTableViewModel.comboFill(query));
27. }
28.
29. }

```

The next method is for populating update dialogs in the application. As seen previously in section 3.2.5, when the user updates a row, the dialog box is already filled with the data of the row they chose. That requires the following method:

```

1. public static List<string> updateFill(string query, Dictionary<string, object> p
arameters)
2. {
3.     DBConnection db = new DBConnection();
4.     db.Connect();
5.     DataSet result = new DataSet();
6.     DataTable table = new DataTable();
7.     List<string> items = new List<string>();
8.
9.     if (db.GetConnectionState())
10.    {
11.        db.SetSql(query);
12.        foreach (var param in parameters) db.AddParameter(param.Key, param.Value
);
13.        result = db.RetrieveRecords();
14.        table = result.Tables[0];
15.
16.        for (int i = 0; i < table.Columns.Count; i++)
17.        {
18.            try
19.            {
20.                String Name = Convert.ToString(table.Rows[0][i]);
21.                items.Add(Name);
22.            }
23.            catch
24.            { }
25.        }
26.    }
27.
28.    db.Dispose();
29.

```

```

30.     return items;
31. }

```

Here are two examples of the two update dialogs available in the CPR user's Certified Staff view:

```

1.  public static List<string> staffUpFill(string id)
2.  {
3.      string query = "select name, title, department from staff where id = :ID";
4.
5.      paramDict = new Dictionary<string, object> { { "ID", id } };
6.
7.      return CPRViewModel.updateFill(query, paramDict);
8.  }
9.
10. public static List<string> certUpFill(string sn)
11. {
12.     string query = "select id, course, status, issue_date, expiry_date, note from certified_staff where sn = :SN";
13.
14.     paramDict = new Dictionary<string, object> { { "SN", sn } };
15.
16.     return CPRViewModel.updateFill(query, paramDict);
17. }

```

The following two methods show how the training activities table is updated whenever a new trainee is added.

```

1.  public static bool coopAdd(string name, string dept, string univ, string start,
    string duration, string type, string email, string phone, string status, string
    note)
2.  {
3.      string query = "insert into trainee(name, department, university, start_date
    , duration, training_type, email, phone, status, note) values(:nam, :dep, :uni,
    to_date(:star,'mm/dd/yyyy'), :dur, :typ, :ema, :pho, :sta, :note)";
4.
5.      paramDict = new Dictionary<string, object> { { "nam", name }, { "dep", dept
    }, { "uni", univ }, { "star", start }, { "dur", duration }, { "typ", type}, { "e
    ma", email}, { "pho", phone}, { "sta", status}, { "note", note} };
6.
7.      bool result = SecViewModel.executeQuery(query, paramDict);
8.
9.      if (result)
10.     {
11.         actTotalUp(type);
12.     }
13.
14.     return result;
15. }
16.
17. public static bool actTotalUp(string type)
18. {
19.     string query = "update training_activities set no_of_trainees = (select coun
    t(sn) from trainee where trainee.training_type = :typ) where activity = :typ";
20.
21.     paramDict = new Dictionary<string, object> { { "typ", type } };

```

```

22.
23.     return SecViewModel.executeQuery(query, paramDict);
24. }

```

Also, when trainees are deleted, the training activities should be updated as well. Here is the function:

```

1. public static bool coopDelete(int sn)
2. {
3.     string query = "delete from trainee where sn = :SN AND training_type = 'Coop
    erative Training' ";
4.
5.     paramDict = new Dictionary<string, object> { { "SN", sn } };
6.
7.     bool result = SecViewModel.executeQuery(query, paramDict);
8.
9.     actTotalUp("Cooperative Training");
10.
11.     return result;
12. }

```

Another method designed for repetitive use is the count method, for returning integer amounts of sums and counts from tables.

```

1. public static int count(string query)
2. {
3.     DBConnection db = new DBConnection();
4.     db.Connect();
5.     DataSet result = new DataSet();
6.     DataTable table = new DataTable();
7.     int num = 0;
8.
9.     if (db.GetConnectionState())
10.    {
11.        db.SetSql(query);
12.        result = db.RetrieveRecords();
13.        table = result.Tables[0];
14.
15.        num = int.Parse(table.Rows[0][0].ToString());
16.    }
17.
18.    db.Dispose();
19.
20.    return num;
21. }

```

Count is used for the values in reports. The following two methods give an example of using the count method:

```

1. public static int totalCmeAtt(string type)
2. {
3.     string query = "select scfhs_credit_hours from cme where activity like '%"
    + type + "%' ";

```



```

4.
5.     return SecViewModel.count(query);
6. }
7.
8. public static DataView totalNonMed()
9. {
10.     string query = "select activity, count(activity), sum(no_of_attendees) from
        non_medical_course group by activity";
11.
12.     return SecViewModel.viewGrid(query);
13. }

```

The last function worth mentioning is the Authentication method for approving username and password combinations and returning their permission key word to the program.

```

1. public static string Authenticate(string UserName, string Password)
2. {
3.     DBConnection db = new DBConnection();
4.     db.Connect();
5.     string resolved = "";
6.
7.     if (db.GetConnectionState())
8.     {
9.         //the connection is open, meaning they're connected to RCYEMPLOY
EE
10.         db.SetSql("select permission from users where (username = :userN
ame) AND (password = :passWord)");
11.         db.AddParameter("userName", UserName);
12.         db.AddParameter("passWord", Password);
13.         DataSet result = db.RetrieveRecords();
14.         DataTable table = result.Tables[0];
15.
16.         if (table.Rows.Count == 0)
17.         {
18.             //the password or username is incorrect / doesn't exist in t
he users table
19.             resolved = "empty";
20.         }
21.         else if (table.Rows.Count == 1)
22.         {
23.             //the password and username were found, user can login
24.             resolved = table.Rows[0]["Permission"].ToString();
25.         }
26.         else if (table.Rows.Count > 1)
27.         {
28.             //more than one password and username match the user's input
,
29.             //which should never happen because the password & username
make up
30.             //the primary key, but just in case:
31.             resolved = "error";
32.         }
33.     }
34.     else
35.     {
36.         //the connection could not be established, not connected to RCYE
MPLOYEE
37.         resolved = "connect";

```

```

38.         //}
39.     }
40.     return resolved ;
41. }

```

In addition to the table creation, tables (SYMP\_CONF, CME, MEDICAL\_COURSE, NON\_MEDICAL\_COURSE, TRAINING\_ACTIVITIES, TRAINEE, CPR, and CERTIFIED\_STAFF) all function under the following sequence/trigger creation:

```

1. CREATE SEQUENCE --seq_name--;
2. CREATE OR REPLACE TRIGGER --trigger_name--
3. BEFORE INSERT ON --table_name--
4. FOR EACH ROW
5. BEGIN
6.     SELECT --seq_name-- .NEXTVAL
7.     INTO :new.SN
8.     FROM dual;
9. END;
10. /

```

The purpose of this code is to solve the problem in of no auto-incrementation available in Oracle DBMSs. What it accomplishes is automatically incrementing the SN field in the above mentioned tables, to mimic an auto-increment action for this field. This is so each field will have an individual SN and so this field can be used as a primary key to distinguish rows.

Line by line, the code does the following: CREATE SEQUENCE begins a new sequence, which should have a different name for each table. This is so each table can count up its rows individually. A sequence is simply a counting mechanism in Oracle, starting with 0. CREATE OR REPLACE TRIGGER assigns a new trigger to the task of incrementing the sequence. BEFORE INSERT ON is important, because the SN field is the primary key for these tables, a row cannot be inserted without first filling the SN field. SEQUENCE\_NAME.NEXTVAL finds the next value from the previous entered into the system. In a perfect world, this would suffice as auto-incrementation, but when rows are deleted in between, it causes the SN values to count out-of-order. This has no detrimental effect on the data, but does cause unpleasant presentation. INTO :NEW.SN inserts the next value of the sequence into the new row's SN field, and FROM DUAL indicates that the sequence values are maintained in a sort of master table Oracle maintains called "Dual".

### 3.2.7 Testing

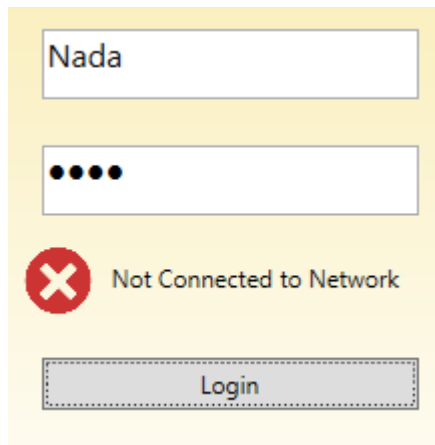
The following are the test cases gone through to test the system:

#### **Test Case 1** – Logging in while not connected to the correct network

Input: Correct username/password

Expected Result: Rejection from entering the system

Result:



**Figure 3.61: Test Case 1**

Description: The computer is not connected to the hospital's RCYEMPLOYEE network. Because the entire application relies on interaction to the hospital's servers, it will not allow a user to enter without being connected.

If the network connection is lost while the user is past the login screen, the application will run incredibly slowly, and no values will appear in any datagrid or dropdown list, but the system will not crash.

#### **Test Case 2** – Logging in with the incorrect username/password combination

Input: Incorrect username/password combination

Expected Result: Rejection from entering the system

Result:

A login form with a yellow background. It contains a text input field with the username "Nada", a password input field with seven black dots, a red circle with a white 'X' icon, and the text "Username/Password Incorrect". Below these is a grey "Login" button.

Figure 3.62: Test Case 2

Description: The username and password combination do not match what is held in the database. User is prevented from entering the remainder of the application in this case.

**Test Case 3** – Creating a new user, choosing the wrong password

Input: Valid username, valid “New Password”, valid but mismatched “Re-Enter Password”

Expected Result: User not created

Result:

A form for creating a new user. It has three input fields: "New Username:" with the value "Marya", "New Password:" with seven black dots, and "Re-Enter Password:" with seven black dots. Below the fields is a grey "Add" button. To the right of the form is a white error dialog box with a red border, a close button (X), the text "Passwords don't match", and an "OK" button.

Figure 3.63: Test Case 3

Description: User tries to create a new user in the Home tab, but the password confirmation fails (new password and old password field don't match). New user will not be created and values will be cleared.

**Test Case 4** – Creating a new user, username/password combination already exists

Input: Username and password that already exist as a combination

Expected Result: User not created

Result:

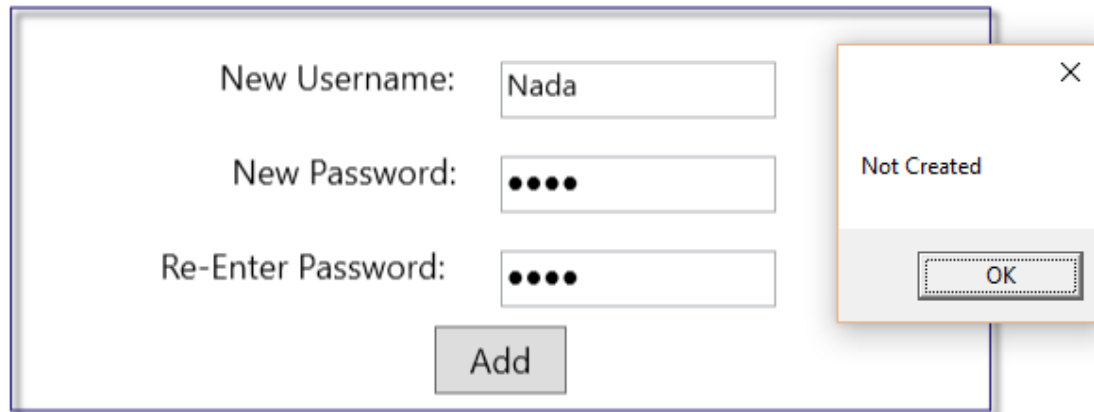


Figure 3.64: Test Case 4

Description: User attempts to create a new user, but username/password combination already exists. The new user will not be created

**Test Case 5** – Change password, enter wrong password

Input: Incorrect old password

Expected Result: Password doesn't change

Result:

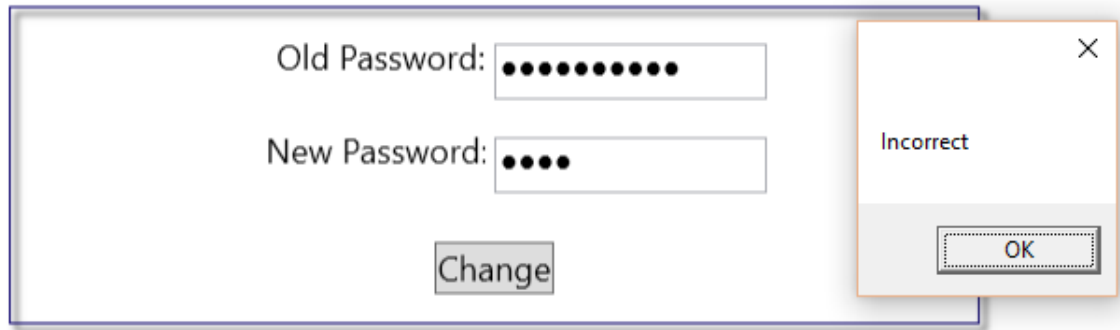


Figure 3.65: Test Case 5

Description: User attempts to change password, but enters the incorrect current (old) password. User is not allowed to change passwords if they can't remember the old one.

**Test Case 6** – “New Year” dialog, user doesn’t fill in confirmation fields

Input: None (fields are left empty)

Expected Result: Dialog doesn’t continue

Result:

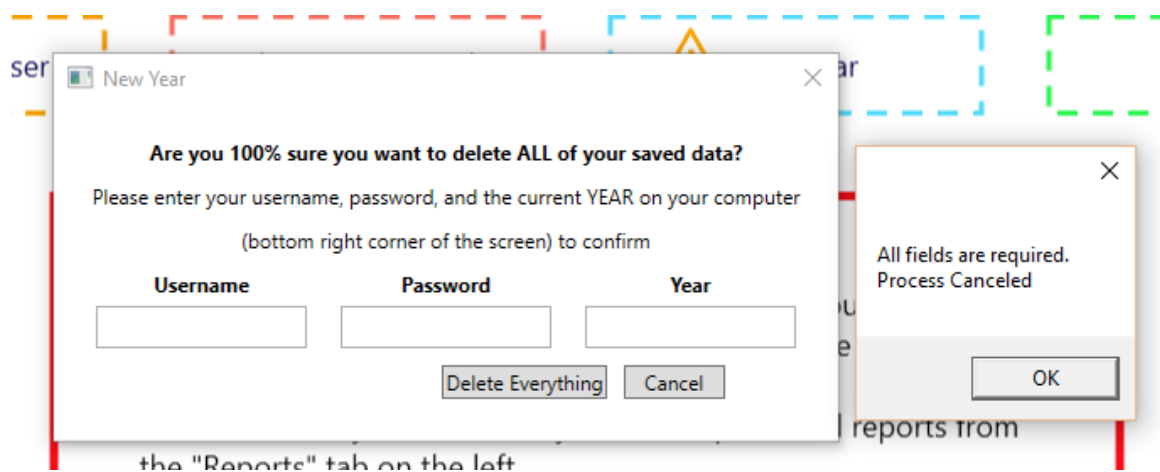


Figure 3.66: Test Case 6

Description: User attempts to delete all data (“New Year” button) and receives the confirmation dialog. The user presses the “Delete Everything” button without filling any fields to confirm. The application does not carry out the deletion process

**Test Case 7** – “New Year” dialog, user enters a confirmation field incorrectly

Input: Incorrect username, correct password, correct year

Expected Result: Dialog doesn't continue

Result:

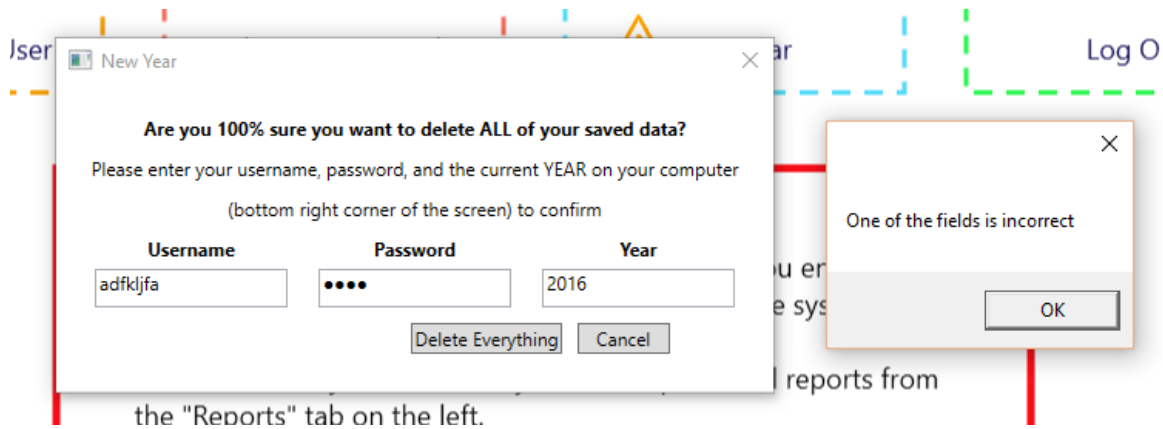


Figure 3.67: Test Case 7

Description: User attempts to delete everything, but one of the fields is incorrect. The application does not carry out the deletion process.

**Test Case 8** – “New Year” dialog, table(s) is empty

Input: Pressing “Delete Everything”, but one or more table in the user’s view is empty

Expected Result: Everything will be deleted, up until the point where the empty table was found. Tables after the empty table won’t be deleted

Result:

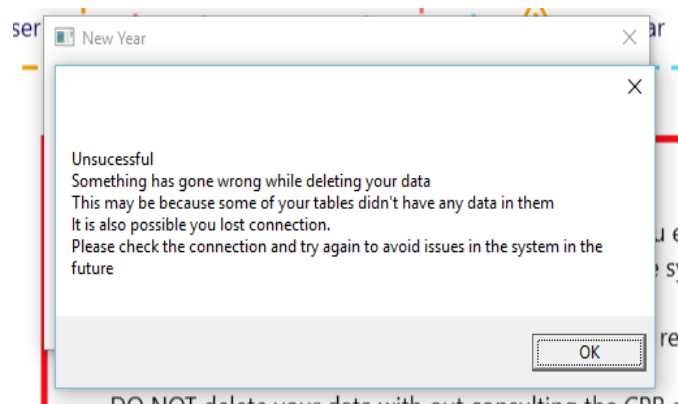


Figure 3.68: Test Case 8

Description: The user tries to delete everything, but some tables are empty. The deletion will stop after the empty table. Also could occur is network is lost in the middle of the deletion process. The user should enter junk data in the empty tables to be able to delete the remaining data.

**Test Case 9** – Adding to any table, without filling in required fields

Input: Anything, but leaving one or all of the required fields blank

Expected Result: Nothing will be added to the table, user will be alerted

Result:

The screenshot shows a web form titled "New COOP Trainee" in blue text. The form contains several input fields: "Name\*" (text), "Dept\*" (dropdown), "University\*" (dropdown), "Start Date" (dropdown with "Select" visible), "Training type\*" (dropdown), "Email" (text), "Status" (dropdown), and "Note" (text). At the bottom right is an "Add" button. A red-bordered error dialog box is centered over the form, displaying the message: "Name, Department, University, and Training Type are all required fields". The dialog box has a close button (X) in the top right corner and an "OK" button at the bottom right.

**Figure 3.69: Test Case 9**

Description: The user attempts to add a row to a table, but doesn't fill in the required fields. Each table has its own required fields. Each table will not allow a row to be inserted without filling those fields first.

**Test Case 10** – Typing a value into a dropdown list that doesn't accept new values

Input: Value that doesn't match any value already populating the list

Expected Result: Field will be automatically cleared

Result:



Name*	<input type="text" value="Marya Belanger"/>	Dept*	<input type="text" value="New Dept"/>
Name*	<input type="text" value="Marya Belanger"/>	Dept*	<input type="text"/>

**Figure 3.70: Test Case 10**

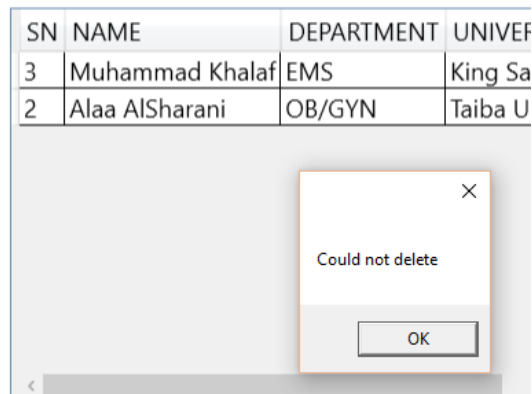
Description: User attempts to enter a new name into a dropdown list that doesn't accept new values, like the department dropdown list. As soon as the user changed focus out of the box, the system will check to see if the value matches one of the items in the box. If not, it will immediately clear the box before anything can be submitted. This prevents values that don't exist as foreign keys for a table being entered into the system.

**Test Case 11 – Deleting a row, SN doesn't exist**

Input: SN number not present in the table

Expected Result: Row won't be deleted

Result:



Delete Row with SN:

**Figure 3.71: Test Case 11**

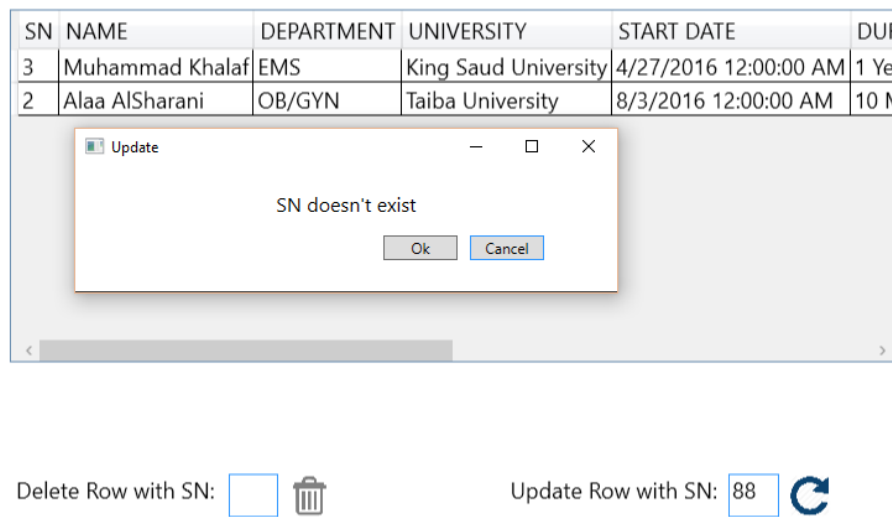
Description: User attempts to delete a row by its SN number, but that SN does not exist in the table. The deletion will not take place.

**Test Case 12** – Updating a row, SN doesn't exist

Input: SN number not present in the table

Expected Result: Row won't be deleted

Result:



**Figure 3.72: Test Case 12**

Description: User attempts to update a row by its SN number, but the row does not exist. The user is not given the option to update anything.

**Test Case 13** – User doesn't fill in required field in update dialog

Input: Any input, with one or more required fields left blank

Expected Result: "Ok" button of dialog will be disabled

Result:

The image shows a software dialog box titled "Update". It contains several input fields and dropdown menus arranged in a grid. The "Name" field at the top left is empty and has a red circle around it. To its right are "Dept" (set to "EMS") and "University" (set to "King Saud Univer"). Below "Name" is "Start Date" (set to "4/27/2016") and "Duration" (set to "1 Year"). To the right of "Start Date" is "Training Type" (set to "Cooperative Train"). Below "Start Date" is "Email" (set to "muhammad141@hotmail.com") and "Phone" (set to "5806779011"). To the right of "Email" is "Status" (set to "Complete"). At the bottom is a "Note" field. The "Ok" button at the bottom right is also circled in red, while the "Cancel" button is not.

**Figure 3.73: Test Case 13**

Description: User updates a legitimate row, but leaves fields necessary for that table blank in the update dialog. When a necessary field is left blank in an update dialog, the “Ok” button is disabled until the user enters a value. The dropdown lists in update dialogs behave the same way throughout the whole application; if the user enters a illegitimate value, it will automatically be cleared.

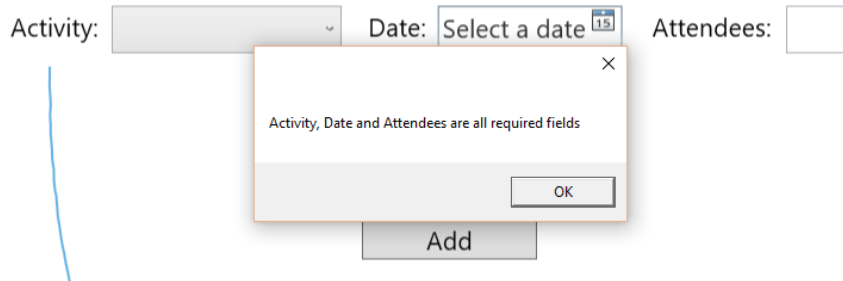
**Test Case 14 – Add CPR activity, field left blank**

Input: Any or all fields left blank

Expected Result: Row not added

Result:

## Add New Activity



The screenshot shows a web form titled 'Add New Activity'. It has three input fields: 'Activity:' (a dropdown menu), 'Date:' (a date picker showing '15'), and 'Attendees:' (a text box). Below these fields is an 'Add' button. A modal dialog box is open in the center, displaying the message 'Activity, Date and Attendees are all required fields' and an 'OK' button. A blue line is drawn on the left side of the form, starting from the 'Activity:' label and extending downwards.

**Figure 3.74: Test Case 14**

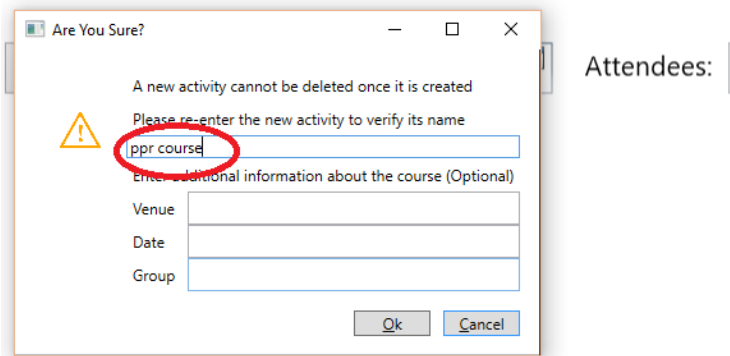
**Description:** Every field in the CPR Activities tab is required, as it is all detrimental to the other reports. Leaving any field blank will block the addition of a new row to this table.

**Test Case 15** – Adding a new CPR activity name, name not confirmed

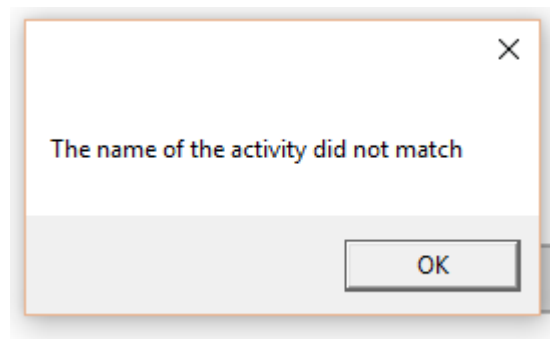
**Input:** First, a valid CPR activity name, but the name in the confirmation dialog doesn't match the original

**Expected Result:** Name will not be added to the master list

**Result:**



**Figure 3.75: Test Case 15 - 1**



**Figure 3.76: Test Case 15 - 2**

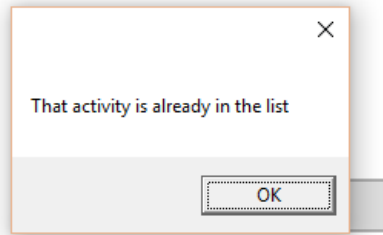
**Description:** If the CPR user wants to add another course, he/she must confirm the name of that course to be sure they haven't made a mistake. If they even change the capitalization of the course name or add a trailing space, the system will not accept the new name.

**Test Case 16** – Adding a new CPR activity name, name already exists

**Input:** The name of a CPR activity that already exists

**Expected Result:** Name will not be added to the master list

**Result:**



• Add a new activity to the list BLS Course

**CAUTION:** Activities cannot be deleted once added to the master list

**Figure 3.77: Test Case 16**

Description: The user attempts to add a “new” CPR course to the list, but it already exists. The system will automatically reject the addition and won’t even ask the user to confirm.

**Test Case 17** – Adding a new staff member, ID is already in the system

Input: ID that’s already in the system

Expected Result: New staff will not be added

Result:

## New Certified Staff

Click here if staff is already in the system  New ID

Name  Dept

Course  Issue

Expiry  Note

That ID is already in the system

OK

**Figure 3.78: Test Case 17**

Description: The CPR unit user adds a “new” staff member, but the staff member’s ID is already in the list. The system will not accept it. To rectify this, the user can go to the Certified Staff view and delete the staff member by their ID, to create a new one.

**Test Case 18** – User creates a report, there is null data in the database in a number field

Input: Pressing any Word “Create Report” button

Expected Result: Message: “Input string is not in the correct format”

Description: If the user attempts to create a report (any of the reports involving integer values being summed or even just listed) and one of the integer values in the table has been left blank, the report creator will deliver this message in a message box after the button is pressed. However, the report will still be created, the system will not hang, but the report will be blank after the point the empty value was encountered. Users should take care to fill all integer fields, even if just with a 0.

### **3.2.8 Documentation**

Documentation was created in the form of user manuals for each of the three user types (secretary, CPR unit, training unit). A sample of the documentation, the secretary’s user manual, can be found in Appendix C. Due to the length of the user manuals combined, only one has been included as a sample.





## GLOSSARY

<b>Term</b>	<b>Definition</b>
<b>YUC</b>	Yanbu University College, Yanbu, Saudi Arabia
<b>RCMC</b>	Royal Commission Medical Center, Yanbu, Saudi Arabia
<b>COOP</b>	Cooperative Training Program, referring to YUC's final semester option for students
<b>IT</b>	Information Technology (department)
<b>TDS</b>	Training & Development Reporting System (TDS is an abbreviation of the system title).
<b>JCI</b>	Joint Commission International, health care accreditation organization
<b>RC</b>	Royal Commission, the government entity that oversees the city of Yanbu
<b>ICU</b>	Intensive Care Unit, hospital department
<b>Visual Studio</b>	IDE by Microsoft
<b>ASP.NET</b>	Web development framework by Microsoft
<b>C#</b>	Pronounced "C Sharp" language developed by Microsoft in the .NET family
<b>Oracle</b>	Technology company, focus on database technology
<b>NuGet</b>	Package manager designed for Microsoft platforms
<b>Safeer</b>	An ERP system designed by Oracle for the RCMC, focuses on administrative and employee affairs in the hospital
<b>IDE</b>	Integrated Development Environment
<b>.NET</b>	Pronounced "Dot net", software framework by Microsoft
<b>SQL</b>	Structured Query Language, data language for communication with database
<b>XAML</b>	Pronounced "Zamel", Extensible Application Markup Language
<b>WPF</b>	Windows Presentation Foundation, application type developed by Microsoft
<b>SN</b>	Serial Number
<b>OS</b>	Operating System
<b>API</b>	Application Program Interface
<b>DBMS</b>	Database Management System

## **APPENDIX A**

### **Interview with Users (Summarized)**

**COOP Week 3 – Feb. 7<sup>th</sup> – Feb 11<sup>th</sup>**

**Q: What is the current workflow?**

A: Any information about activities or trainees we store in Excel files. When I (secretary) need information from the training department or the CPR instructor, I go to them or they send me their own Excel sheets. I need new information from them almost every day. For activities coordinated outside our department, I have to go to those departments and ask how many attendees they had.

**Q: Who will use the new system?**

A: Secretary: responsible for the report, the COOP trainees and all activities (except for non-medical). Training Unit: responsible for all other trainees and non-medical activities. CPR Unit: responsible for several different types of CPR courses that take place daily.

**Q: What part of the current process do you want to improve?**

A: Definitely the communication issues. I want to be able to receive all the updated information I need from the training unit and CPR unit automatically. The manual way this process is done takes too much time and causes too many problems. This includes doing the report manually. I am left adding numbers from many Excel sheets, some of which are out of date; it's a bad way to do it.

**Q: Who do you receive the info from? Would it be helpful for them to enter the info into the system themselves?**

A: Yes, for the training unit and the CPR unit, because the largest amount of information, and the most frequently arriving information, comes from them. For the other information, from the conferences and continuous medical education for example, it comes from outside our department and there is another (manual) system in place for that process.

**Q: Would it be useful for the new system to check for dates of activities, when venues are being used, etc, like a booking system?**

A: The library already has this system and we get that information from them.

**Q: Is keeping track of attendees by name, rather than just amount, something that may be helpful to you?**

A: For the monthly and annual reports, we just need to know the amount of attendees. The people putting on the activity collect that information for themselves and their own departments, we don't need it.

**Q: What do you want to be able to do with the trainee and student data? (Besides just store it?)**

A: Sorting it is important. Now we just keep separate Excel sheets for all the different types of students, plus the main sheet of all students, which we need for different information. Looking up trainees easily is important, by when they started, what school they're from, things like this. And being able to print a report that doesn't spread out across two pages (in landscape format) because of all the fields we keep for trainees.

**Q: What exactly do you need from the CPR unit?**

A: Because the courses happen so frequently and there are so many types of CPR training like ALS or BLS, we just want to easily get the information from them about what the type of course is, the date, and number of attendees. Also, the CPR unit requested that the system be able to track staff certifications in CPR courses.

**Q: The main goal of the system is...**

A: Sharing information between the parts of the training department, knowing that the information is as free from error as possible, and producing reports in the format of the annual report requested by management.