Inventory management Application

Documentation of the development process for this application. Development by: Mark Silvius and Luke Caraccioli

Cardiac Cath Lab Inventory Management resource

**Table Of Contents**

**Phase I**

* 1. Description of the Product………………………………………2
  2. Product Goals…………………………………………………….4
  3. Programming Tools……………………………………………...5
  4. Significance………………………………………………………7
  5. Framework……………………………………………………….9

**Phase II**

* 1. Research…………………………………………………………13
  2. Planning……….…………………………………………………15
  3. Milestones……………………………………………………….18

**Phase III**

3.1 Back End………………………………………………………...22

3.2 Front End………………………………………………………..

3.3 Reports………………………………………………………….

**Phase I**

**1.1 Product Description**

This product application will also be our Senior Seminar project. We set our sights on designing a real world application that could not only serve the purpose of satisfying a school requirement but also be a milestone on our resume and possibly be the foundation for a development business. The application will offer many useful ways to not only keep up with inventory no matter how small or large the price but to also help manage these resources with extensive and useful reporting tools and statistics management.

The application will be a web application that will have the assumption that when in operation it will be on a server that is located under the umbrella of a large corporate intranet that has a firewall and that contains many other applications for the operation of the company.

The application will hold all inventories for the Cardiac Cath Lab in this case. The application will keep up with the manufacturer, Name of Product, Part Number, Expiration Date, Serial Number, Lot Number, and Location of all Products that enter the department from the receiving department of the hospital. The user will be able to search the system of existing parts for certain parts and be able to tell the location of that part with just a click of a button. The application will also have an expiration page that will tell the user what expires today, and any other date that they want to search for to be sure that the products on the shelf are used and to also set priority of what products should be used first to ensure they are used before they expire.

Another feature that will be present is keeping up with when a product is used, how much was spent per date on products, how much of what vender and what product is being used per date, and a market share of what percentage of the product inventory belongs to which vendor. These features will enable the user to both have the knowledge of exactly what is being used and also to negotiate with vendors about price based on market share in order to lower overall cost of operation.

Each product used will be unique in our system, and to keep up with each part individually we have come up with the idea of creating our own bar code system. We will generate a barcode for each product entered into the database and once it is scanned in the system a barcode report will be generated to print out and attach to the product so it can be scanned once it is used and the system will change the status of that particular product and give it a date used. The date used will be the key to generating cost analysis. The barcode enables us to keep up with each product as most companies carry many of the same product on the shelf.

**1.2 Product Goals**

* Store Inventory
* Track Inventory
* Enter New products easily
* Generate a barcode for each new entry
* Generate useful reports
* Export Reports to pdf

**1.3 Programming Tools**

This application development will use Microsoft SQL server 2012 for the server which will be a stand-alone machine for our server. We will use Microsoft Visual Studio for the programming environment with .NET entity framework, C# coding, AJAX, CSS, HTML5, and Telerik tools for the front end programming and reporting tools.

**1.3.1 Microsoft SQL Server**

Our server runs Microsoft Windows Server 2008 R2. This machine has no other programs on it. This machine is a dedicated server machine with a static IP address so that it can be accessed from anywhere.

We chose Microsoft SQL server because it is the most stable current working database server system for windows application development. We will create the entities, define the primary keys and foreign keys, and define the entity relationships in this environment.

**1.3.2 .NET entity framework**

.NET framework is a software framework developed by Microsoft and includes a large library and language interoperability across several different programming languages. Code that is written in .NET executes in an environment known as CLR, which is a virtual machine that provides services such as security, memory management, and exception handling.

We chose this framework because it is the latest, most advanced language for developing applications and also it is very user friendly. We are able to write queries for the server there and also code in C#, HTML, JavaScript when needed, and use Telerik Tools all in the Microsoft Visual Studio environment.

**1.3.3 C#**

We chose to use C# for our programming language because we learned that in databases, and also this language is so versatile especially when coupled with the .NET framework. Another point to be made is that this is still the most technilogically

**1.3.4 AJAX, HTML5, CSS, & Telerik Tools**

AJAX is an acronym for asynchronous JavaScript and XML. This is a group of interrelated web development techniques used to develop the client side to create applications for the web. AJAX allows for the web application to send data to and receive data from the server without disturbing the front end display. AJAX can be used in conjunction with HTML, CSS, and JavaScript making is very versatile.

We chose to use AJAX because of the above qualities and also for the fact that it is the industry standard for developing web applications. We then decided that the direction that we wanted to go with the front end also going to require for us to use a cutting edge tool for front end and reporting so we chose Telerik Tools.

Telerik makes the most cutting edge controls for .NET framework and are at the forefront of tools developed for AJAX. These tools will prove very useful when we implement grids to hold our data, and customizing the look of our front end and also creating informative and useful reports that easily export to many different formats with the click of a button.

**1.4 Significance**

This application will take the place of the current methods which are used in the Cath Lab. The current method is not keeping track of anything but cardiac stents and pacemakers. The way these are kept track of is through hard copy reports printed out and handed in to a statistic keeper and then are put into an excel spread sheet. This data relies heavily on humans to document the usage correctly, and because of the way the data is stored there is a lot of room for error, also there is no reporting or price comparisons, or real accountability for what is being spent, or what the vendor is charging the Cath Lab for the products and the Cath Lab is at the mercy of the vendor on pricing in a lot of cases. This application will provide so many tools and accountability that has never been available in the past the entire way that products are kept up with will be much improved.

**1.5 Framework**

**Product Entity**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | ID | Part Number | Serial Number | Expiration Date | Location ID | Date Used | Lot Number | Status ID | | Description | unique key | foreign key | part serial num | date product expires | product location | date product was used | Product Lot # | Status designator | | Domain/Type | int | var/char | var/char | date/time | int | date/time | int | int | | Value Range | 0-9 | any | any | any | 0-9 | any | 0-9 | 0-9 | | Default Value | auto assigned | none | none | none | none | none | none | none | | Nullable | no | no | yes | yes | yes | yes | yes | yes | | Unique | yes | no | no | no | no | no | no | no | | Single or multiple value | single | single | single | single | single | single | single | single | | Simple or Composite | simple | simple | simple | simple | simple | simple | simple | simple | |

**Part Number Entity:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  | PartNum | NameSize | ManufaturerID | PoductTypeID | Cost | MediClick | Par | | Description | foreign key | name of part | foreign key | foreign key | product cost | SJCH reorder # | number or required parts | | Domain/Type | var/char | var/char | int | int | Int | var/char | int | | Value Range | 0-9 | Any | 0-9 | 0-9 | 0-9 | any | 0-9 | | Default Value | auto assigend | Any | auto assigend | auto assigend | Any | any | auto assigend | | Nullable | no | no | no | no | Yes | yes | no | | Unique | yes | No | no | no | no | no | no | | Single or multiple value | single | single | single | single | Single | single | single | | Simple or Composite | simple | simple | simple | simple | Simple | simple | simple | |

**Location Entity:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | ID | LocationName | | Description | foreign key | part location | | Domain/Type | int | var/char | | Value Range | 0-9 | any | | Default Value | auto assigend | any | | Nullable | no | no | | Unique | no | no | | Single or multiple value | single | single | | Simple or Composite | simple | simple | |

**Manufacturer Entity:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | ID | Name | PhoneNumber | Email | Address | | Description | foreign key | Name of Company | Phone # | email | mailing address | | Domain/Type | Int | var/char | int | var/char | var/char | | Value Range | 0-9 | any | 0-9 | any | any | | Default Value | auto assigend | auto assigned | auto assigend | auto assigend | auto assigned | | Nullable | No | no | no | no | no | | Unique | No | no | no | no | no | | Single or multiple value | single | single | single | single | single | | Simple or Composite | simple | simple | simple | simple | simple | |

**Product Type Entity:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | ID | Type | | Description | foreign key | Product Type | | Domain/Type | int | var/char | | Value Range | 0-9 | any | | Default Value | auto assigend | auto assigned | | Nullable | no | no | | Unique | no | no | | Single or multiple value | single | single | | Simple or Composite | simple | simple | |

**Table Structure:**

****

**Phase II**

**2.1 Research**

We knew beforehand about some of the methods used at the San Joaquin hospital Cath Lab for data collection and product storage. We then sat down with the two employees that would use this database and interviewed them about the processes, methods, and needs that they have for data collection. We learned that most of the data is dependent upon hard copies of the nurses complete at the procedure room. Once the procedure has been completed the nurse will turn in a hard copy of the billing sheet showing what procedures and what inventory has been used.

The key to our research was the discovery that only some parts are actually kept up with. Most of the inventory that is used never gets documented, never is calculated in the overhead, and is never documented that it was used. There is a lot of inventory in this facility that is brought in and put in on the shelf and never accounted for again.

We then interviewed the director of the Cath Lab and realized the key to our design had to be a way to keep up with ALL products and ALL costs for them and keep up with everything that is being used and when. He stated in the interview that there are a lot of products that attempt to manage inventory on the market but are unable to keep accurate inventory over the smaller items that are used in the day to day operation (ie: wires, sheaths, catheters, etc) In order for this to happen we have to make every product a unique instance. How do we accomplish this? We knew that we could make a key that auto incremented every time there was a new part entered, but we had to come up with a way to then recall that entry quickly to add a used date to the product.

What we ended up with was plans to have a status flag for products and products that have a status of scanned are queried and then added to a report that generates bar codes. These bar codes are exportable to pdf and can be printed onto sticker sheets and then placed on the product before moving to its destination in the Cath Lab.

When we sat down with the director of the Cath Lab, he told us that there was a lot of software on the market that did keep track of inventory but the challenge has always been that they do not do a good job and keeping up with the small more inexpensive items.

We knew our mission objective was to make this product easy to use and keep up with all the cost accrued per date.

**2.2 Planning**

**2.2.1 Server**

First, we will implement our server. We will do this by setting up a machine that would run Windows SQL Server 2012 R2. The machine will be a stand-alone machine that is a dedicated server with a static IP. We are going to use an old computer that Mark has and install the Microsoft SQL server client on this machine. We will set up a static IP so that we can get remote desktop access when needed. Also, we will be able to access the data on the server machine from our respective computers by using the Microsoft SQL server management studio.

**2.2.2 Database**

Next, we started to design the database on paper. This involved deciding what the entities we needed, what the attributes would be, and what the relationships would be to connect the entities would be. We decided that the Product entity would be the center of the project and to keep from having to make many clicks to make choices all but one attribute would be auto assigned to the product when it was entered. This attribute would be the Product Location. The other tables will trigger based off of the information given when the part number is recognized. These tables include the Manufacturer, Product Type, and Partnumber tables

**2.2.3 Front End**

On paper we drew out what the reports should look like and what information was important in a report to the user. Based on what the demands of the user were, we decided that we would duplicate the reports that the company already has in place but in a much nicer and easy to use format. Currently they use excel and access to generate reports for stent usage and utilization. We designed our report to take it a step further and show not only utilization reporting but to also show the exact dollar amount that is being spent per manufacturer and type of stent that is used. What this will do is provide the user with a tool to show a cost break down by manufacturer that will enable them to negotiate price with the vendors. We also added functionality to generate a market share report to show what percentage of the stock held in the department belongs to what vendor to further drive home the value of the vendor constantly working to give the company a better cost per capita.

Our page layouts were straight forward and to the point. We want our users to be able to see the data easily and not have to click more than one or two times to find the data that they are after. The pages we decided on will be a home page to welcome the user, an inventory page that allows the user to browse the current inventory with just a few clicks of a hieratical filter, an expiring page that shows products on the shelf that expire for the current date and gives the user options to explore what products are going to expire in the near future. Also we want to implement a reports page to do all of the reporting that we plan for.

**2.3 Milestones**

**2.3.1 Server**

For the server we used an older computer with an AMD X2 dual core processor, 4 GB of RAM, and a 320 GB hard drive. This should meet the requirements of the template database we are designing and if the product is sold the server machine will have to be more robust.

We acquired Microsoft SQL server 2008 R2 and SQL server 2012 via the Microsoft Academic Alliance for free. We then installed the software on the server and set the LAN IP address to static. Next, we set up the router to port forward Remote Desktop (RDP) and SQL server requests to the server. We configured the RDP and created user accounts for the server. Last, we created system administrator and programmer accounts on SQL server for administering the database, and querying the database respectively.

**2.3.2 Database**

Our database in SQL server is named ‘cathlab’. We created our entities and the attributes, and then we assigned parameters for each of the attributes as was outlined in our entity tables above. Once we had all of that dialed in we then created the relationships that each of the entities would have, normalized our tables, and made sure we had simple attributes in all cases.

Next, we went to the cath lab at SJCH and received a list of parts that they keep in their inventory. We then took this data and put it in the attributes of the database. We used all of the actual inventory the cath lab has and uses. We randomized expiration dates, lot numbers, used dates, and locations for testing purposes.

We set up many stored procedures to pull the data that we needed from the server. We used some of the stored procedures to populate our grids, and other stored procedures were used to populate reports.

**2.3.3 GitHub**

Because of the size of this project we found it necessary to find a way to be able to work on the same project from different machines and be able to save changes made so that the other person could see them as well. We chose to put the project on GitHub. Using this tool to develop really made a big difference especially when the quarter started and we were unable to meet as much in person as we did before the quarter started.

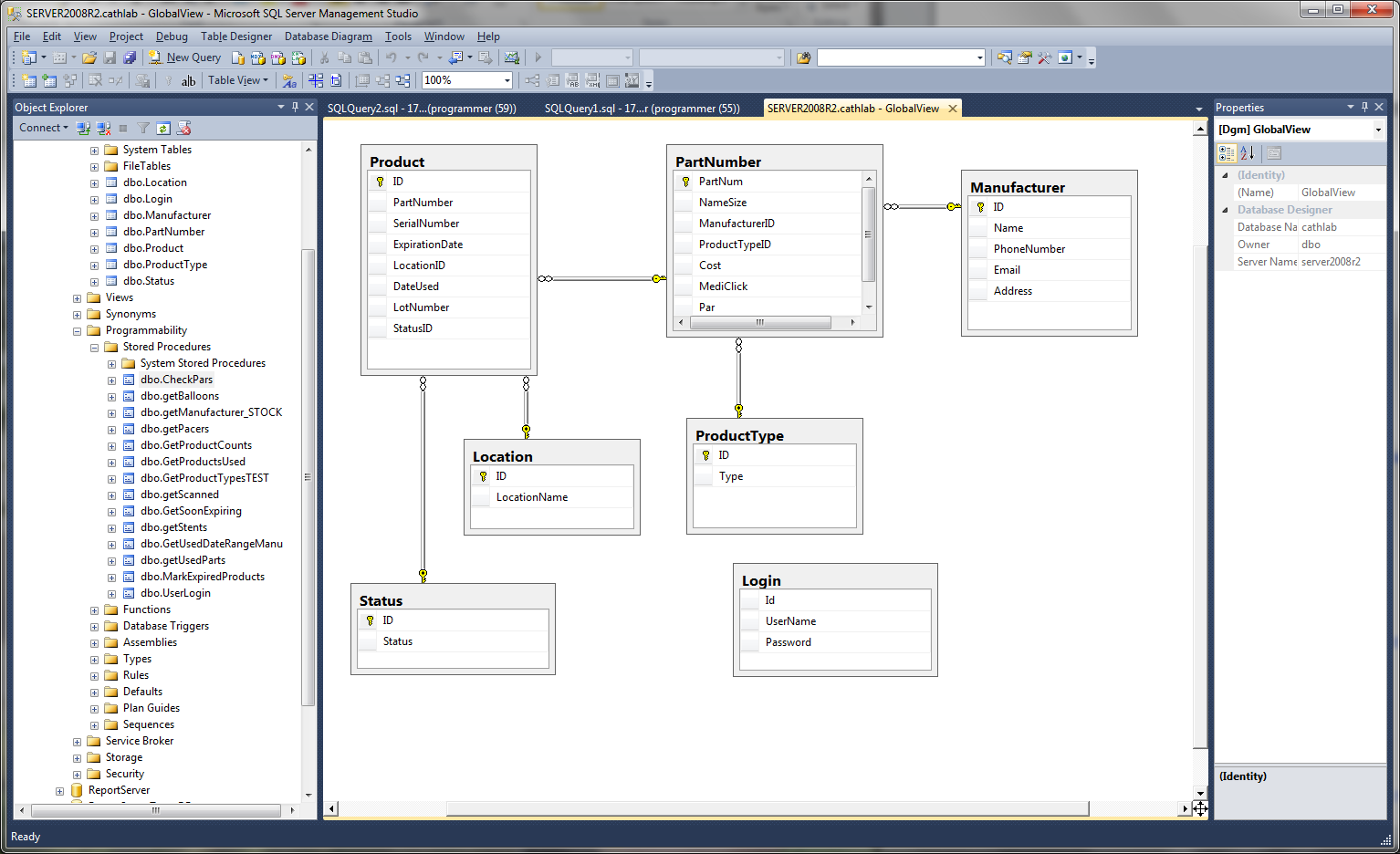
**2.3.4 Front End**

We used the Microsoft Visual Studio environment to create all the graphical, and functional parameters for the front end. The application had to be a web app per Dr. Wang’s request, so we created a new web application in visual studio. The code for this portion is a combination of CSS, AJAX, Telerik, and HTML5. We designed the home page that has a slide show, and the available options for the user on the site. The Inventory page allows the user to browse current inventory via selection boxes at the top of the div, and also by clicking on the grid itself the part will show how many are in stock as well as the location of that particular part. There is an expiring page that shows a grid with all expiring parts and the side menu on the page lets the user filter that data into ‘today’, ’10 days’, and ’30 days’ from today’s date. There is a types page that enables that user to edit Manufacturer data, part type data, as well as add manufacturers and part types when there is a need to add new ones. The Reports page has 4 reports that we think are the basic needs for the cath lab business, which are: Stent usage report, Stent cost report, Market share report, and a reorder report that shows all parts that are below the recommended par level for that particular product.

**Phase III**

**3.1 Back End**

**3.1.1 Tables**



This is a screen shot of the tables at the finished stage. We added a Status table to give each product a status in the database. 1. Available, 2. Expired, 3. Used, 4. Scanned. These statuses help us in the use of stored procedures to pull products for reports.

**3.1.2 Stored Procedures**

**Get Pars**

USE [cathlab]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CheckPars] Script Date: 11/22/2013 4:15:23 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: Dream Team

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

ALTER PROCEDURE [dbo].[CheckPars]

-- Add the parameters for the stored procedure here

--<@Param1, sysname, @p1> <Datatype\_For\_Param1, , int> = <Default\_Value\_For\_Param1, , 0>,

--<@Param2, sysname, @p2> <Datatype\_For\_Param2, , int> = <Default\_Value\_For\_Param2, , 0>

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

select PartNumber

,Counts.Count

,pn.Par

,m.Name

from ( SELECT distinct PartNumber, count(\*) as Count

FROM Product as p

GROUP BY PartNumber) as Counts

inner join PartNumber pn on Counts.PartNumber = pn.PartNum

inner join Manufacturer m on pn.ManufacturerID = m.ID

where Counts.Count < pn.Par

order by m.Name

END

**Manufacturer Stock**

USE [cathlab]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[getManufacturer\_STOCK] Script Date: 11/22/2013 4:41:51 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: Dream Team

-- Create date: November 2013

-- Description: crunch time ninja

-- =============================================

ALTER PROCEDURE [dbo].[getManufacturer\_STOCK]

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

SELECT count( Product.PartNumber ) as Number

,Manufacturer.Name

from Product join PartNumber on Product.PartNumber = PartNumber.PartNum

join Manufacturer on Manufacturer.ID = PartNumber.ManufacturerID

group by Manufacturer.Name

END

**Get Scanned**

USE [cathlab]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[getScanned] Script Date: 11/22/2013 4:44:35 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

ALTER PROCEDURE [dbo].[getScanned]

-- Add the parameters for the stored procedure here

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

select Product.PartNumber

,Product.ID

,Product.SerialNumber

,Product.ExpirationDate

,Product.LotNumber

,PartNumber.NameSize

,Manufacturer.Name

from Product join PartNumber on Product.PartNumber = PartNumber.PartNum

join Manufacturer on Manufacturer.ID = PartNumber.ManufacturerID

where Product.StatusID = 4

order by Manufacturer.Name

END

**Get Stent**

USE [cathlab]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[getStents] Script Date: 11/22/2013 4:47:02 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: Luke Caraccioli

-- Create date: October 2013

-- Description: Gets the Name, Type, and Cost of Stents Used by Date

-- =============================================

ALTER PROCEDURE [dbo].[getStents]

@StartTime datetime,

@EndTime datetime

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

--SET NOCOUNT ON;

-- Insert statements for procedure here

select count(Product.ID) as Quantity

,(Manufacturer.Name + ': ' + ProductType.Type) as Manufacturer

,sum(Partnumber.Cost) as Cost

from product

join PartNumber on Product.PartNumber = partnumber.PartNum

join Manufacturer on PartNumber.ManufacturerID = Manufacturer.ID

join ProductType on ProductType.ID = Partnumber.ProductTypeID

where DateUsed >= @StartTime and DateUsed <= @EndTime and Type like '%stent'

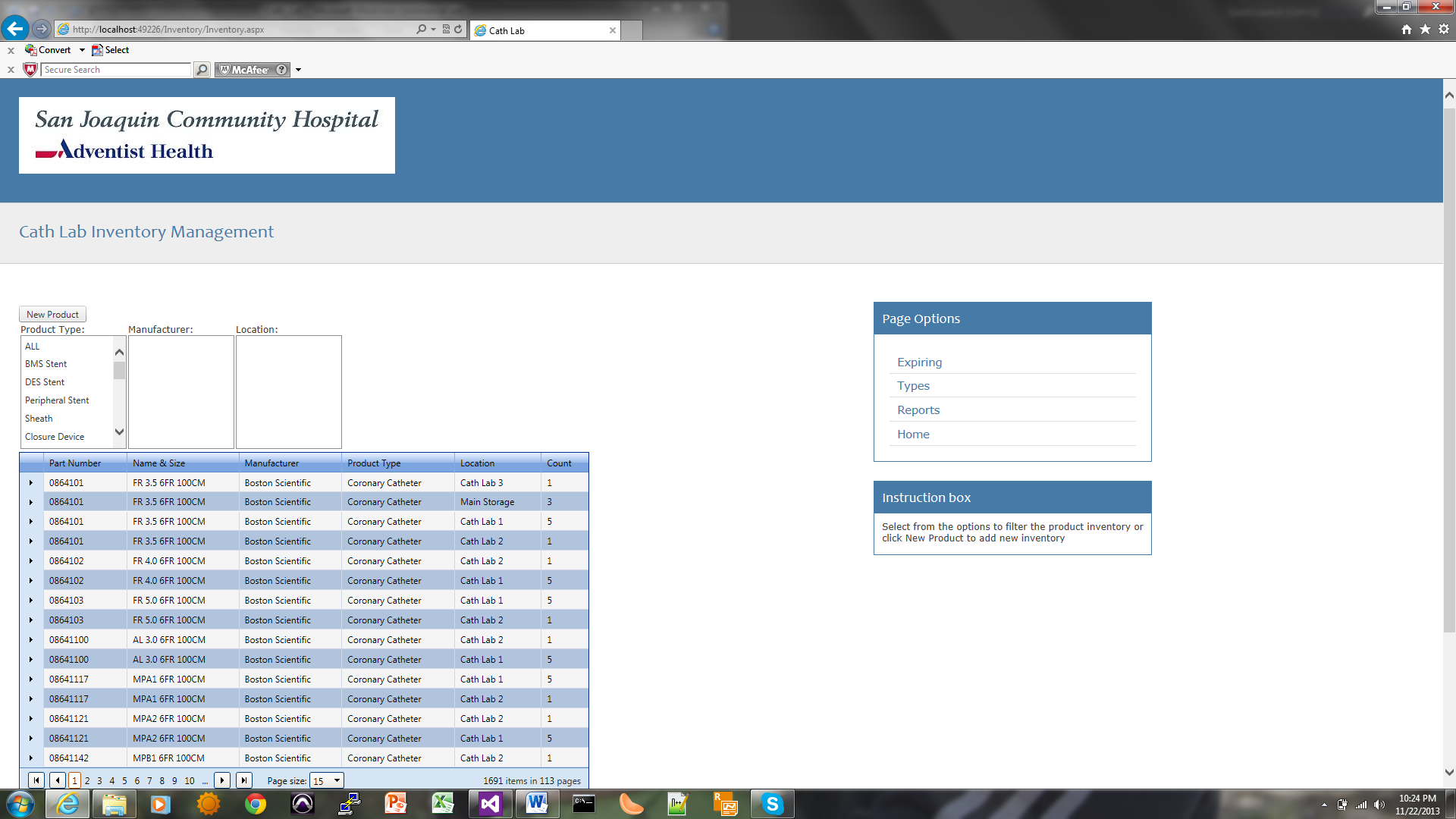
GROUP BY PartNumber.Cost, Manufacturer.Name, ProductType.Type

order by Manufacturer.Name

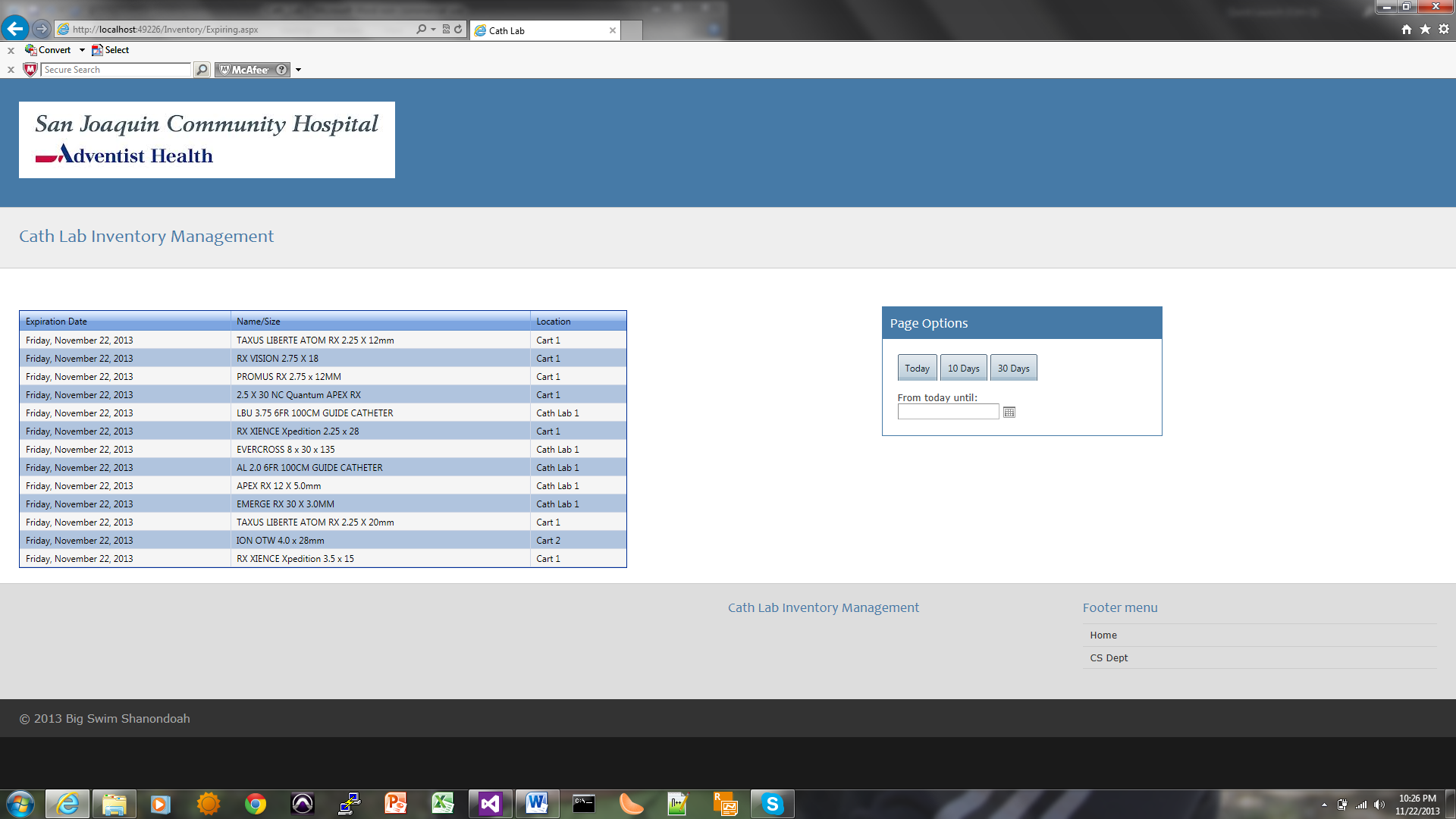
END

**3.2 Front End**

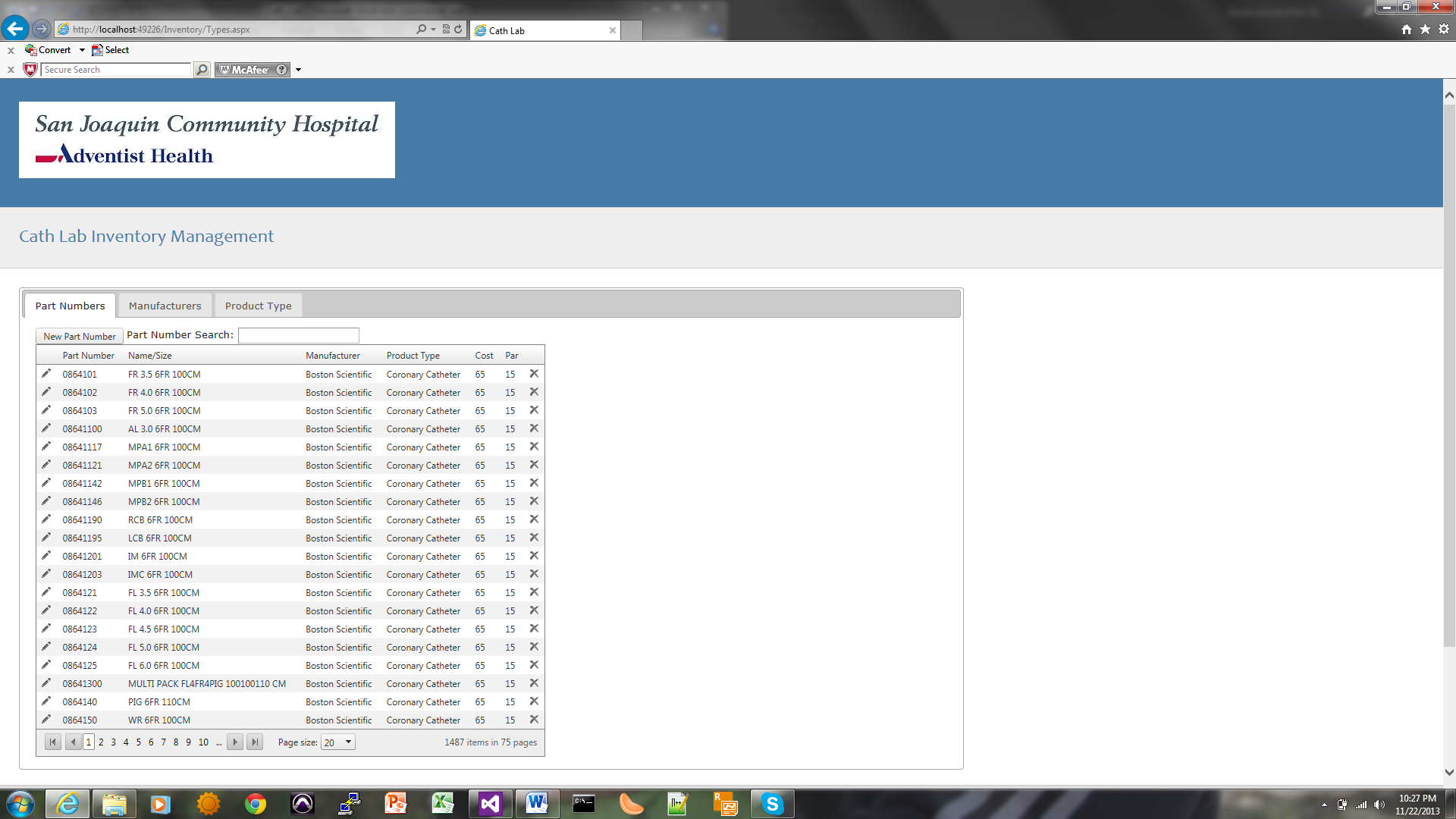
**Inventory Page:**



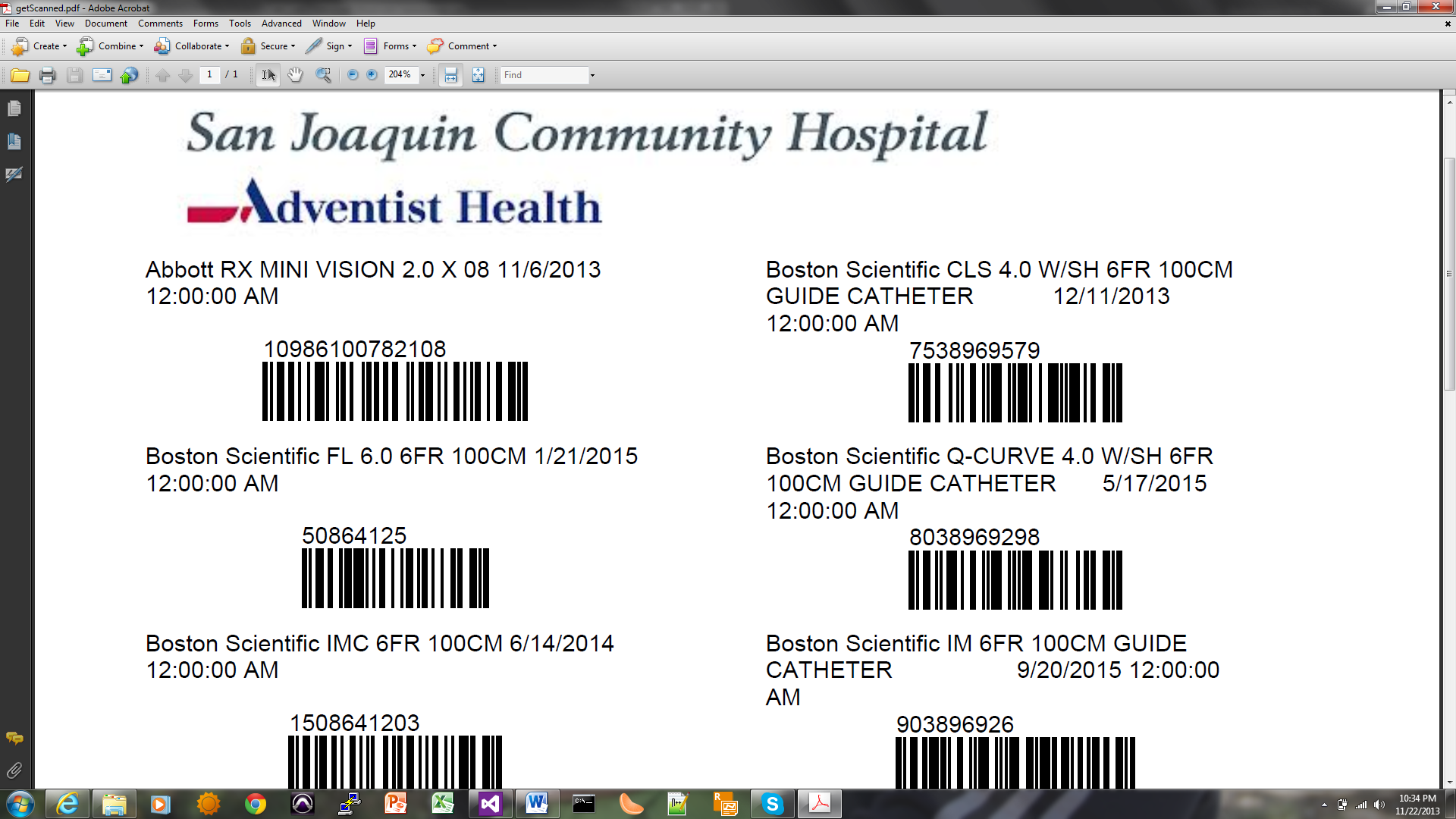
**Expiring Page:**



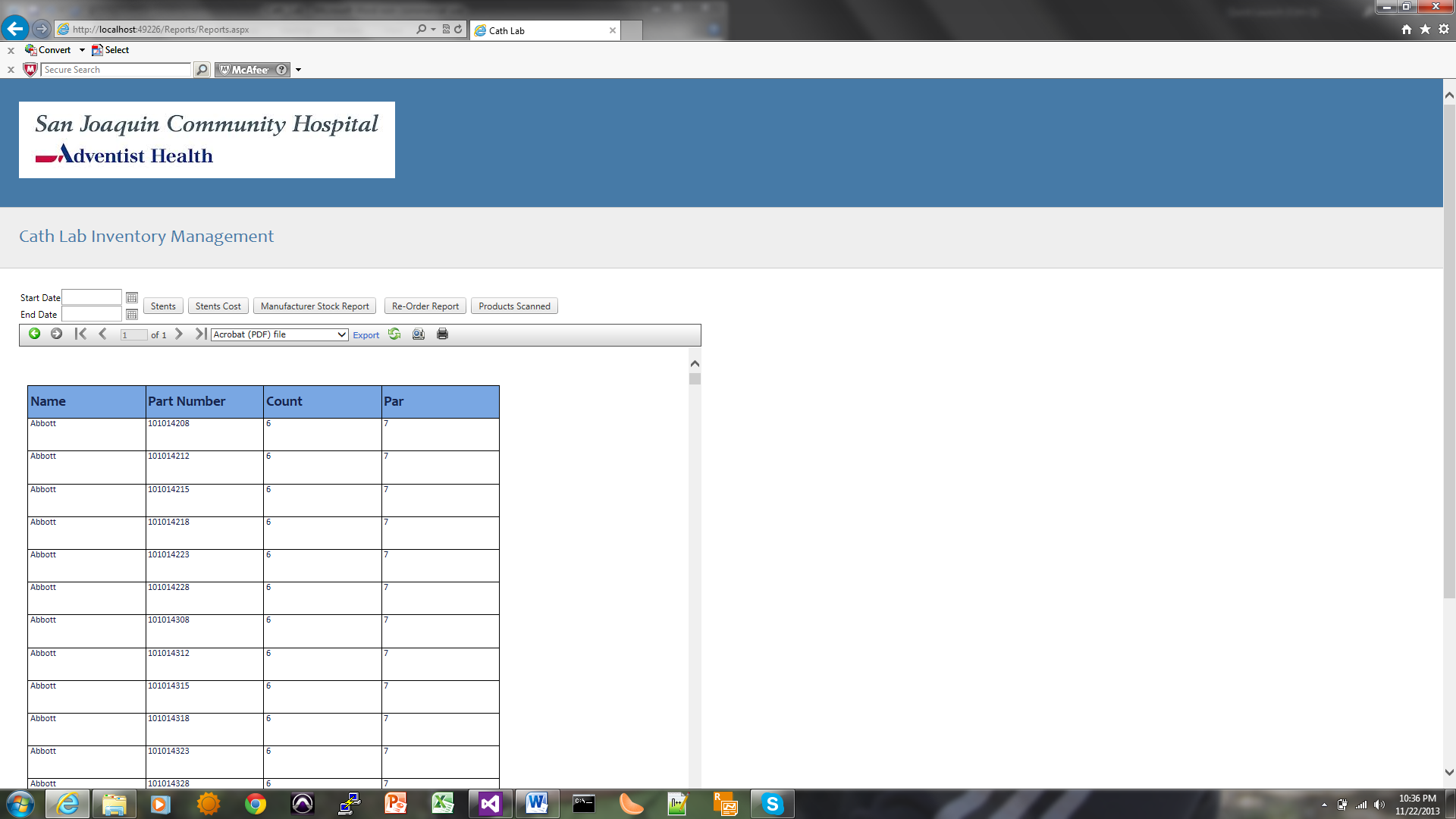
Types Page:



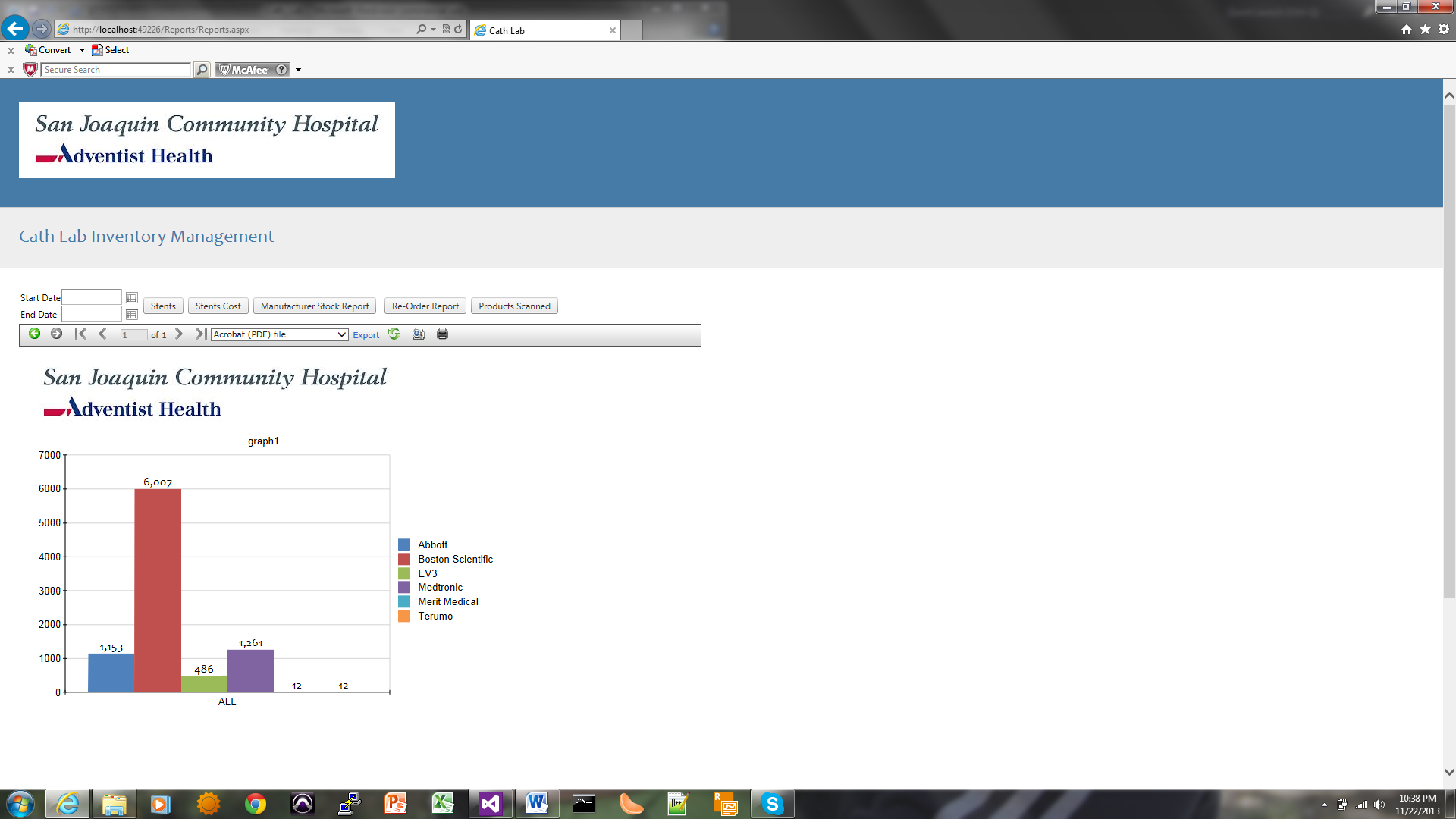
**Bar Code Report:**



**Re-Order Report:**



**Manufacturer Stock Report:**



**Stent Usage Report:**

