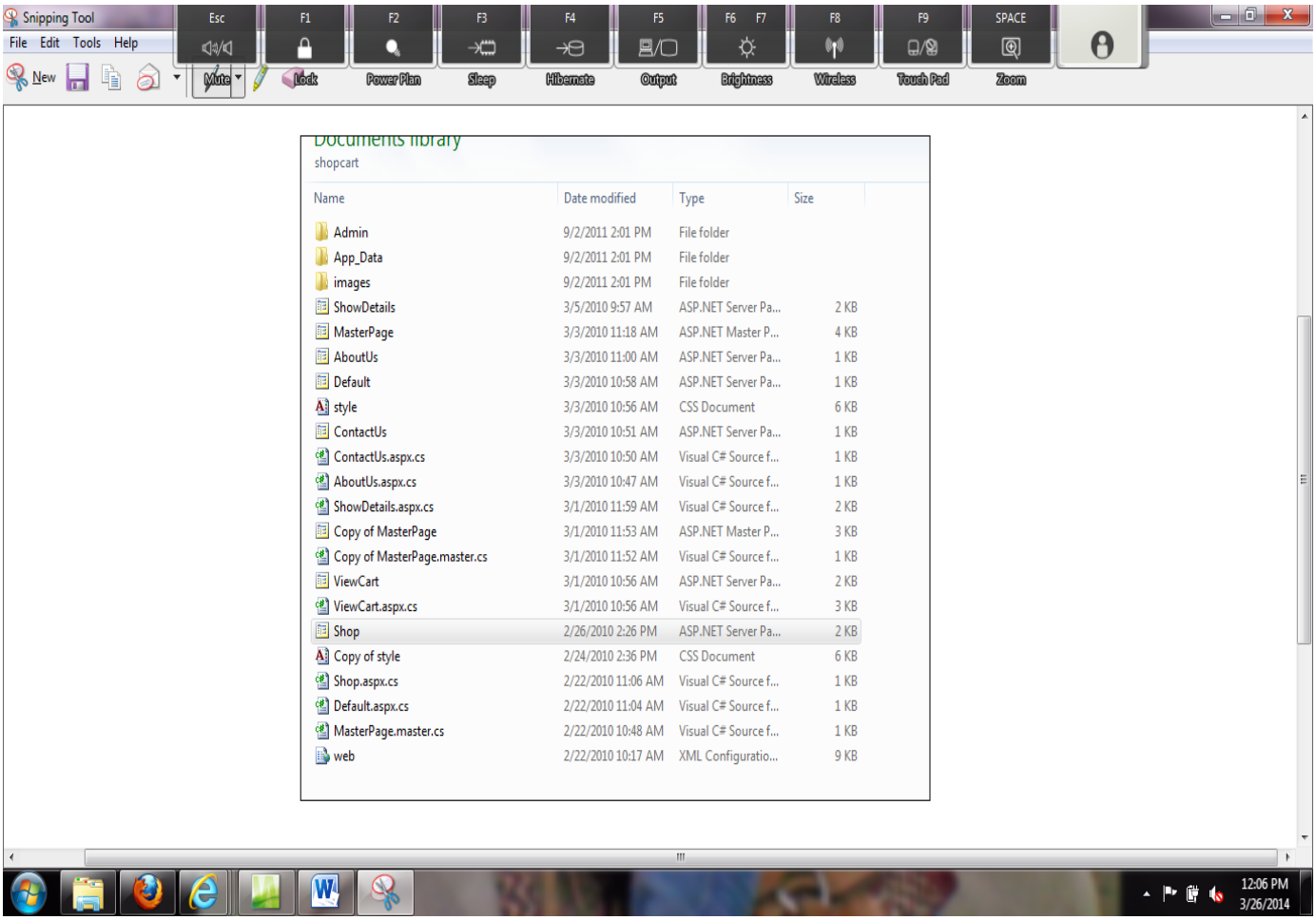
This was a fully function Shopping Cart website, which was hosted on the university's server, which I no longer can access. I received an A on this assignment. The directions are listed below for your reference and overview.



Shopping Cart Instructions Setup Database / Edit & Delete Products

Create a new ASP.Net Web Site on your local machine. Name it ShopCart .

A. Run the application. Register a user with the login links.

B. Refresh the App_Data folder in the Solution Explorer. You will see a ASPNETDB.MDF SQL server File.

C. Inspect the layout of the database created for you in the Server Explorer.

D. Inspect the other aspx files created for you.

1. **Create a Master Page (site.master)**
   A. Download a web page design, unzip it, and copy the files to the Solution Explorer.
   B. Merge the index.html (from the downloaded site) with site.master
      Make sure to keep the following code in the **header**:

```
<asp:ContentPlaceHolder ID="HeadContent" runat="server">
</asp:ContentPlaceHolder>
```

      Make sure to keep the following code in the main content area of the **body**:

```
<div id="content">
 <form id="Form1" runat="server">
```

```
            <asp:ContentPlaceHolder ID="MainContent" runat="server"/>
        </form>
    </div>
```

Modify the location of the style sheet, Add the login division, change the title, etc.

C. Make a backup of the site.master when you get it set up correctly.

2. Modify default.aspx and about.aspx. Add some information about the web site.

3. Make a directory called /ProductImages. Copy your product images to this folder.

4. **Maintain Products / Edit and Delete**A. Create an admin folder

B. Create a default.aspx file under the admin folder, linking it to the Master Page (site.master)

C. Create a "Products" table in ASPNETDB.MDF (Right click on Tables, Add New Table)

a. Minimum Fields (set all text fields sizes as small as needed):

ID (Right click on it to Set Primary Key)

(Go to the properties and set Identity Specification -> (Is Identity) to **YES**

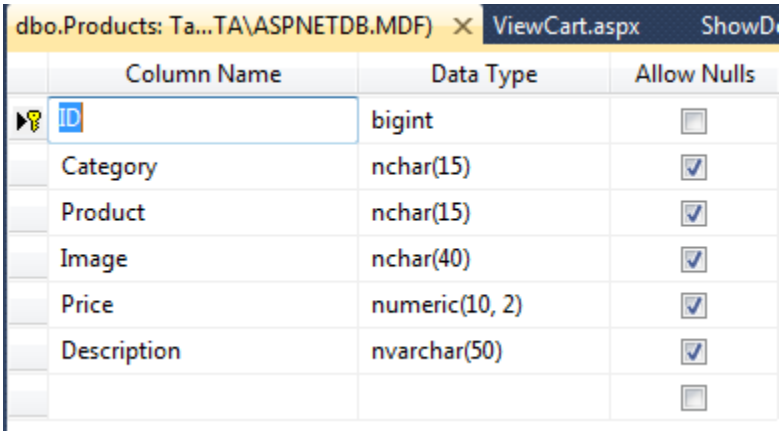
Category

Product

Price

Description

Image (This field need to be long enough to hold a file name starting with ~/ProductImages/)



b. Save the table as **Products.** Add some data. (To add data, right click on table ->Show Table Data)

Image values needs to be in the form of **~/ProductImages/some.jpg**

D. Drag the SQLDataSource into the ContentPlaceHolder on the /admin/default.aspx page

Configure the datasource (Rename SQLDataSource1 to **dataMaintainProducts**)

1. Pull down and choose the ASPNETDB.MDF database in the APP_DATA folder
2. Have the wizard create the Connection String by clicking NEXT

3. Choose the Products table
4. Specify columns from a table or view. Select ***** for all fields.
5. Click the "Order by" button to specify how the data should be sorted (Category/Product)
6. Click the "Advanced" tab and click the checkbox to create all of the CRUD SQL
7. Continue with the Wizard (Next…) . Click on Test Query if you want. Finish the Wizard.

E. Drag the GridView into the ContentPlaceHolder
Set up the GridView (Name it **gvEditDeleteProducts**)
Choose Data Source as dataMaintainProducts

1. Enable "Paging", "Sorting", "Editing", "Deleting"
2. Choose "Edit Columns"
3. Delete the ID and Image fields from the "Selected" fields
4. Add an ImageField from the available fields
   From its properties: Set DataImageUrlField -> Image
   Styles -> ControlStyle ->Height / Width to set image size
5. Set the DataFormatString on the Price field to {0:c}
6. Modyify the ButtonType on the CommandField, if desired
7. Highlight the CommandField and click on "Convert this field to a Template Field"
8. Click OK to exit the "Edit Columns" . Choose "Auto Format" the Grid
9. Add the **Image** field to the GridView's properties DataKeyName (with the ID)
10. Go to the source view, and find the gvEditDeleteProducts code. Modify the following code:

```
<asp:Button ID="Button2" runat="server" CausesValidation="False" OnClientClick="return
confirm('Are you sure you want to Delete?')" CommandName="Delete" Text="Delete" />
```

by adding the

```
OnClientClick="return confirm('Are you sure you want to Delete?')"
```

phrase.

You should now be able to Edit and Delete products.

Maintain Products - Insert/ Add New

1. Drag a button to the ContentPlaceHolder for "Add New Product" (btnAddNewProduct)
2. Drag the DetailsView (**dvAddProduct**) into the Content Panel
   -- Choose Data Source as dataMaintainProducts
   -- Enable "Inserting"
   -- Edit Fields
      ID - InsertVisible = False
      Convert the Image field to a Template Field
      OK to close Edit Fields
   -- Edit Templates (*****_**InsertItem**_ Template -- Choose from drop down list)
      Remove the Text Field
      Drag the fileupload into the template [End Template Editing]
   - Auto Format the Details View
5. Program the following events:

```
protected void Page_Load(object sender, EventArgs e)
{
    gvEditDeleteProducts.Visible = true;
    dvAddProduct.Visible = false;
    btnAddNewProduct.Visible = true;
}
protected void btnAddNewProduct_Click(object sender, EventArgs e)
{
    btnAddNewProduct.Visible = false;
    gvEditDeleteProducts.Visible = false;
    dvAddProduct.Visible = true;
    dvAddProduct.ChangeMode(DetailsViewMode.Insert); //put into insert mode
}
protected void dvAddProduct _ItemInserted(object sender, DetailsViewInsertedEventArgs e)
{
    btnAddNewProduct.Visible = true;
    gvEditDeleteProducts.Visible  = true;
    gvEditDeleteProducts.DataBind();  //Refresh the data
    dvAddProduct.Visible = false;
}
protected void dvAddProduct _ItemInserting(object sender,  DetailsViewInsertEventArgs e)
    {   FileUpload flUp = (FileUpload)(dvAddProduct.FindControl("FileUpload1"));
        if (flUp.HasFile)
        {   dataMaintainProducts.InsertParameters["Image"].DefaultValue =   "~/ProductImages/" +
flUp.FileName;
            string filePath = Server.MapPath("~/ProductImages/" + flUp.FileName);
            flUp.SaveAs(filePath);
        }
        else
        {
            dataMaintainProducts.InsertParameters["Image"].DefaultValue=
                "~/ProductImages/noimage.jpg";
        }
    }
```

You should now be able to Add products and upload images.

Modify the Menu on Master Page to list distinct brands from the database.

1. Drag the SQLDataSource into the Master Page (site.master). Make sure it is not in the ContentPlaceHolder

Configure the datasource (rename it to **dataProductCategories**)

- Pull down and choose "Connection String" (Connection to ASPNETDB.MDF)
- Specify a custom SQL statement or stored procedure (Get a distinct list of Categories)

Click checkbox on Category and checkbox on "Return only unique rows",

Clock on "Order By" an choose Category

Statement should look like this: SELECT Distinct Category FROM Products ORDER BY Category

2. Place a Data "Repeater" into your menu where you want a list of "Categories"

Goal: Make the menu choice go to shop.aspx and pass a QueryString variable called Category.

Go to the Source page and insert an item template similar to:

```
<asp:Repeater ID="Repeater1" runat="server"  DataSourceID="dataProductCategories">
    <itemtemplate>
     <li>
     <a href="/shopcart/shop.aspx?Category=<%# DataBinder.Eval(Container.DataItem, "Category") %>" >
           <%# DataBinder.Eval(Container.DataItem, "Category") %>
     </a>
     </li>
    </itemtemplate>
</asp:Repeater>
```

Shop.aspx

1. Create a /shop.aspx web form and apply the master page

2. Drag the SQLDataSource into the ContentPlaceHolder on the shop.aspx page

Configure the datasource (Rename to dataProductsByCategory)

- Pull down and choose "Connection String" (Connection to ASPNETDB.MDF)
- Specify columns from a table or view. Select * for all fields.
- Click the "Where" button to specify column "Category" = the QueryString "Category" (set a default)
- Click the "Order by" button to specify how the data should be sorted (by Product?)

3. Drag the GridView into the ContentPlaceHolder

Set up the GridView (Rename to gvProductsByCategory)

Choose Data Source as dataProductsByCategory

Enable "Paging"

Edit Columns

Change the Image to an ImageField as was done on the admin page (E.4. on page 2)

Add a HyperLink Field. **Change the Text property to "View"**

- Set the DataNavigateURLFIeld to **ID**

- Set the DataNavigateURLFormatString to **ShowDetails.aspx?ID={0}**

Create the ShowDetails.aspx File on your own. Use a DetailsView (name it dvProduct) and an SQLDataSource that uses the ID QueryString variable passed from shop.asp. Leave room for a form that will allow the user to type in a Quantity to add to the cart.

Add to Cart / View Cart

1. On ShowDetails.aspx

Create a place to enter the quantity and add to cart.

Type in the word “Quantity” and next to it add a Text Box (used to enter the quantity –name this txtQuantity)

Copy the following validation directly under the txtQuantitiy in the code (Omit the textbox in the copy)

```
<asp:TextBox ID="txtQuantity" runat="server">0</asp:TextBox>

<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
ControlToValidate="txtQuantity" ErrorMessage="Enter only Numbers"
ValidationExpression="[0-9]*"></asp:RegularExpressionValidator>
```

Add a button (btnAddToCart / Add to Cart)

Add to the dvProduct, DataKeyNames: ID, Category, Product, Price

(These are used to get the values for the cookies in the code below)

Program the button click event for btnAddToCart to create the cookie and redirect to ViewCart.aspx

```
protected void btnAddToCart_Click(object sender, EventArgs e)
{
   HttpCookie myCookie = new
               HttpCookie(dvProduct.DataKey.Value.ToString());
   myCookie["Category"] = dvProduct.DataKey["Category"].ToString();
   myCookie["Product"]= dvProduct.DataKey["Product"].ToString();

   myCookie["Quantity"] = txtQuantity.Text;
   myCookie["Price"] = dvProduct.DataKey["Price"].ToString();
   myCookie.Expires = DateTime.Now.AddDays(1d);
   Response.Cookies.Add(myCookie);
   Response.Redirect("ViewCart.aspx", true);
}
```

2. Create a new web form: ViewCart.aspx

From the toolbox, drag a TABLE to the ContentPlaceHolder. Name it tblCart.

Go to the **Rows** property (Collection) .

Add 1 row. Go to the **Cells** property (Collection) of the Row and add 5 columns.

Label the by setting the **Text** property of columns (0-4) with

Category, Product, Price, Quantity, Total

Below the table, type in the words “Order Total:” and next to it add from the toolbox a Label (call it lblOrderTotal)

Program the Page Load event with the following:

```
protected void Page_Load(object sender, EventArgs e)    {
    double total = 0.0;
    double price = 0.0;
    int quantity = 0;

    System.Collections.Specialized.NameValueCollection UserInfoCookieCollection;
    for(int i = 0; i< Request.Cookies.Count ; i++)
    {
        UserInfoCookieCollection = Request.Cookies[i].Values;

        price = Convert.ToDouble(Server.HtmlDecode(UserInfoCookieCollection["Price"]));
        quantity = Convert.ToInt16(Server.HtmlDecode(UserInfoCookieCollection["Quantity"]));

        TableRow tr = new TableRow();
        TableCell tcCategory = new TableCell();
        TableCell tcProduct = new TableCell();
        TableCell tcQuantity = new TableCell();
        TableCell tcPrice = new TableCell();
        TableCell tcProductTotal = new TableCell();

        tcCategory.Text = Server.HtmlDecode(UserInfoCookieCollection["Category"]);
        tr.Cells.Add(tcCategory);
        tcProduct.Text= Server.HtmlDecode(UserInfoCookieCollection["Product"]);
        tr.Cells.Add(tcProduct);
        tcPrice.Text = String.Format("{0:c}", price);
        tcPrice.HorizontalAlign = HorizontalAlign.Right;
        tr.Cells.Add(tcPrice);
        tcQuantity.Text= Convert.ToString(quantity);
        tr.Cells.Add(tcQuantity);
        tcProductTotal.Text = String.Format("{0:c}", quantity * price);
        tcProductTotal.HorizontalAlign = HorizontalAlign.Right;
        tr.Cells.Add(tcProductTotal);
        if (quantity != 0 )
            tblCart.Rows.Add(tr);
        total = total + quantity*price;
    }
    lblOrderTotal.Text = String.Format("{0:c}", total);
}
```

Play with the format the table to make it look good!

Add a button (btnClearCart) to ViewCart.aspx with a "Clear Cart" caption.

Program the click event for the button.

```
protected void btnClearCart_Click(object sender, EventArgs e)
    {
        HttpCookie aCookie;
        string cookieName;
        int limit = Request.Cookies.Count;
        for (int i = 0; i < limit; i++)
        {
            cookieName = Request.Cookies[i].Name;
```

```
            aCookie = new HttpCookie(cookieName);
            aCookie.Expires = DateTime.Now.AddDays(-1);
            Response.Cookies.Add(aCookie);
        }
        Response.Redirect("ViewCart.aspx", true);
    }
```

Ad Rotator

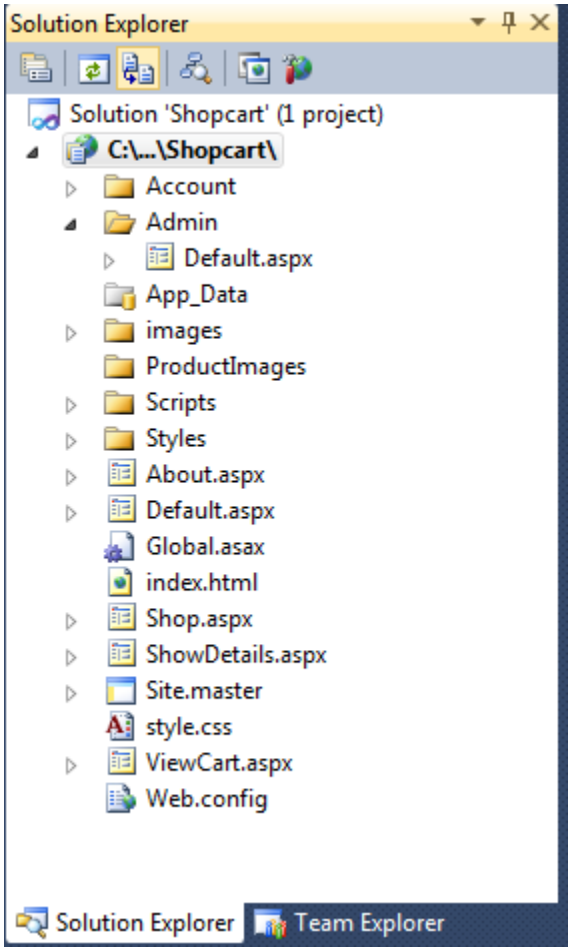
Right click on App_Data folder, Add New Item, XML File. Name it Ads.xml.
Copy the following code and paste it in the XML File. Modify as necessary.

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
<Ad>
  <ImageUrl>images/titleist.jpg</ImageUrl>
  <NavigateUrl>http://www.titleist.com</NavigateUrl>
  <AlternateText>Titleist Golf Balls</AlternateText>
  <Keyword>Titleist</Keyword>
  <Impressions>25</Impressions>
</Ad>
<Ad>
  <ImageUrl>images/nike.jpg</ImageUrl>
  <NavigateUrl>http://www.nike.com</NavigateUrl>
  <AlternateText>Nike Golf Balls</AlternateText>
  <Keyword>Nike</Keyword>
  <Impressions>25</Impressions>
</Ad>
<Ad>
  <ImageUrl>images/precept.jpg</ImageUrl>
  <NavigateUrl>http://precept.com</NavigateUrl>
  <AlternateText>Precept Golf Balls</AlternateText>
  <Keyword>Precept</Keyword>
  <Impressions>50</Impressions>
</Ad>
</Advertisements>
```

On the Master Page, Drag an XML datasource to the page (XMLDataSource1).
Make sure you did not put it into the ContentPlaceHolder.
Configure the datasource to point to App_Data/Ads.xml.

On the Master Page, Drag an AddRotator to the page.
Make sure you did not put it into the ContentPanel.
Set its datasource to XMLDataSource1.

Your final directory structure should look like the following:



Folders that were added: /images, /ProductImages, /Admin

Files that were added:
 /Admin/Default.aspx, /Shop.aspx, /ShowDetails.aspx, /ViewCart.aspx

The /index.html file came from the downloaded web design. Once merged into site.master, the file is no longer needed.

You may or may not need the /Styles folder.

PUBLISHING THE WEB SITE

Modify Links

Before publishing, links in site.master may need to be modified. All links in the format of

<a href=”/ShopCart/Default.aspx”>
will need to be changed to
<a href=”/yourid/ShopCart/Default.aspx”>

This will adjust the links as necessary to run from the server configuration that we have.

Publish the web site by going to Build -> Publish Web Site.
Type in the Target location as shown on the back Window in the diagram below.

Double click the dot-dot-dot to get the top Window.