Intern Program: Week 3 Assignment

**Purpose:** This document outlines the assignment that will be handed out following the first review for the intern program. This project is to be worked on while you complete the assigned readings for week 3, and must be completed for your Week 3 review.

# Week 3 Requirements:

For this week you will be building the basic ASP.NET Project, and will be constructing the following pages.

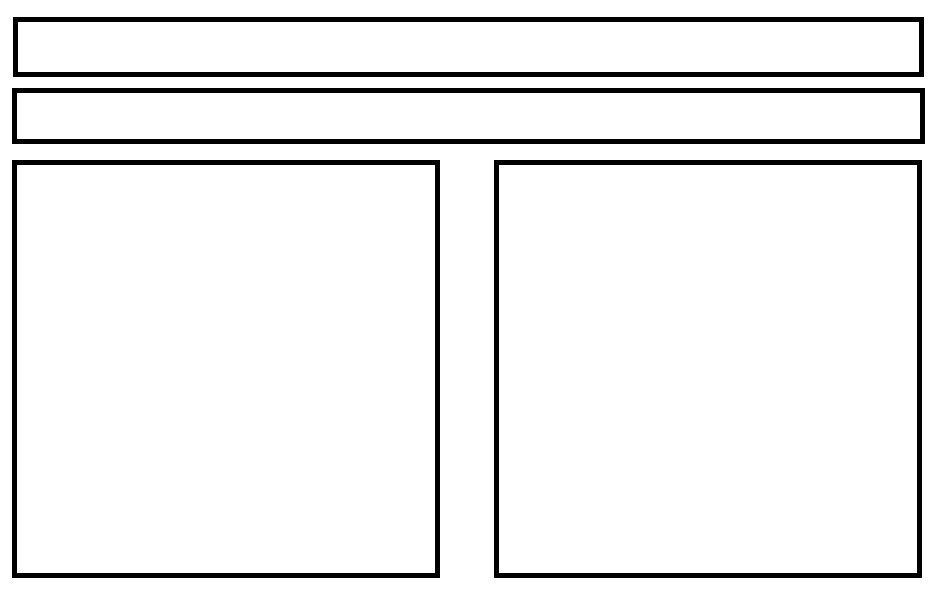
* **Default.aspx:** This page will be a default page of the application, where the user is provided with information about the application. It will function as the landing page of the application.
* **ProductsAdmin.aspx:** This page will allow administrators to display a listing of all products in the application for management by the administrator.
* **ProductAdminDetails.aspx:** This page will allow for the administrator to see all information on a product from the Product table. Specifically allowing them to update this information as necessary.
* **CustomersAdmin.aspx:** This page will allow for administrators to see a listing of all users in the application, and the associated details for each user account.
* **CustomerAdminDetails.aspx:** This page will allow for updating of basic information for a user, this will include displaying the contents of the User and their Addresses.
* **GitHub:** Once this assignment is completed you will publish to your own GitHub repository.

# Notes

When creating your project, you may select VB or C# as the language. Choose the template Web -> ASP.NET Web Application. After naming the project and clicking OK, you will be prompted to select the ASP.NET template, choose **Web Forms and check the “MVC” checkbox** before clicking OK.

## Default.aspx:

This page will allow users to read information about the application and provide them with a place to land when first arriving to the web application. The page may implement the following layout.



## Master Page:

This application will require you to create a master page for your application, and this should be called “Site.Master”. This page will be the master page for the general pages of the application that do not fall on the customer side of the application. Default.aspx will implement this page.

## CustomersAdmin.aspx:

This page will make use of a new stored procedure called “spGetUsers” which will accept no parameters and return a listing of all records in the User table. It will display this information using a GridView, and will have the following columns:

* UserID
* UserName
* EmailAddress

Additionally this grid will have a column with an “Edit” link displaying, and should be located on the far left hand side of the grid. This page will make use of a sql data source control.

The grid should be sortable by clicking on the column headings and have paging set to 50.

Also, this page should implement a “DetailsView” control to allow for adding records to the User table. This should use a stored procedure called “spAddUser” which will accept the following parameters:

* @UserName
* @EmailAddress

## CustomerAdminDetails.aspx:

This page will make use of a new stored procedure called “spGetUser”; this stored procedure will take in a parameter of “UserID” and return the details of that user from User. The UserID should be passed to this page as part of the query string to be created in the edit link on the CustomerAdmin Page.

This page will make use of a details view to provide the ability to update the information in the User table for this specific record. This should use a new procedure called “spUpdateUser” which will accept the following Parameters:

* @UserID
* @UserName
* @EmailAddress

This page should not display a field showing the user’s password, which should be hidden from the user.

There should also be a button that allows for the deleting of users. This should make use of a new stored procedure called “spDeleteUser”, and will accept the “@UserID”. This procedure should delete the selected record from the User table.

It will make use of a new procedure called spGetUserAddresses to return this information. The stored procedure should accept a parameter @UserID, and return all the addresses associated with that user. The grid should display the following columns:

* AddressID
* Address1
* Address2
* Address3
* City
* State
* ZIpCode

The grid should be sortable by clicking on the column headings and have paging set to 50.

## ProductsAdmin.aspx:

This page will make use of the spGetProducts procedure from last week to return a listing of all products in the application. Those products will be displayed in a grid format, with the following columns showing:

* ProductID
* Product Name
* IsPublished
* Price

Additionally this grid will have a column with an “Edit” link displaying, and should be located on the far left hand side of the grid. This page will make use of a sql data source control.

The grid should be sortable by clicking on the column headings and have paging set to 50.

Also, this page should implement a “DetailsView” control to allow for adding records to the Product table. This should use a stored procedure called “spAddProduct” which will accept the following paramters:

* @ProductName
* @Description
* @IsPublished
* @Price

## ProductAdminDetails.aspx:

This page will make use of a new stored procedure called “spGetProduct”; this stored procedure will take in a parameter of “ProductID” and return the details of that product from Product. The product ID should be passed to this page as part of the querystring to be created in the edit link on the ProductManagement Page.

This page will make use of a details view to provide the ability to update the information in the Product table for this specific record. This should use a new procedure called “spUpdateProduct” which will accept the following Parameters:

* @ProductID
* @ProductName
* @Description
* @IsPublished
* @Price

There should also be a button that allows for the deleting of products. This should make use of a new stored procedure called “spDeleteProduct”, and will accept the “@ProductID”. This procedure should delete the selected record from the Product table.

## CSS:

A separate CSS file should be implemented for use on all pages, all styling should happen in that shared CSS file (you may use Site.css). You may also use external libraries (such as Bootstrap, Font Awesome, etc.) but external libraries should not be modified.

## GitHub:

Create a new GitHub repository with StoreFront as the name, and publish it to your GitHub account (create a new one if necessary). Create this repository as a private repository that you will be sharing with the instructors for future reviews.