Tyler Truitt

Represent Printing

LinkedIn: www.linkedin.com/in/tylertruitt

Email: admin@trucoding.com

trucoding.com

"Hello! My name is Tyler Truitt, and I am a 26 year old student at Centriq Training. I am from Melbourne, FL, but I have lived a few places, and I would call Kansas City my home. I have taken a variety of College Courses since the age of 18 wanting to pursue an Engineering degree, but the only issue with that is that I did not know what type of Engineering I wanted to pursue. After working in the Restaurant, Healthcare, and General Labor industries for a number of years, I decided it was time for a change. I want to pursue a career where I am able to display my innovative skills and assets which are problem solving and engineering, and that is when I decided to pursue Software Development."

"I heard about Centriq Training from a family friend, and hearing about how short the program was, I contacted Russ for more information. I was really happy with what I saw upon entering the school, and that gave me hope that if I work really hard at this then I can make a career out of it. The best part of Web Development is that there are always more concepts to learn."

Requirements

Project Requirements:

This is a Print Shop application designed in ASP.NET MVC to display the current skill-set which I have obtained at Centriq Training. The project uses Entity Framework as an Object Relational Mapper to manage the data of the shop.

The Minimum requirements for this project are as follows:

Manage Departments:

- Only available to admin users
- Index view should show all departments
- CRUD functionality (**Soft Delete**)
- Index view with all active departments

Manage Employees:

- Only available to admin users
- Index view should be set to only active employees but be filterable to inactive employees
- CRUD functionality (**Soft-Delete**)
- Upon inserting or editing the employee they are taken to the index view with all active employees

Support Tickets:

- Index view should be set to only "open" (pending & In process) requests
- Anonymous users can view master-detail read-only mode
- If logged in, can insert new request (but may not edit)
- Authenticated users should be able to filter between either Open (Pending or In Process) and Resolved Requests

- If logged in as an Administrator or Tech, users should be able to access edit functionality in details view and toggle status between Resolved and In Process from the master view
- Upon completing the new request (or clicking button to see Open requests) they are taken to the master view with all open requests
- CRUD functionality (**Soft-Delete**)

All New Support requests should:

- Be in Pending Status automatically. (No user activity/selection)
- Have a null closed date
- Should NOT be able to see the Tech Notes field
- Allow for null tech id in the creation
- The only viewable fields for Authenticated NON ADMIN users should be the Requesting Employee and Requestor Notes

Notes:

- BASIC Requirements should be fulfilled BEFORE any additional functionality is added
- jQuery DatePicker controls should be used for any dates that are not automatically added
- All non-PK ID selections for Inserting and Updating should be done with drop down lists
- Notes fields will need to be converted to Multi-line text boxes
- All required fields should be validated
- Once you have base functionality your project should be backed up and not touched

Example Status Records: (know which constitute open/closed)

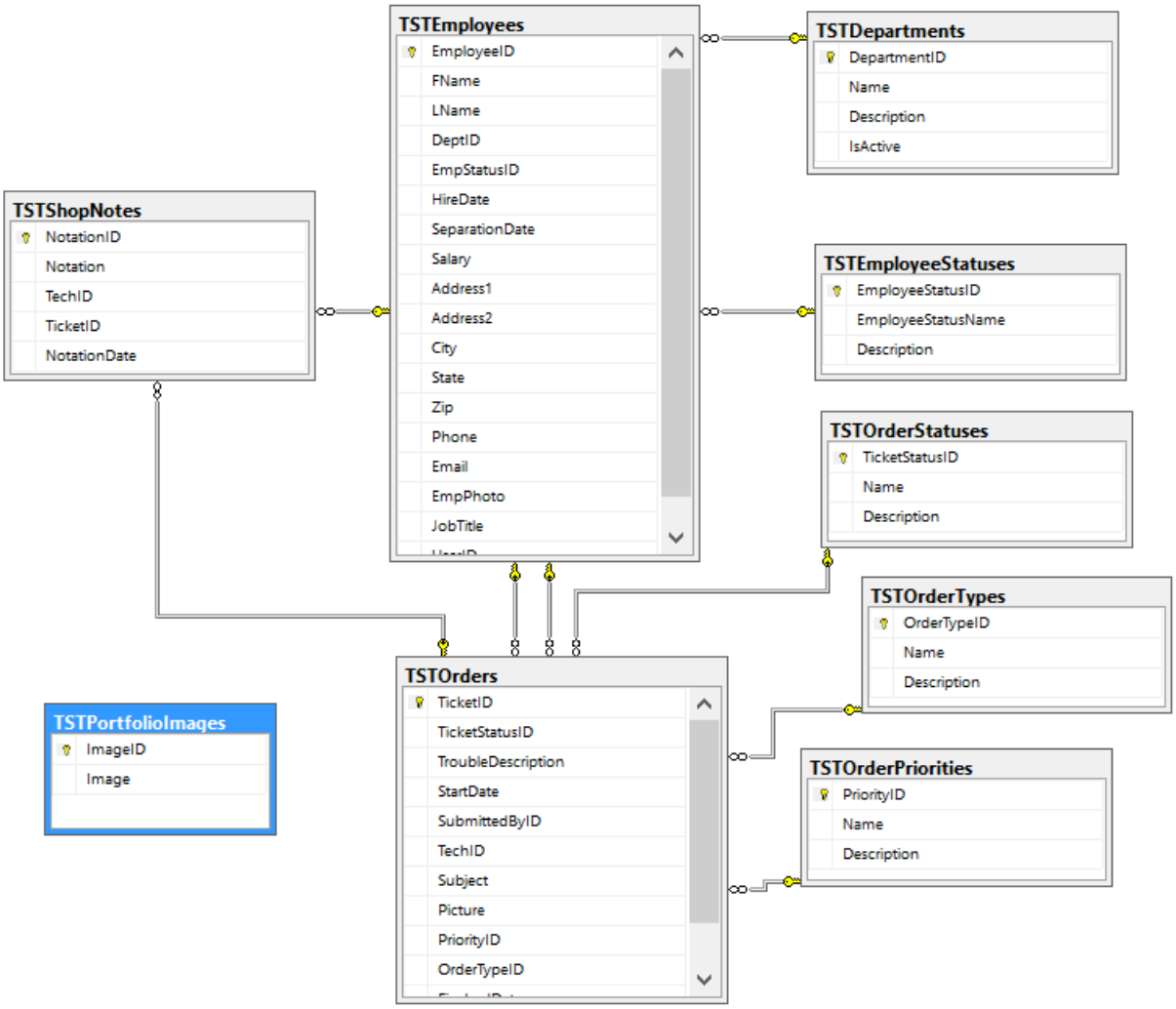
- Pending
- Assigned
- In Process
- On Hold
- Resolved

The base requirements for this projects were to have the following tables:

- TSTOrders
- TSTEmployees
- TSTDepartments
- TSTShopNotes
- TSTOrderPriorities

Some additional tables have been added to my database-structure

- TSTEmployeeStatuses
- TSTOrderStatuses
- TSTOrderTypes
- TSTPortfolioImages



About The Project

This project entitled "Represent Printing" represents a print shop with four roles. These roles are entitled "Employee", "Manager", "HR", and "Owner". The average user is able to log in and see open orders that have been received, but none of the orders that have been assigned to a printer or have been worked on yet. Users in the role of "Employee" can view more than just received orders, they can toggle between received, assigned, and completed orders. This role is able to log in and create new(open) orders and view their details, but users in the role of "Manager" need to assign each order to a printer or shop-worker. The "Manager" can also delete orders, as well as edit them. Users In the "Manager" role can also access the create, update, or delete any images in the portfolio, which is established on my site. The "HR" role is there to deal with the Employee relations, therefore they can only view the open orders in "read-only" mode just like an anonymous user. However, they have

access to Employee records with full CRUD(Create, Update, Delete) of the employee records.

USE CASE

The Base Requirements of the project only require three levels of access, or three roles. I chose to have four roles for this particular project to satisfy the role-requirements of being a Print Shop.

	Owner	**HR**	**Manager**	**Employee**	**Unauthenticated Users**
Employees					
Index	X	X			
Details	X	X			
Create	X	X			
Edit	X	X			
Delete	X	X			
Departments					
Index	X	X			
Details	X	X			
Create	X	X			
Edit	X	X			
Delete	X	X			
Employee Statuses					
Index	X	X			
Details	X	X			
Create	X	X			
Edit	X	X			
Delete	X	X			
Orders					

Index	X	X-Only Unassigned	X	X-View All Orders	X-Only Unassigned
Details	X	X-Only Unassigned	X	X	X-Only Unassigned
Create	X		X	X	
Edit	X		X		
Delete	X		X		
Order Statuses					
Index	X		X		X-Only Unassigned
Details	X		X		X
Create	X				
Edit	X				
Delete	X				
Order Type					
Index	X		X		
Details	X		X		
Create	X				
Edit	X				
Delete	X				
Order Priorities					
Index	X		X		
Details	X		X		
Create	X		X		
Edit	X		X		

Delete	X		X		
Shop Notes(No Controller)					
Details	X		X	X	
Create	X		X	X	
Users Admin					
Index	X	X			
Details	X	X			X
Create	X	X			
Edit	X	X			
Delete	X	X			
Roles Admin					
Index	X				
Details	X				
Create	X				
Edit	X				
Delete	X				
Images Portfolio					
Index	X	X	X	X	X
Details	X		X		
Create	X		X		
Edit	X		X		
Delete	X		X		

Model Example

```
public class OrderMetadata
{
    public int TicketID { get; set; }
    public int TicketStatusID { get; set; }
    public int SubmittedByID { get; set; }
    public Nullable<int> TechID { get; set; }

    [Display(Name = "Order Description")]
    [Required(ErrorMessage = "**This is a required field")]
    [StringLength(4000, ErrorMessage = "Please do not enter mor
    [UIHint("MultilineText")]
    public string TroubleDescription { get; set; }
    [Range(20,1000,ErrorMessage ="Please select no less that 20
    [Display(Name ="QTY")]
    [Required(ErrorMessage ="**This is a required field**")]
    public int Quantity { get; set; }

    [Display(Name = "Start Date")]
    [DisplayFormat(DataFormatString = "{0:d}",ApplyFormatInEdit
    //[Required(ErrorMessage = "**This is a required field")]
    public System.DateTime StartDate { get; set; }

    //[Required(ErrorMessage = "**This is a required field")]
    [StringLength(100, ErrorMessage = "Please do not enter more
    public string Subject { get; set; }

    [StringLength(100, ErrorMessage = "Please upload an image w
    public string Picture { get; set; }

    public int PriorityID { get; set; }

    public int OrderTypeID { get; set; }

    [Display(Name = "Finshed Date")]
    [DisplayFormat(NullDisplayText = "(still active)")]
    public Nullable<System.DateTime> FinshedDate { get; set; }

}

[MetadataType(typeof(OrderMetadata))]
public partial class TSTOrder
{

}
```

This is an example of one of my “Metadata” classes for my Orders Table. All of the validation for my project is done in this file, which is housed inside of a Class Library which serves as a layer of the project.

The Required Field Validators, have been commented out due to hard-coded fields being generated in the controller. The Required Field Validators were left to demonstrate, that they need to be there if “StartDate” and the “Subject” were not automatically generated in the Orders Controller.

Controller Example

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(
    [Bind(Include = "EmployeeID,FName,LName,DeptID,EmpStatusID,HireDate,Separat
TSTEmployee tSTEmployee, HttpPostedFileBase prodImage,
    string[] selectedRoles)//must match the name value of the input - casing do
{
    if (ModelState.IsValid)
    {

        Add UserID to Employee Object

        file uploade in create view for employee

        AutoGenerate Date of Create
        Autogenerate Employee Status
        Autogenerate Salary Based on Department chosen

        Autogenerate Position based on department


        //#region Disable any selection of an inactive department
        //tSTEmployee.TSTDepartment.IsActive = true;
        //#endregion//this one does not seem to work (return later

        db.TSTEmployees.Add(tSTEmployee);
        db.SaveChanges();
        return RedirectToAction("Index");
```

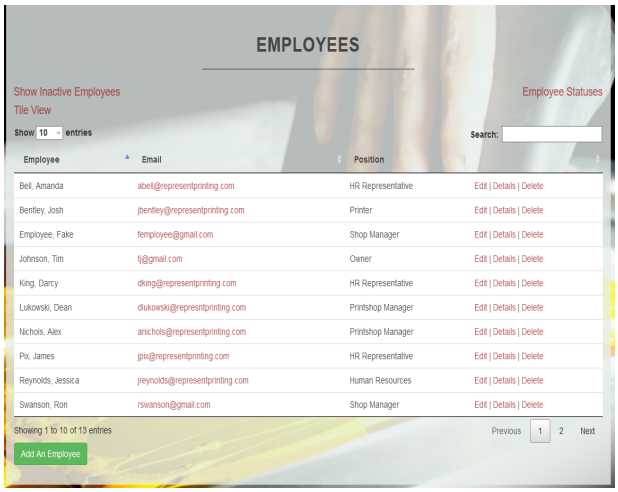
This is an example of my Create Post in my Employee Controller. All of the gray fields are regions that store code to auto-populate certain fields in the creation of an employee. Below are two examples of code to auto-populate certain fields in my Employee Controller.

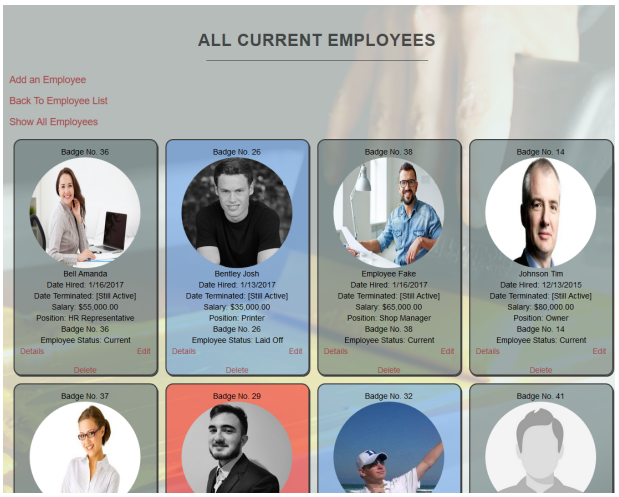
```
#region file uploade in create view for employee

string imageName = "noimage.jpg";
if (prodImage != null)
{
    imageName = prodImage.FileName;

    string ext = imageName.Substring(
        imageName.LastIndexOf('.'));

    string[] goodExts = new string[]
    {
    ".gif",".jpg",".bmp",".jpeg", ".png"
    };

    if (goodExts.Contains(ext))
    {
        imageName = Guid.NewGuid() + ext;

        prodImage.SaveAs(Server.MapPath("~/Content/img/employeephotos/" + imageName));
    }
    else
    {
        imageName = "noimage.jpg";
    }
}
tSTEmployee.EmpPhoto = imageName;
```

```
#region Autogenerate Salary Based on Department chosen

if (tSTEmployee.DeptID == 1)
{
    tSTEmployee.Salary = 55000m;
}
else if (tSTEmployee.DeptID == 2)
{
    tSTEmployee.Salary = 65000m;
}
else if (tSTEmployee.DeptID == 3)
{
    tSTEmployee.Salary = 30000m;
}
else
{
    tSTEmployee.Salary = 100000m;
}

#endregion
```

Unique Features

1. View Toggle Between “Tile” and “List” format in Employee and Order Indexes. The “List” view is the main Index view of the page, which has the client-side sorting and paging from datatables.net. This feature is very useful with companies with tables containing hundreds of records. The tile view has been color-coded by manipulation of inline styles with C# embedded code in Razor based on the employment status of the employee. It was useful to have that extra Employee Status Table outside the normal project requirements in order to display this.





2. Color-coding of different tiles in the Order’s Index based on Priority of the Order. The color red is orders of high priority, the color blue are orders with medium priority, and the color green have low priority. This has been done with C# code embedded in a razor block in my Index view, which changes the inline styles of each of the tiles based on the “PriorityID” foreign key. Just like the employees index, there is a “List” view for unassigned orders index.

Below is the code example to enable this feature.

```
@foreach (var item in Model)
{
    #region color-coding the background back on priority
    var backgroundcolor = "rgba(252, 8, 8, 0.68)";
    if (item.PriorityID == 3)
    {
        backgroundcolor = "rgba(57, 255, 80, 0.7)";
    }
    else if (item.PriorityID == 2)
    {
        backgroundcolor = "rgba(49, 129, 233, 0.4)";
    }
    else
    {
        backgroundcolor = "rgba(252, 8, 8, 0.68)";
    }
    #endregion
    <div class="smallData col-lg-4" style="background-color: @backgroundcolor">
        <div class="smallDataContent">
            @if (User.IsInRole("Owner") || User.IsInRole("Manager"))
            {
                @Html.DisplayNameFor(model => model.TSTOrderPriority.Name) @Html.DisplayFor(modelItem => item.TSTOrderPriority.Name)<br />
            }
            <img src="@Url.Content("~/Content/img/OrderImages/"
                    + item.Picture)" alt="" width="100" height="100" style="border-radius:50%;" /><br />
            @Html.DisplayFor(modelItem => item.TroubleDescription)<br /><br />
            @Html.DisplayNameFor(model => model.StartDate):
            @Html.DisplayFor(modelItem => item.StartDate)<br />
```

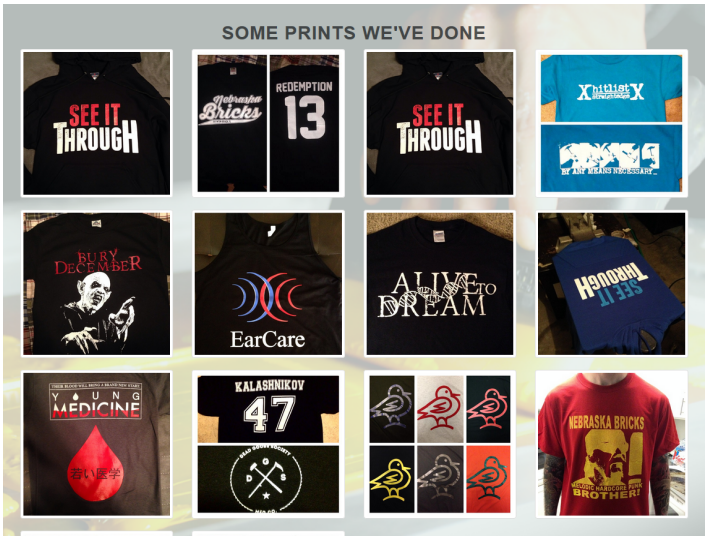
3. The page fades between each page load, unless the User is in the Role of “Owner”. There is a jQuery plugin, I bought in to fade the elements of the page on page load. The “Owner” role has been disabled from seeing the page load. I figured that anyone in the role of “Owner” wouldn’t want to see all that page fade. Below is the line of code in the _Layout.cshtml (shared layout) of the project.

```
<!--If the User is logged in as an admin, then no jQuery fade will occur.-->
@if (User.IsInRole("Owner"))
{


}
else
{
    <script src="~/Content/fadeelementplugin/src/jquery.fadeInAmate.js"></script>
}
```

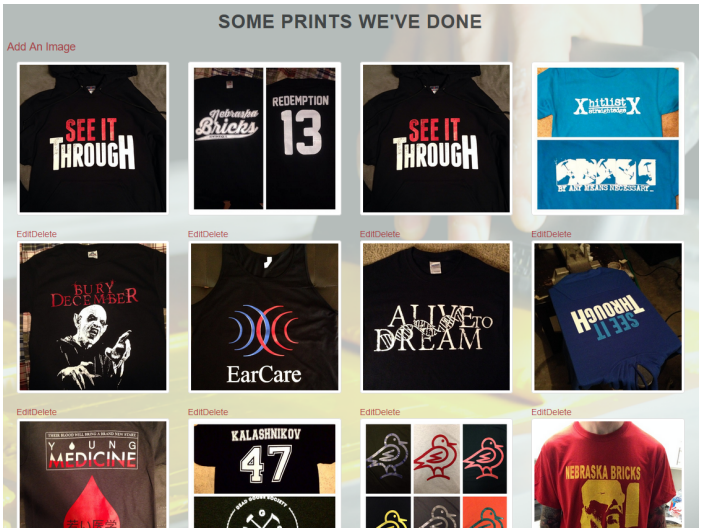
4. The Image on the home-page was designed by a friend several years ago, but the image has been manipulated in Adobe PhotoShop Elements, to disable the white background.

5. Images in the portfolio page are able to be uploaded, updated, or deleted by any user in the role of "manager" or "owner". To the average user, the Portfolio Page, looks like a normal static page with no CRUD functionality.

Viewed without "Owner" or "Manager" roles

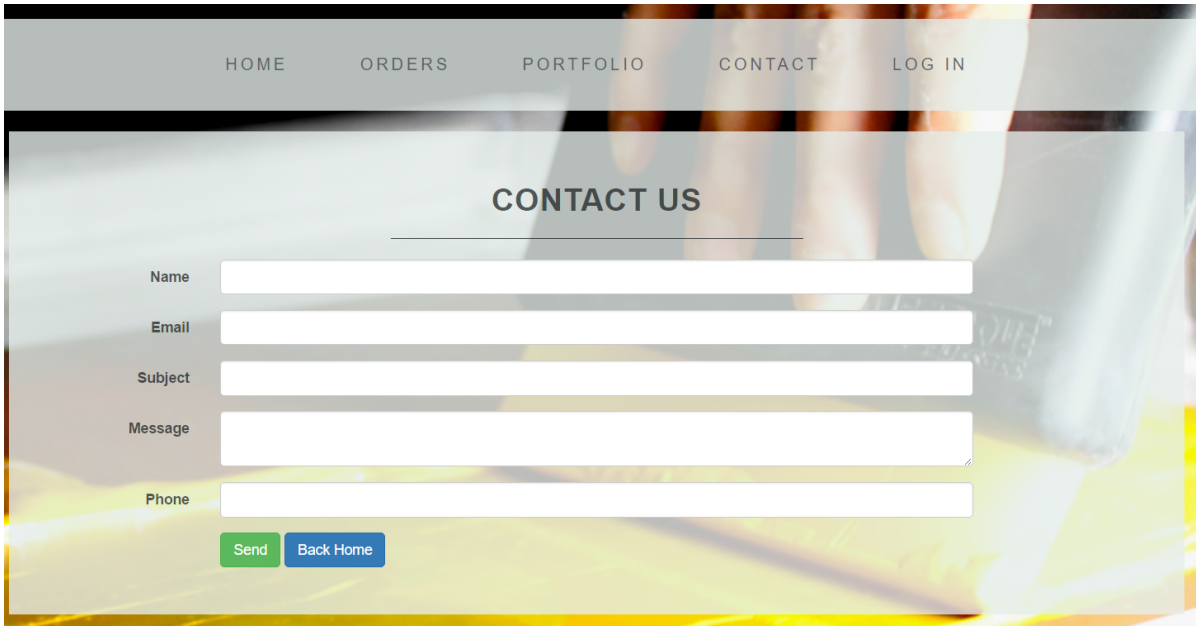


Viewed with "Owner" or "Manager" roles.



6. An extra Action was added to my Orders Controller in order for users in the appropriate roles to be able to “Create”, “Assign/Edit”, “Complete”, and “Archive” the orders. I needed that fourth action in order to make this possible.

7. A contact form that I have programmed in the Contact Action in the Home Controller in order to send the company a message directly to their email. In this case is it send directly to my SmarterASP.NET email since this is a fictional company for project perposes.



Tyler Truitt

10442 Wornall Road, Kansas City, MO 64114 H: 3215378677 ♦ tyleretruitt@gmail.com

http://www.trucoding.com/

Qualifications and Skills

Ability to build static webpages, as well as Model View Controller websites
Ability to perform consistent customer service when necessary
Ability to maintain perfect organization inside of the workplace
Ability to Integrate Data into a Two Tier Project with a Data Layer and a User Interface layer
Excels in C#, HTML5, CSS3, JavaScript, and jQuery
Ability to operate Visual Studio 2015 and Microsoft SQL Server 2012
Ability to follow directions, and work in a team oriented environment
Ability to work with Web Forms
Ability to configure a database in SQL Server Management Studio

Education

Certification of Completion: .Net Application Development, Current
Centriq Training - Leawood, Kansas

- Studying front end, middle tier, and back end programming
- Working with Entity Framework as an Object Relational Mapper
- Studying C# Fundamentals
- Emphasis on .Net Development
- Created a secure ASP.NET MVC application for managing the life-cycle of a trouble ticket associated with hardware and software within an organization. Administrators have the ability to manage employee and department data as well as manage all details of submitted trouble tickets.
- Created a secure two-tier ASP.NET MVC application that displays a Print Shop where the employee, order, and department data is managed with CRUD functionality depending on users and roles.
- Creating a variety of side projects which can be viewed on my personal site above.

High School Diploma: 2009
Holy Trinity Episcopal Academy - Melbourne, Florida

- Top 15% of class
- Recipient of Bright Futures Scholarship
- Graduated with Honors
- Maintained a Grade Point Average of 3.6

Work History

Delivery Driver, 06/2015 to 09/2016
Sarpino's Pizzeria – Kansas City, Missouri

- Operated a Public Operating System in order to route my orders for delivery.
- Greeted customers in a friendly and professional manor, and received an optional gratuity.
- The following process was repeated until there were no orders available. If they were not available, other restaurant tasks would be completed.

Machine Operator, 02/2015 to 04/2015
ACI Services – Blue Springs, Missouri

- Operated small machinery involving the use of a computer program that was used to adjust the size of the natural

gas products.
- Special kits and other parts were assembled for the orders of customers, and shipment was performed.

Sales Associate, 08/2013 to 12/2014
Fast Fix 123 – West Palm Beach, Florida
- Inbound calls were answered and a diagnostic was performed on potential client's computers to address the initial issue was that they were calling about.
- If the customers needed further assistance, remote technical support was sold, which was performed by a Microsoft Certified Technician.

Intake Specialist, 01/2012 to 06/2013
Palm Partner's Treatment Center – Delray Beach, Florida
- An assessment was performed on clients upon their arrival into Palm Partner's Detox facility, taking down their basic information and insurance information.
- Client's files were organized and charted for documentation.
- Supervisor was consulted after all Intake Assessments were completed.

Professional Summary

Background in Customer Service, General Labor as well as a background in the medical field and service industry. Organization was maintained in the workplace as well as a calm and collective mind.

Accomplishments

State Champion Cross Country at Holy Trinity Episcopal Academy years 2006, 2007, and 2009
Captain of the Cross Country team at Holy Trinity Episcopal Academy senior year of High School
Member of Mu Alpha Theta in High School which was a Calculus club that met extra-curricular
Member of the Track and Field team at Holy Trinity Episcopal Academy and has a current record of 2:01 for the 800 meter race
Considered a valuable team member of any job I have had, and have never been terminated by a company