KAMRON PERRY

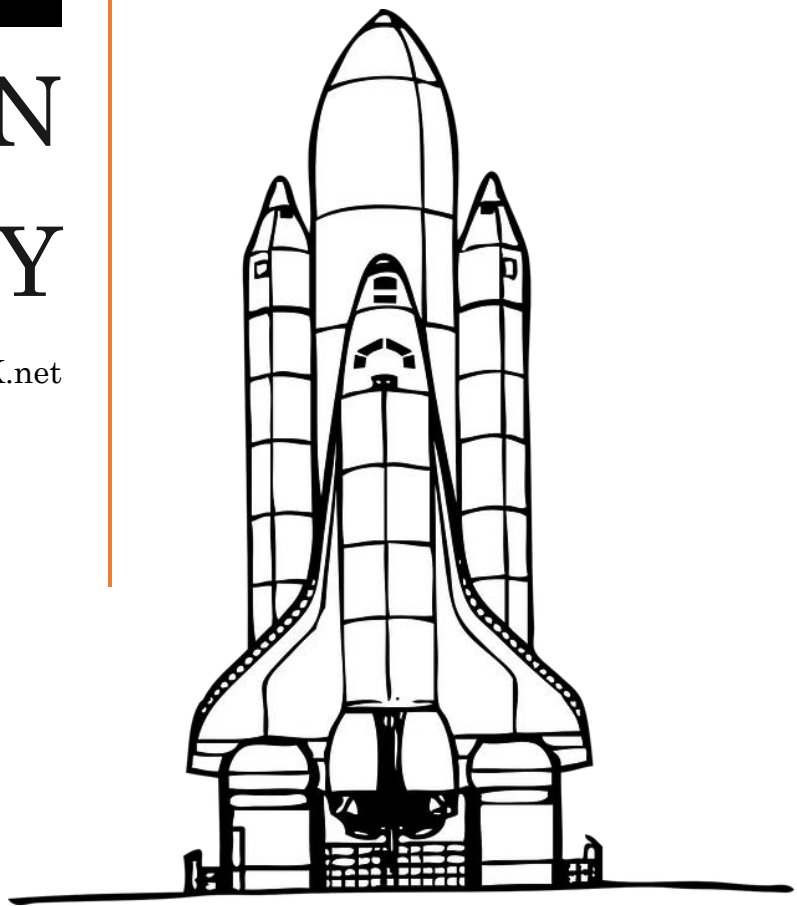
PerryK.net

CENTRIQ - FINAL PROJECT

Chá Tea – Internal Job Board

September 29th, 2017

Kamron Perry
Full-Stack developer



Kamron Perry

I came to Centriq in June 2017 to begin a career as a full-stack developer. The decision came from personal experience with electronic hardware and the desire to learn how it works, and why.

My previous professional experience is in the labor force, these labor jobs gave me experience leading, but most importantly, I learned to work on teams with people from many different backgrounds and personalities.

Most recently I worked moving clients into new homes and businesses as a straight truck driver and trainer with Two Men and A Truck. It was my responsibility to load the client's possessions onto the truck with my crew, secure them properly, drive them to the destination, and unload with my crew. I quickly took on additional responsibilities in the moving company as my calm demeanor and willingness to listen stood out among my co-workers. In addition to my role moving clients I began covering for managers in the office, handling over the phone payments, giving advice to crews running into problems, and entering data for the location's services and revenue that day.

The final push to begin my path building a career in technology came from my desire to grow and better myself. I did not feel I could continue to learn and grow as a manager in the office of a labor job.

Taking that into consideration becoming a developer seemed like a natural fit. There are always new emerging technologies and more to learn to become proficient with the technologies you know. The ability to sit down and create something new is also very satisfying for me.

I am excited to begin my journey as a developer and would love to build my skills and earn the ability to work on impactful projects later in my career.

Project Brief : Internal Job Board

Organizational Background

Mission

Chá Tea aims to provide its customers premium quality beverages.

Programs/Services

Chá Tea provides made to order tea and coffee to its customers at multiple locations in the mid - west United States.

Target Audience

Employees, Managers, Corporate

Strategic Objectives

Chá Tea is beginning to grow and must find a way to share job openings between their locations. The system needs to allow managers to post openings while also allowing employees to create an account, upload a resume, and one - click apply for open positions. Managers should be able to view applications for their location and employees should be able to view the status of their applications.

Admin/Corporate Functionality

Full CRUD functionality for all locations, positions, applications, and notes.

Manager Capabilities

CRUD functionality for positions at their location.

CRUD functionality for applications at their location.

Employee Capabilities

CRUD functionality for their account.

Upload a Resume to their account.

Apply for positions and view if they were declined.

Branding

This is an internal application, so Chá Tea is open to any branding and template options available.

Project Brief : Internal Job Board

Technology Environment :

Current Tech Environment

Website – fsdp.PerryK.net

Back End Systems – A full database build for provided database schema

Integrations – None

Programming Objectives/Challenges

Chá Tea's employee base is comprised largely of Millennials or younger who are very tech savvy and typically interact with online media via smart phones, thus the website must look good and function well on mobile devices.

Users First and Last name should be stores in the AspNetUsers table and able to be accessed through the Identity User object.

Employees should be able to apply for an application with a single click.

ApplicationDate should be populated automatically.

Desired Enhancements

Organization Objective :

Enable publication, application and tracking of available job openings

Create a job openings database

Provide easy to use application process for available openings

Provide easy to read listing of job openings for employees and report of applicants for managers

Target Audience & Users :

Beachside Corporate (Admin)

Managers

Employees

Technical Architecture :

ASP.Net MVC

Login Capabilities

CRUD Functionality for the job opening database

Technologies Used

- MSSQL Server
- C#
- ASP.NET MVC
- ASP.NET Identity
- Visual Studio 2015
- Entity Framework
- jQuery
- JavaScript

Planning Tools

A Trello board was used to organize tasks and track progress.

Specific requirements and the Use Case Diagram were hand written.

"TODO:" comments were periodically added to the project to note additional functionality desired that would be put on backlog while core functionality was polished.

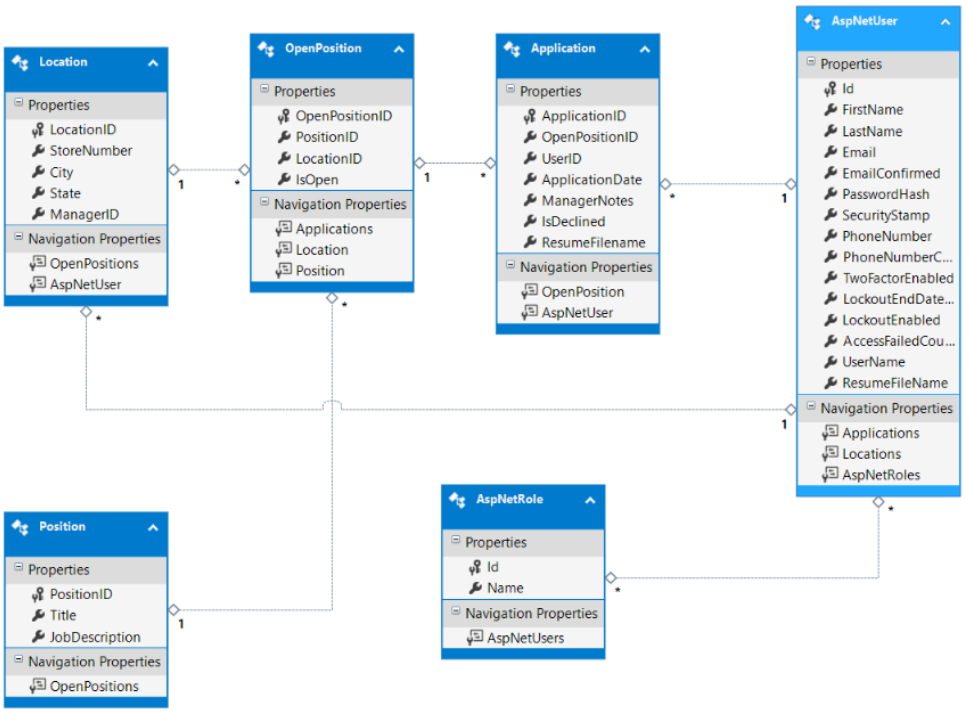
BackLog/Future Features

Give users the ability to upload a new resume that replaces the old one attached to their account without changing the dependency of applications the user has already made.

Allow authorized users to apply to jobs without being employees provided minimal external functionality and allot managers to place new hires into the employee role.

Polish the navigation bar so that it is user friendly on a phone.

Database Schema



Code Snippets

Uploading a new resume – In Progress

```
        #endregion                                  //New model created for this upload   File sent on Post
        public async Task<ActionResult> NewResume(EditResumeModel model, HttpPostedFileBase ResumeFileName){
            string fileName = ""; //Declare string for holding resumeFileName
            if (!ModelState.IsValid){
                return View("Index",model);
            }
            if (ModelState.IsValid){
                var user = await UserManager.FindByEmailAsync(User.Identity.GetUserName()); //Find the user object based on current user
                if (user == null){
                    return View("Index",model);
                }
                if (ResumeFileName != null){
                    string oldFile = user.ResumeFileName; //hold the user's old resumeFileName
                    fileName = ResumeFileName.FileName;  //holds uploaded resume File Name
                    string ext = fileName.Substring(fileName.LastIndexOf('.'));
                    string[] goodext = { ".pdf" };
                    if (goodext.Contains(ext.ToLower())){
                        if (oldFile != null){
                            fileName = oldFile;  //changes the new fileName to the old fileName
                        }
                        else{
                            fileName = Guid.NewGuid() + ext; //If no previous file create a new file name

                        }
                        user.ResumeFileName = fileName; //sets the ResumeFileName data in teh user table to the current fileName
                        ResumeFileName.SaveAs(Server.MapPath("~/Content/Resumes/" + fileName)); //Saves the file - Will overrite existing file (change?)
                    }
                    else{
                        throw new InvalidFileTypeException("Invalid file type uploaded. Only .pdf files are allowed.");
                    }
                }
            }
            return View("Index");
        }
    }
```

```
    public class EditResumeModel
    {
        [Required]
        [Display(Name = "Resume")]
        [DataType(DataType.Upload)]
        public string ResumeFileName { get; set; }
    }
```

```
@{
    ViewBag.Head = "Chá Tea";
    ViewBag.LikeAhiTuna = "Tea Tea";
}
```

Populating Contact Page information with User Information

```

@model FPJobBoard.UI.Models.ContactViewModel
@using FPJobBoard.UI.Models
@using Microsoft.AspNet.Identity.Owin;
@using Microsoft.AspNet.Identity;
@{
    ApplicationUserManager aUserManager = HttpContext.Current.GetOwinContext().GetUserManager<ApplicationUserManager>();
    ApplicationUser user = aUserManager.FindById(User.Identity.GetUserId());
}


    @using (Html.BeginForm())
    {
        if (Request.IsAuthenticated) {
            Model.Name = user.FirstName + " " + user.LastName;
            Model.Email = user.Email;
        }
        <div class="customColumnFlex bg-faded form-item-width">

```

```
        [Authorize]
        public ActionResult Contact()
        {
            ViewBag.Title = "Send Us A Message!";
            ContactViewModel contact = new ContactViewModel();
            return View(contact);
        }
```

This functionality required that I create the object and pass it to the Page before submitting the form. Then I could assign those values modeled object before the form inputs were created so they would populate with the information passed to them.