Full-Stack Developer Project Presentation

October 27, 2017

By: Jeremiah Mata

Internal Job Board

Contact Info:
www.jeremiahmata.com
0jeremiahmata@gmail.com
linkedin.com/in/jeremiah-m-35373a110/

Content :

Introduction to Web Development:

Before I say much about myself I would like to thank you for attending my project presentation! My name is Jeremiah Mata, and I have been immersed in technology since the beginning of high school. I was involved in Combat Robotics which some may also know it as Battlebots a previously televised show.

I brought Battlebots over to Kansas State University as the founder and president of the club. While I was at KSU I was studying Electrical Engineering and I finished my first year of college early this year. Before college I had gained an internship opportunity from Kiewit Engineering Inc., where I learned an immense amount of information. I learned from the engineers there that most of what they had learned in college wasn't going to be applied, a good 80% of what they learned. This later became a key aspect to my transition, but what I really was looking for was a faster track to my end goal. My end goal starting college was to do something with programming. At the time I didn't know much about development but after some research, reputable sources, and recommendations I decided to invest my time at Centriq.

Centriq will always be, to me, my fundamental starting ground towards my career. I have learned a vast amount about programming but I have also learned I am just scratching the surface. My favorite subject that I learned at Centriq was Front-End Development. The reason being is I have a natural knack for designing. I have always enjoyed coding but even more so the free-range aspect of designing from Robots, to Presentations, and now Front-End coding.

Lastly looking towards the future I am interested in learning many other languages some specific examples are Python and Ruby. I also know that coding will always bring challenges that I am excited to overcome. Finally I

aspire to reach that senior developer status and by that point be well invested in a company that provides me with those challenges and has their employees best interests at heart.

PRO1 **2017 Version 1.02.00**

Project Brief: Internal Job Board

Organizational Background:

Mission (Visions & Values)

Beachside Seafood aims to be the premier provider of fresh seafood in the United States.

Programs/Services

Beachside Seafood provides lunch and dinner in fine dining environment with locations throughout the United States.

Target Benefactors

Corporate, Managers, Employees

Strategic Objectives

Beachside Seafood has recently become a publically traded company and thus must find a way to share job openings easily between all their locations across the United States. The system needs to allow managers to post openings while also allowing employees to create an account, upload a resume and simply apply for open positions. Manager should be able to view applications for their location and employees should be able to view the status of their applications.

Admin/Corporate Capabilities:

- *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 Beachside Seafood is open to any branding and template options available*

Project Brief: Internal Job Board

Technology Environment:

Current Tech Environment

- *Website – FSDP.YourDomain.com*
- *Back End Systems – need a full database build for provided database schema*
- *Integrations - None*

Programming Objectives/Challenges

Beachside Seafood'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 names should be stored in the AspNetUsers table and able to be accessed through the Identity User object. (Blog Resource: http://bit.ly/2qXKKZF*)*

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*

Learned from Project:

Flow

- Developing for a week on a project can not be done effectively without flow!
- It is very easy to lose your flow from coding and learning with efficiency to rabbit-hole coding and learning
- Once your fast and on track flow is gone it can be hard to get back on track without an organization method

Organization

- For me always referencing the order of my tasks was key to staying on TRACK and meeting DEADLINES!
- It is easy to lose track of time while coding and before you know it a whole day went by on you researching one part to getting a piece of functionality working
- Knowing the task order at hand and having the pseudocode to complete the task isn't enough, because there are rabbit-holes everywhere in coding especially when it comes to researching so knowing how to use all of my resources effectively became another key learning aspect for me

Using Resources Effectively

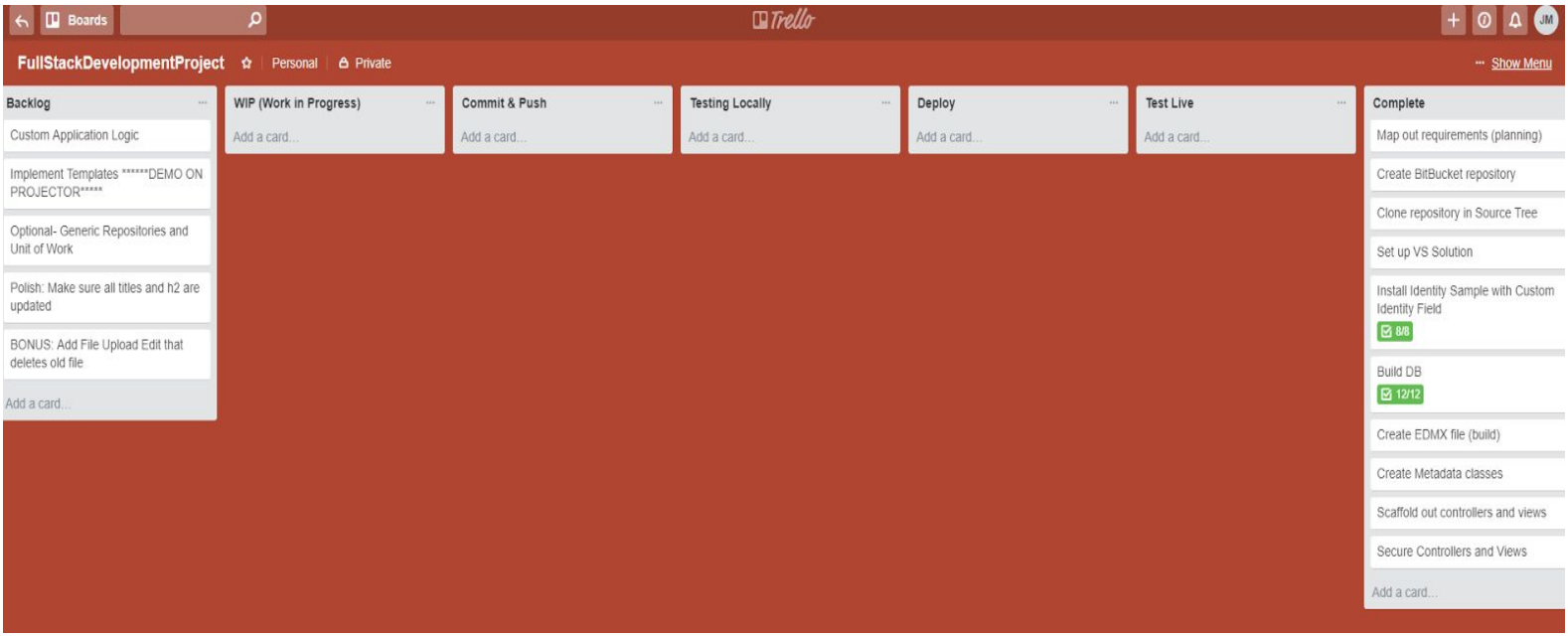
- Coworkers or in this case my classmates were a huge resource to me when asking the right specific questions
- Microsoft, Visual Studio, and Stack Overflow were also a another huge resource, but when using them knowing how to research the answer to your question became detrimental to the success of your research. In most cases knowing the appropriate technical terms that fit the best with the shortest possible way to ask the question helped me the most.

Documentation:

Database Diagram:

Documentation:

Trello Task Organization Board:



Sample Code-One-Click Apply:

```
[OverrideAuthorization]
[Authorize(Roles = "Employee,Corporate,Manager")]
public ActionResult CreateApplication(int? id)
{
    var currentUserID = User.Identity.GetUserId();
    var userManager = System.Web.HttpContext.Current.GetOwinContext().GetUserManager<ApplicationUserManager>();
    //var appInfo = db.Applications.Include(a => a.AspNetUser);
    //application data
    var user = userManager.FindById(currentUserID);
    //var userLocationApplications = db.Applications.Where(x => x.OpenPosition.Location.ManagerID == currentUserID).Include(b => b.OpenPosition.Location);
    //check applications table for records where userid=currentuser and openpositionid = the parameter passed above. count over 1.
    bool alreadyApplied = db.Applications.Where(a => a.UserID == currentUserID).Where(a => a.OpenPositionID == id.Value).Count() > 0 ? true : false;
    if (alreadyApplied == true)
    {
        return RedirectToAction("Index", "Applications", new { Message = ApplicationsController.ApplicationsMessageId.AlreadyApplied });
    }
    else
    {
        Application oneClick = new Application();
        oneClick.OpenPositionID = id.Value;
        oneClick.ApplicationDate = DateTime.Now;
        oneClick.IsDeclined = false;
        oneClick.UserID = currentUserID;
        oneClick.ManagerNotes = null;
        oneClick.ResumeFileName = user.ResumeFileName;


        db.Applications.Add(oneClick);
        db.SaveChanges();
        return RedirectToAction("Index", "Applications", new { Message = ApplicationsController.ApplicationsMessageId.ApplicationSuccess });
```

Documentation:

Sample Code- Find, Delete, and Replace Old File:

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult changeResumeFile(RegisterViewModel model, HttpPostedFileBase ResumeFileName)
{
    var currentUserID = User.Identity.GetUserId();
    var userManager = System.Web.HttpContext.Current.GetOwinContext().GetUserManager<ApplicationUserManager>();
    var user = userManager.FindById(currentUserID);

    string deletePath = (Server.MapPath("~/Content/Resumes/RegisterResumes/" + user.ResumeFileName));
    string oldResumeName = user.ResumeFileName;
    string resumeName = "Nothing.pdf";
    string oldExt= oldResumeName.Substring(oldResumeName.LastIndexOf('.'));
    if (user.ResumeFileName!=null)
    {
       resumeName = ResumeFileName.FileName;
        string ext = resumeName.Substring(resumeName.LastIndexOf('.'));
        string[] goodExts =
            {".docx",".pdf"};
        if (goodExts.Contains(ext.ToLower()))
        {
            if (ext == oldExt)
            {
                ResumeFileName.SaveAs(Server.MapPath("~/Content/Resumes/RegisterResumes/" + resumeName));
            }
            else
            {
                FileInfo file = new FileInfo(deletePath);
                if(file.Exists)
                {
                    file.Delete();
                }
                resumeName = Guid.NewGuid() + ext;
                ResumeFileName.SaveAs(Server.MapPath("~/Content/Resumes/RegisterResumes/" + resumeName));
            }
        }
        else
        {
            ViewBag.Message = "ERROR, .pdf and .docx are the only accepted document types";
            return View();

        }
    }
    user.ResumeFileName = resumeName;
    userManager.Update(user);
    return RedirectToAction("Index",new { Message = ManageMessageId.ChangeResumeSuccess });
```

Documentation:

Sample Code- Notifications:

```
[OverrideAuthorization]
[Authorize(Roles ="Employee,Corporate,Manager")]
public ActionResult Index(ApplicationsMessageId? message)
{
        ViewBag.ApplyMessage =
           message == ApplicationsMessageId.ApplicationSuccess ? "We have recieved your application!"
           : message == ApplicationsMessageId.Error ? "There was an error"
           : null;
    ViewBag.AlreadyAppliedMessage =
        message == ApplicationsMessageId.AlreadyApplied ? "You have already applied for this position!"
        : null;

    var currentUserId = User.Identity.GetUserId();
    var applications = db.Applications.Include(a => a.OpenPosition);
    //var applicationsUser = db.Applications.Include(a=>a.);
    if (Request.IsAuthenticated && User.IsInRole("Corporate"))
    {
        return View(applications.ToList());
    }
    else if(Request.IsAuthenticated && User.IsInRole("Manager"))
        {
        var userLocationApplications = db.Applications.Where(x=>x.OpenPosition.Location.ManagerID==currentUserId).Include(b => b.OpenPosition.Location);
        return View(userLocationApplications.ToList());
    }
     else
    {
        var userApplications = db.Applications.Where(x => x.UserID == currentUserId).Include(b => b.OpenPosition);
        return View(userApplications.ToList());
       }
}
public enum ApplicationsMessageId
{
    ApplicationSuccess,
    AlreadyApplied,
    Error
}
```

Core Struggles:

One-Click Apply- Figuring out the pseudocode wasn't extremely difficult actually it was figuring out how to store my input into my new application and then figuring out how to find the openPosition Id value for the job they were applying for.

Find, Delete, and Replace- This took a lot more time then I allotted for and in the end to not really work for this project specifically because if I was to delete an old resume that they had already used to apply for an application it would leave that slot orphaned in the application generating errors.

Time- Originally I had a pretty good idea of the obstacles I was going to face throughout the project and good estimations on the time it would take me to complete those tasks. It

turned out that it took me less time for the tasks I thought would take me the longest and the problems that took me the most time were ones I didn't anticipate existing

Documentation:

3 Tier Solution with Model View Controller Design and Identity: