# John Swift

# Scientist Learning Management System

# An internal Continuing Education tracking system

February 4th, 2021

About me

I found the process of developing this final project was a microcosm of the emotions of coding- the relaxing time when everything is working early, the lows of not knowing how to start a piece of functionality, the highs of success, and the panic of something breaking without the knowledge of why it is breaking. Luckily, the final emotion was one of contentment that only comes with issues resolved and development complete.

I chose to enter the technology field because my previous jobs were in fields that were contracting. My sister had success being a self-taught developer through LaunchCode, and I decided to attempt to follow suit. I became hooked at the end of my first week, as soon as I got to javaScript, and the logic elements that you find in many languages.

Since working on the basic branching logic that first week, I have always enjoyed the problem solving aspect of coding. Whether its sticking with javaScript, or using C# here at Centriq, writing the logic has been my favorite part of development. That is why I chose the Learning Management System for a final project- our instructor had told us that it used more logic then the others.

Going forward I will be looking to advance my logic-based skills. They were sufficient for the program, but recently I have found myself going to Hackerrank and trying to advance my skills there as well (I am telling myself that spending Saturday nights doing hacker rank is a product of the current pandemic, and not the shape of my future Saturday nights).

Project Requirements

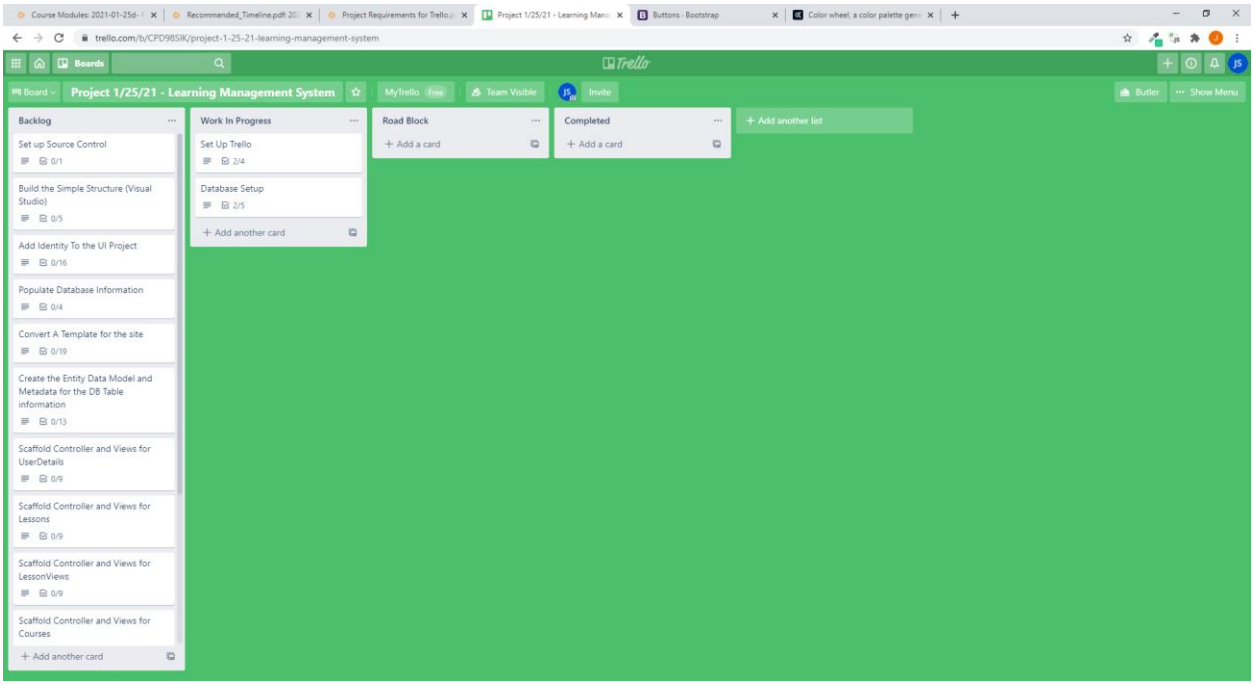
The project was a Learning Management System that kept track of Continuing Education completion across three different levels of employee- administrators, managers, and employees.

The employees would have the ability to go into different courses that they needed to complete, see how many lessons were in each course, and complete those courses (either through watching a video or viewing a pdf). They would also need to see what courses and lessons they have completed, and how many courses they had left to complete. Once a course was completed, an email would be auto-generated and sent to the employee's manager.

Managers would have access to employees progress and would receive the auto-generated emails when an employee completed the program.

Administrators (admins from now on) would have complete CRUD functionality. The admins would be able to see employees progress, create new employees, lessons, and courses, deactivate current courses, and view the two types of objects that were created upon an employee progressing through their CE: the lessonView and the courseCompletion.

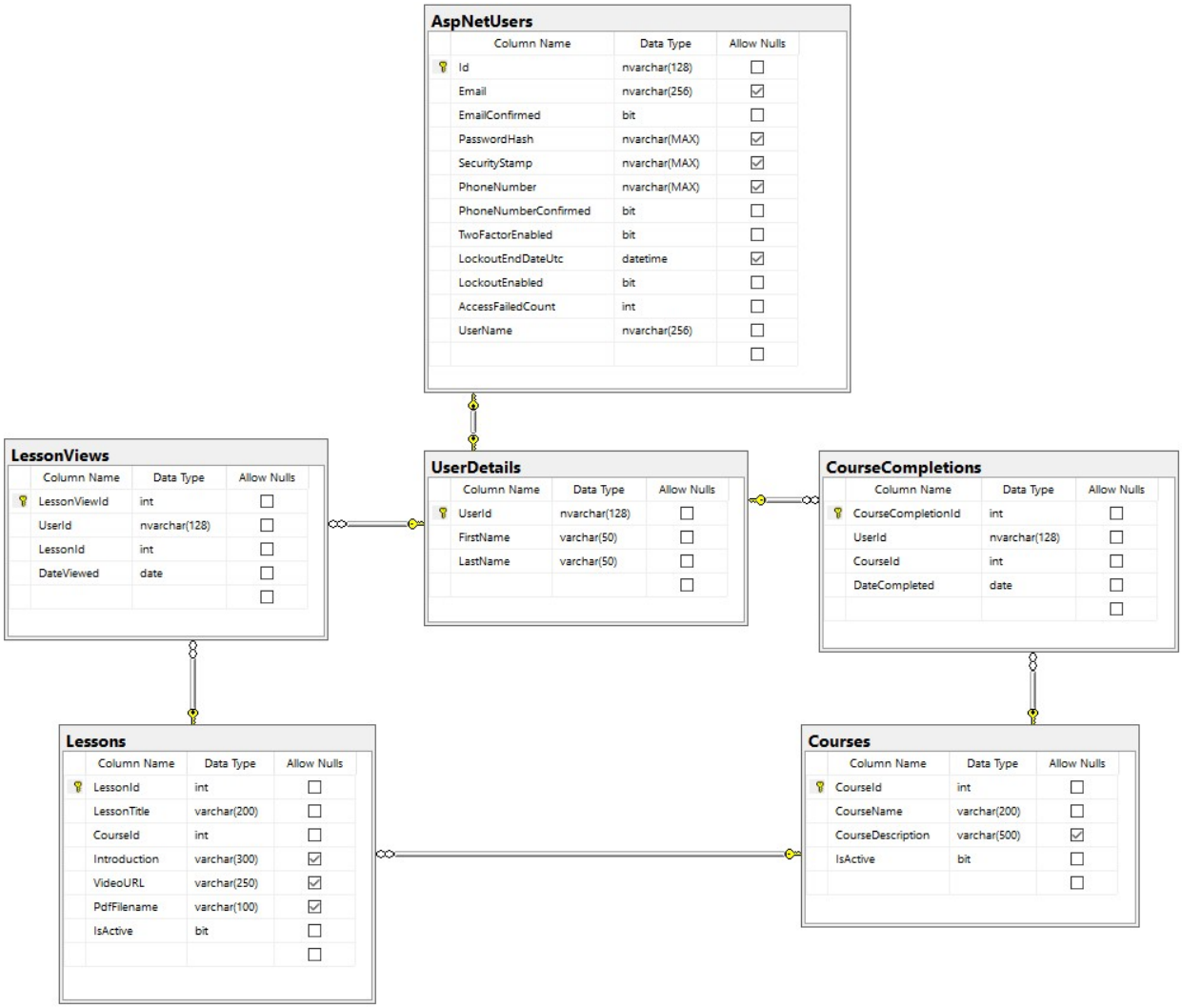
The first step in the was organization- this was done by creating a trello board:



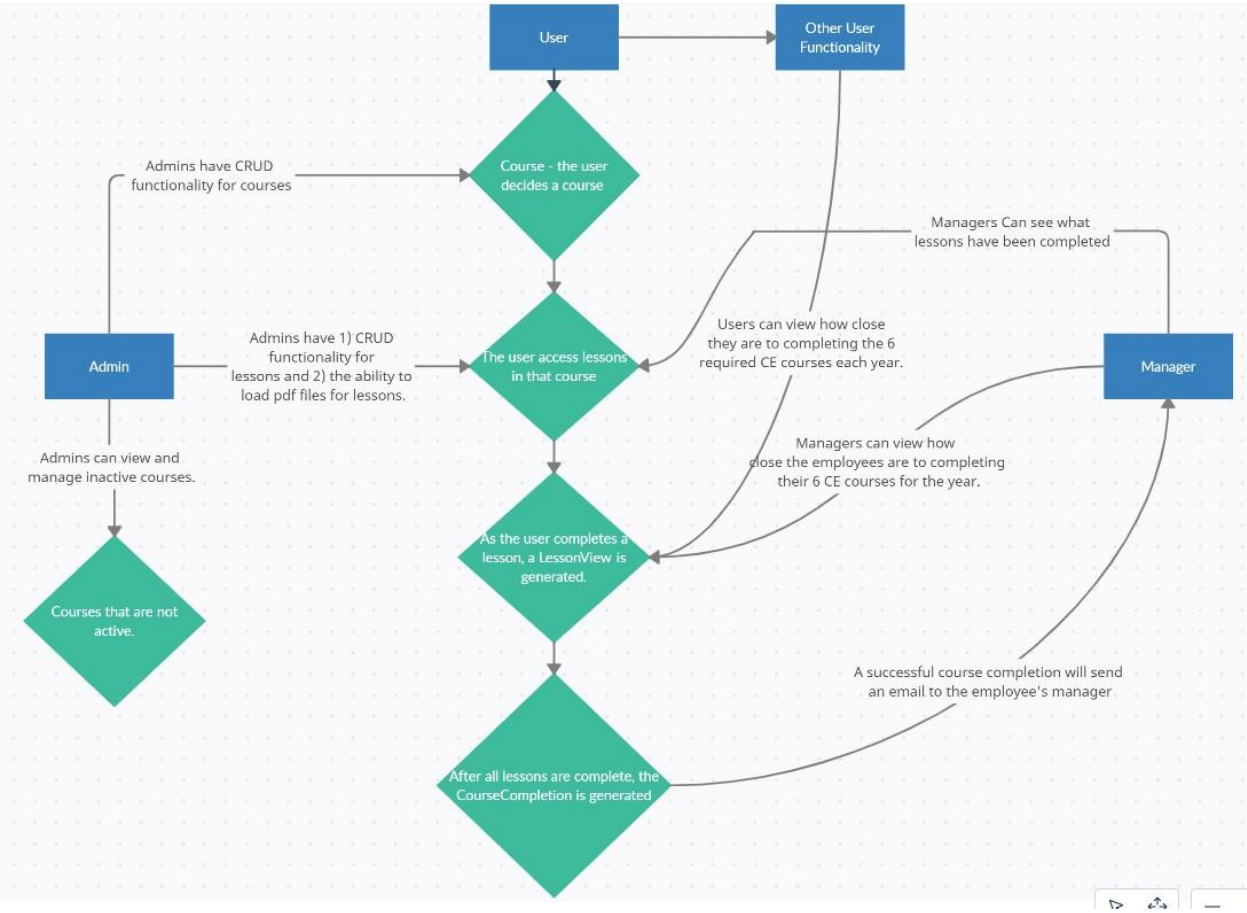

Overall workflow progress went well, having access to checklist functionality, the work in progress, and roadblock segments helped me stay organized as time went on, and reminded me of returning to development segments I had split up. For example, I wanted to get one controller/views scaffolded out with css complete so that I did not have to worry about it later. When I went back to do the rest later, I had already had a head start on the basics and just had to implement them everywhere.

Later in development, another column was added, titled 2.Awesome. It was feature that were to be completed after minimum viable product was completed.

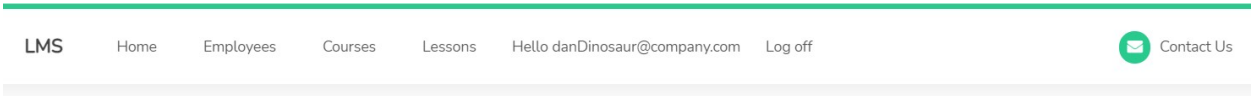
ER Diagram


AspNetUsers
Column Name Data Type Allow Nulls
Id nvarchar(128)
Email nvarchar(256)
EmailConfirmed bit
PasswordHash nvarchar(MAX)
SecurityStamp nvarchar(MAX)
PhoneNumber nvarchar(MAX)
PhoneNumberConfirmed bit
TwoFactorEnabled bit
LockoutEndDateUtc datetime
LockoutEnabled bit
AccessFailedCount int
UserName nvarchar(256)
LessonViews
Column Name Data Type Allow Nulls
LessonViewId int
UserId nvarchar(128)
LessonId int
DateViewed date
UserDetails
Column Name Data Type Allow Nulls
UserId nvarchar(128)
FirstName varchar(50)
LastName varchar(50)
CourseCompletions
Column Name Data Type Allow Nulls
CourseCompletionId int
UserId nvarchar(128)
CourseId int
DateCompleted date
Lessons
Column Name Data Type Allow Nulls
LessonId int
LessonTitle varchar(200)
CourseId int
Introduction varchar(300)
VideoURL varchar(250)
PdfFilename varchar(100)
IsActive bit
Courses
Column Name Data Type Allow Nulls
CourseId int
CourseName varchar(200)
CourseDescription varchar(500)
IsActive bit

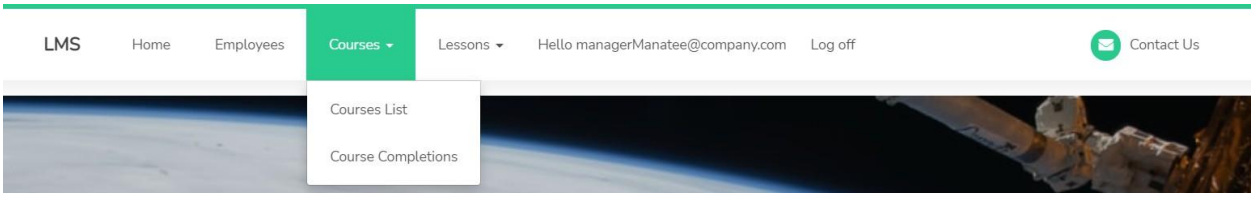
A diagram of functionality for the site:



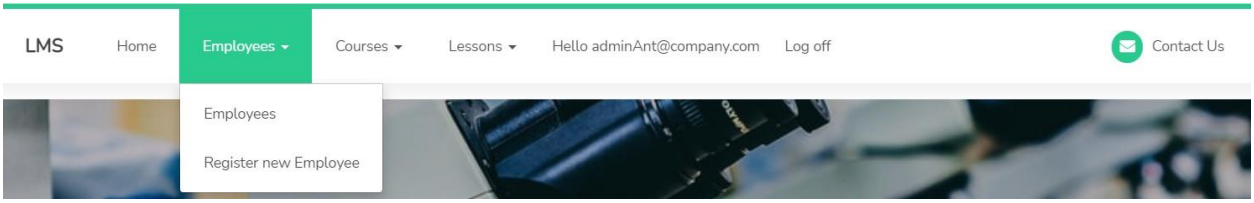
The Navigation bar was going to have different options based on whether the person logged in was as an employee:




A Manager (added dropdowns for courses, to see course list and completions, and for lessons, to see lesson list and lessonViews):

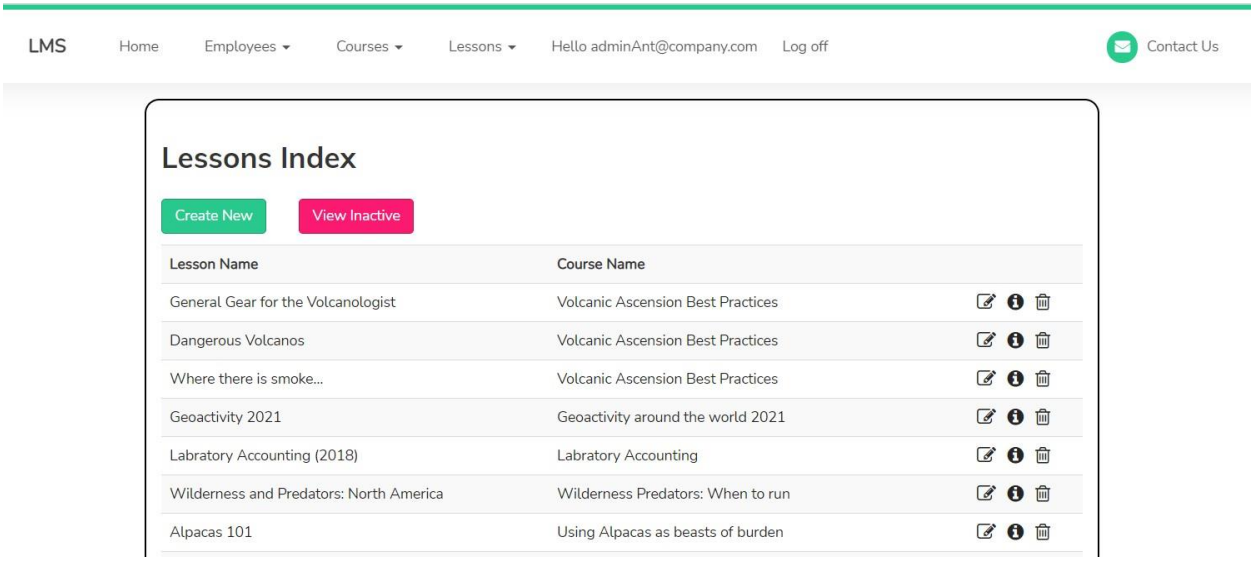



Or an admin (added dropdown for Employees so admins can see an index of all employees or register new ones):

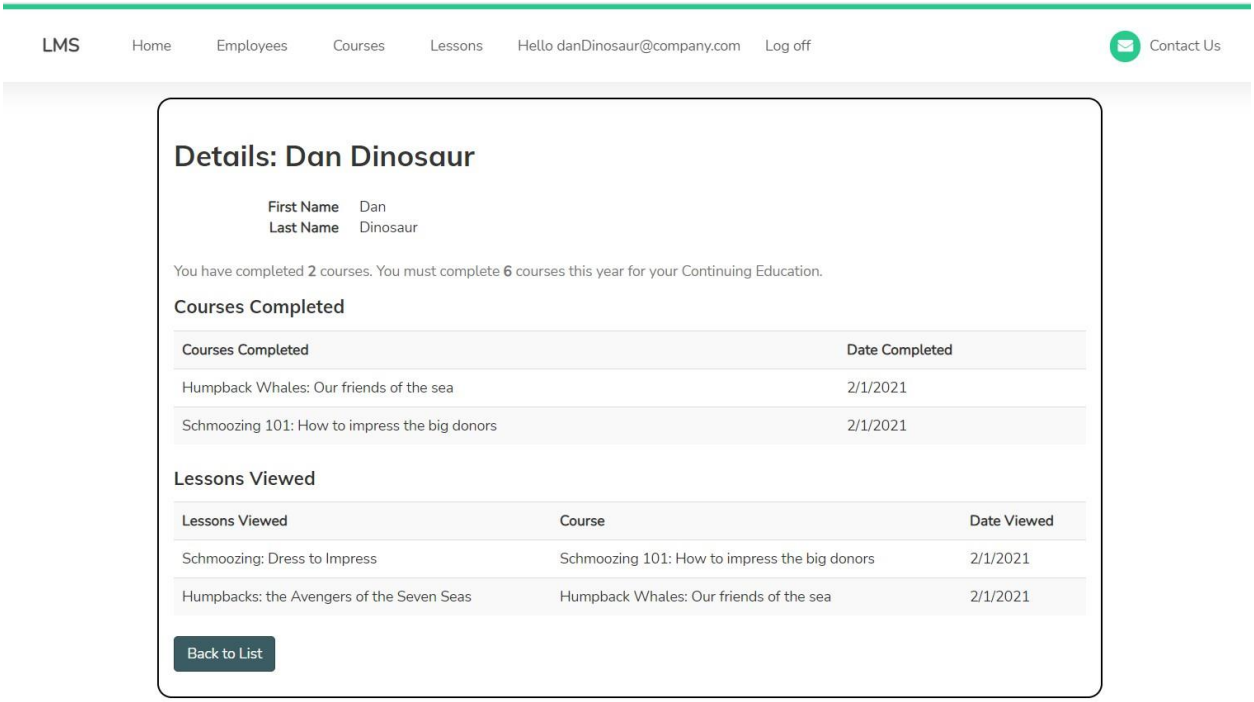



Some UI examples-

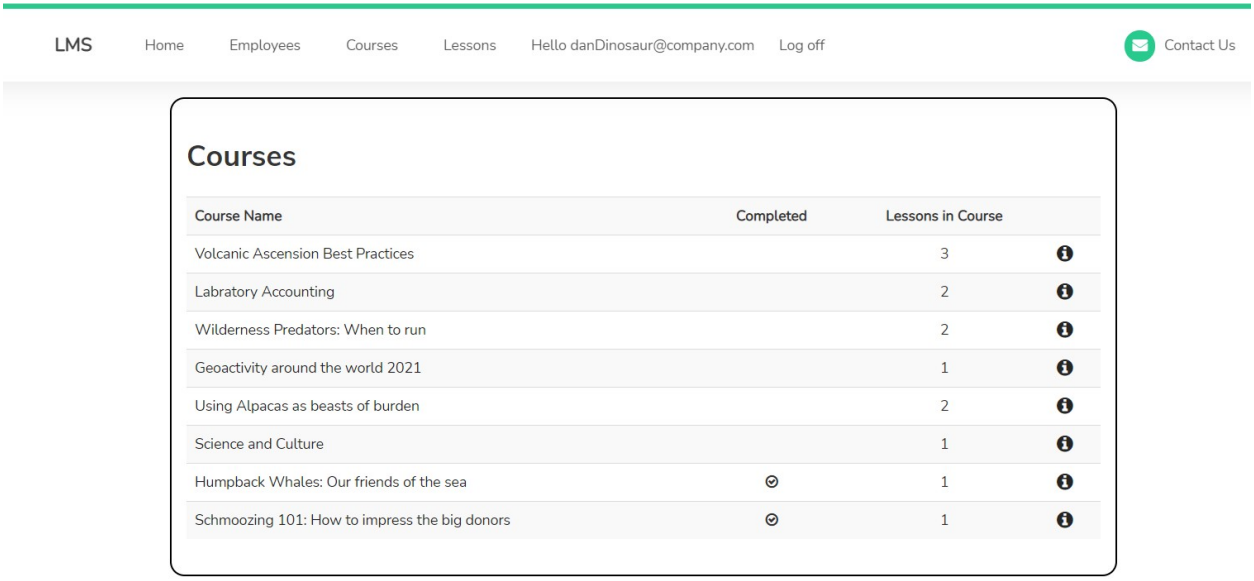
First, an example of the admin view on lessons. You can see the create and view inactive buttons, which are not available for other role types



Second, you can see two views the employee sees has where they discern how many more courses they need. First, the employee, or user, details:




And what they see when they go to the courses page:



Some of the unique feature of this project where that when an employee completes a lesson, a new lessonView is automatically generated. The code below identifies if the currently logged on individual is in the role of “User”, create a new lessonView object, check to see if there is already a lessonView with the lesson Id and the User Id, and then save that new lessonView object to the database if it doesn’t already exist.

```
//string user =
string user = User.Identity.Name;
string theUsersID = User.Identity.GetUserId();
//check is user is in the role of employee
if (User.IsInRole("User"))
{
    LessonView newLV = new LessonView();
    bool myLoop = false;
    int newId = id.Value;
    foreach (LessonView item in db.LessonViews)
    {
        if (item.LessonId == newId && item.UserId == theUsersID)
        {
            myLoop = true;
        }
    }
    //create the lessionview
    if (myLoop == false)
    {
        newLV.LessonId = newId;
        newLV.UserId = theUsersID;
        newLV.DateViewed = DateTime.Now;
        db.LessonViews.Add(newLV);
        db.SaveChanges();
    }
}
```

An issue I had arose when trying to go to the next step in the process- check if all lessons had been viewed in a course for an employee, and if they had, create a new courseCompletion object. Initially, I had two loops that counted the number of lessons in a course and then the lessonViews for a course. After some testing, I discovered that this was going to be bad functionality, as the database would only save the courseCompletion object if those two numbers were equal. This would mean that the number of employees that could generate a course completion was limited, and inaccurate. Both loops (shown below for posterity) were replaced with single line LINQ statements that have the code act like it is supposed to.

```
//loop through lessons getting number of total lessons in a course
//int totalLessonsInThisCourse = 0;
//foreach (Lesson item in db.Lessons)
//{
//    if (item.CourseId == courseParentId)
//    {
//        totalLessonsInThisCourse++;
//    }
//}

int totalLessonsInThisCourse = db.Lessons.Where(x => x.CourseId == courseParentId).Count();

//loop through lessonViews getting total number of lessons the user has completed
//int totalLessonViewsTheUserHasFinished = 0;
//foreach (LessonView item in db.LessonViews)
//{
//    if (item.Lesson.CourseId == courseParentId)
//    {
//        totalLessonViewsTheUserHasFinished++;
//    }
//}

int totalLessonViewsTheUserHasFinished = db.LessonViews.Where(x => x.Lesson.CourseId == courseParentId).Where(y =>
  y.UserId == theUsersID).Count();
```

```
    int totalLessonViewsTheUserHasFinished = db.LessonViews.Where(x => x.Lesson.CourseId == courseParentId).Where(y =>
      y.UserId == theUsersID).Count();

    //This will save it if they have
    if (myCourseLoop == false && totalLessonViewsTheUserHasFinished == totalLessonsInThisCourse)
    {
        cc.CourseId = courseParentId;
        cc.UserId = theUsersID;
        cc.DateCompleted = DateTime.Now;
        db.CourseCompletions.Add(cc);
        db.SaveChanges();

        send an email upon course completion
    }
}
```

# John Swift

FULL-STACK DEVELOPER

913-488-9388
jdavidswift@gmail.com
johndavidswift.com


## Qualifications

- Solid foundational knowledge of designing and developing full-stack web applications using .NET framework.
- Excellent communication, whether with co-workers or clients and an aptitude for problem solving.

## Technical Skills

| | |
|---|---|
| **Front End:** | HTML5, JavaScript, jQuery, jQueryUI, CSS3, Responsive/Mobile Web Development, Bootstrap, ReactJS, D3 |
| **Middle Tier:** | Visual Studio, C#.NET, ASP.NET, LINQ, MVC, EF |
| **Back End:** | ADO.NET, SQL, SQL Server, SSMSE, Node.js |

## Independent Development Projects

- **Personal Site**: www.johndavidswift.com
- **StoreFront**: Created a secure application for managing product data. Application is built to simulate an online store front with a shopping cart. Administrators have the ability to manage product, category and vendor data.
- **S.A.T. Scheduling Administration Tool:** Created a secure application for managing product data. Application is built to simulate an online class scheduling system. Administrators will have the ability to manage students, courses, scheduled classes, and enrollments.
- **Final Project**: Created a secure data-driven ASP.NET MVC application from design through deployment for managing the tracking and organization of hardware and software within a company. Administrators have the ability to manage employee, department data and all details relating to assigned hardware and software.

## Technical Training and Education

**CENTRIQ TRAINING | KANSAS CITY, MO** **OCTOBER 2020 – FEBRUARY 2021**

FULL-STACK WEB DEVELOPER PROGRAM

***Core Competencies:***

- MVC Framework
- Trouble Shooting & Debugging
- Source Control
- Agile/Scrum (Created Team Project)
- Website Deployment
- Pair Programming
- Code Review
- Professionalism, Teamwork, Problem Solving & Effective Communication

**UNIVERSITY OF KANSAS STATE | MANHATTAN, KS** **2006 – 2010**

SPECIALIZED IN BIOLOGICAL AGRICULTURAL AND CIVIL ENGINEERING.

## Professional Experience

**AIG | OLATHE, KS** **MAY 2015 – OCTOBER 2016**

ADJUSTER LICENSING ANALYST

- Handled applications, renewals, and continuing education for 2,000 adjusters.
- Lead companywide training program courses by reviewing the basics of insurance licensing.
- Primary contact for the Departments of Insurance for four states.

**JOHNSON COUNTY PARK AND RECREATION DISTRICT, JOHNSON COUNTY, KS** **MARCH 2012 – MAY 2015**

SCHOOL-AGE SUBSTITUTE

- Directed after-school programs for school-age children; managed the hiring of staff; communicated with upset customers and was able to retain their business.
- Planned activities, communicated with SRS, handled fee payments, and was knowledgeable with staff, children, and parents at over 20 sites.