Fall 2021

eLearning

Christopher Burch

eLearning

By

Christopher Burch
B.A., Carthage College 2015

GRADUATE CAPSTONE SEMINAR PROJECT

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,

With a Major in Computer Science

Governors State University
University Park, IL 60484

2021

ABSTRACT

The ongoing COVID-19 pandemic had a drastic impact on the education industry. With the sudden digital transformation, the education sector invested in eLearning, which surged to a $250 billion industry with incredible growth. Our goal is to create an eLearning website which provides the common functionality seen in the marketplace.

From a feature perspective, the website will have user logins and roles which align with the industry such as students, instructors, and administrator accounts. The available features are spilt out by roles. Instructors will be able to add their course materials to the site, interact with students via a discussion board, and view their progress as the course carries on. Students will be able to create an account, enroll in a course, and work through the course materials provided by the instructors. The administrator account will be able to manage the other accounts and existing courses. Furthermore, the site will have a course landing page where a student can search for courses by keywords, explore featured courses, view their enrolled courses, and sign up for notifications and alerts related to their courses.

From a technical perspective, the overall architecture will be a microservice architecture. It will feature a frontend, an API-driven backend, and a database for storage. The frontend will be composed of HTML and CSS for views, JavaScript for driving frontend functionally, and bootstrap for tying those three pieces together into a responsive web application. The backend will be based on .NET Core and written in C#. It will be an API-based server-side application which handles authentication and authorization, routes traffic, serves the frontend pages, and sends calls to the database for data operations. The database will be a Microsoft SQL Server instance which stores the application information in a relational database. In terms of deployment and infrastructure, it will be written as a cloud-native application using Linux docker containers to promote a modern, scalable cloud pattern. Simple version control will be implemented via a private Gitlab repo.

Table of Content

1 Project Description

eLearning is a website designed to allow users to learn a variety of subjects at their own pace. Courses are uploaded and maintained by certain users called *instructors*, who are responsible for the content of the lessons. This content may include any number of lessons, each of which may have a PDF file, a video file, and/or a YouTube video.

Once a user registers for the site, they may register for any number of courses. Users may immediately view all course lessons, including all associated media, and may progress at whatever pace they are comfortable with. Users may access previously completed material at any time.

Instructors are also considered users, and therefore may register to take any number of courses, as well as creating their own.

In addition to regular users and instructors, the site also supports administrative users, or admins. These users will be responsible for tasks related to the operation of the site itself. Currently, this is limited to changing a user's status as an instructor and/or an admin. As with instructors, admins are considered to be users, and therefore may register for courses.

1.1 Competitive Information

There are many "self-learning" websites, the three websites that we have as our competitors are Udemy, Pluralsight and Coursera. These websites are world renowned and have helped many students learn new skills.

Coursera is a subscription-based site; with three different subscription options; single learning program $49 – $79 /month, Coursera Plus Monthly $59 /month, and Coursera Plus Annual $399 /year. Pluralsight is a paid website that has 1 million individual users. Udemy has boasts of over 50 million students (about twice the population of Texas) and has courses beyond technical courses with regular "flash sales". Unlike Coursera and Pluralsight, once a student pays for a class, they have access to the class forever.

1.2 Relationship to Other Applications/Projects

Our eLearning project does not have any direct affiliations to existing projects, but it does use design principles taken from existing eLearning websites. Namely, it builds off the free structure provided by Khan Academy where instructors can upload materials and users can consume them at their own pace. The simple-to-follow UI design was inspired by Coursera and Udacity. And finally, the warm color scheme was inspired by the Governors State University website's design.

1.3 Assumptions and Dependencies

The eLearning website requires several infrastructure technologies. It requires Gitlab for storing code, creating containerized builds, storing those containerized builds in a container repository, and deploying those containers to a production environment. Each of these steps are critical to the application's development and deployment.

In addition to Gitlab, it also requires several cloud-hosted technologies. An FTP server must be created which can store course materials and discussion boards. Ideally, it would be stored in Google Cloud Platform (GCP) which provides a simplified process for creating an FTP server instance which can then connect to the application's backend. Other than the FTP server, the application also requires a SQL

Server instance. The SQL Server instance is used by the backend to store data related to users, courses, and lessons, and therefore should be accessible by the backend but secured from non-authorized users. The SQL Server should also be cloud-hosted, so hosting it in GCP is also the preferred method.

Finally, in addition to the other cloud-hosted instances, the eLearning application's backend should also be hosted in a GCP instance. Gitlab handles the deployment to a GCP instance so that simplifies an otherwise complicated cloud deployment. As a result, we must simply create a Google Kubernetes Engine cluster which can then be linked to Gitlab.

1.4 Future Enhancements

Although we are starting our website as a free website, we are planning to monetize our product. We will eventually start charging for a subscription and add ads to any unpaid features.

Eventually we plan to add certification options for certain learning plans. Beyond the single certification options, we would like to partner with smaller universities such as Governors State University to better prepare our students for job roles in the IT field.

We are also planning to allow users and instructors to link the E-learning profiles to their social media accounts. This will allow individual users to announce achievements and classes that they are enrolled in, to friends on their pages/ This can also provide our website with free and unsolicited advertising.

1.5 Definitions and Acronyms

API – Application Programming Interface

AWS – Amazon Web Service

FTP – File Transfer Protocol

GCP – Google Cloud Platform

2 Project Technical Description

The project is primarily composed of cloud-ready containerized components. The frontend is a react application which is served to the user by the backend's webserver. The frontend should follow basic react principles regarding components and reusable code. The backend should be a cloud-native ASP.NET Core application which is capable of serving and interacting with the frontend, in addition to the SQL Server instance and FTP server.

The SQL Server instance should also be a cloud-native instance capable of being deployed and hosted by any cloud provider. This simple interface provides the backend design with a clear and concise contract which helps provide stability while preventing security threats. Similarly, the FTP server should follow the same principles. It should also be a cloud-native solution which can easily be hosted in a cloud environment. The backend should be able to store and retrieve files, while preventing unauthorized users from accessing the file store. In addition to the FTP server-based materials, instructors should also be able

to upload links to YouTube-hosted videos. As a result, the application must be capable of presenting the user with embedded YouTube videos.

Each of the three user groups has a variety of workflows. Administrators should be able to change existing user privileges by giving them instructor and admin roles. Instructors should be able to create courses, add lessons to those courses, and finally add materials to those courses in the form of PDF's, mp4 files, and YouTube-hosted video links. Users should be able to sign up for courses added by instructors, view the course lessons and their materials, post to a course discussion board, and change their user profile details. Finally, non-users should be able to sign up to the site so they can become a user.

Although the eLearning website provides minimal security risk since it does not take any form of payment information from its users, it does contain a limited amount of user data – mainly their email addresses and possibly their names. As a result, the website's data is secured using cloud-hosted SQL Server instances which prevent unauthorized access to the data, and the backend's API methods prevent unauthorized users from requesting restricted data.

2.1 Application Architecture

The application architecture is primarily designed around 2 concepts – cloud-native design and the single responsibility principle. In particular, cloud-native design focuses on containerization, which is a separation of modules – frontend, backend, database, file store, infrastructure, etc., are functionally decoupled. Instead, they interact through well-defined contracts. The single responsibility principle states that each class should be responsible for a single functionality. As a result, classes are more concise and other classes interact with them based on more strict contract decisions. These architecture choices allow the team to break down concepts and features, work on them separately, and combine them together after the fact to have a functional product. As a result, the team is capable of working on nearly every feature in any order. This system allows the team to work on features in parallel.

Whereas the cloud-native design and single responsibility principle describe the eLearning website from a macro-architecture perspective, the details of each module are slightly more complex. And overview can be seen in Figure 1. The frontend is a react application which is served to the user from the backend. React applications are self-contained web applications which allow for fast navigation and concise use of data objects. React can leverage the type of fluidity of JavaScript to rapidly receive data from the backend APIs and dynamically display it in its components. This allows developers to design the structure of a page based on an assumed data type. These assumptions give the frontend developer freedom over the design and decouples it from the backend's models. As a result, frontend developers can write pages based on the model design, independently of backend developers, which allows rapid development practices. For this reason, we chose React as the frontend technology.

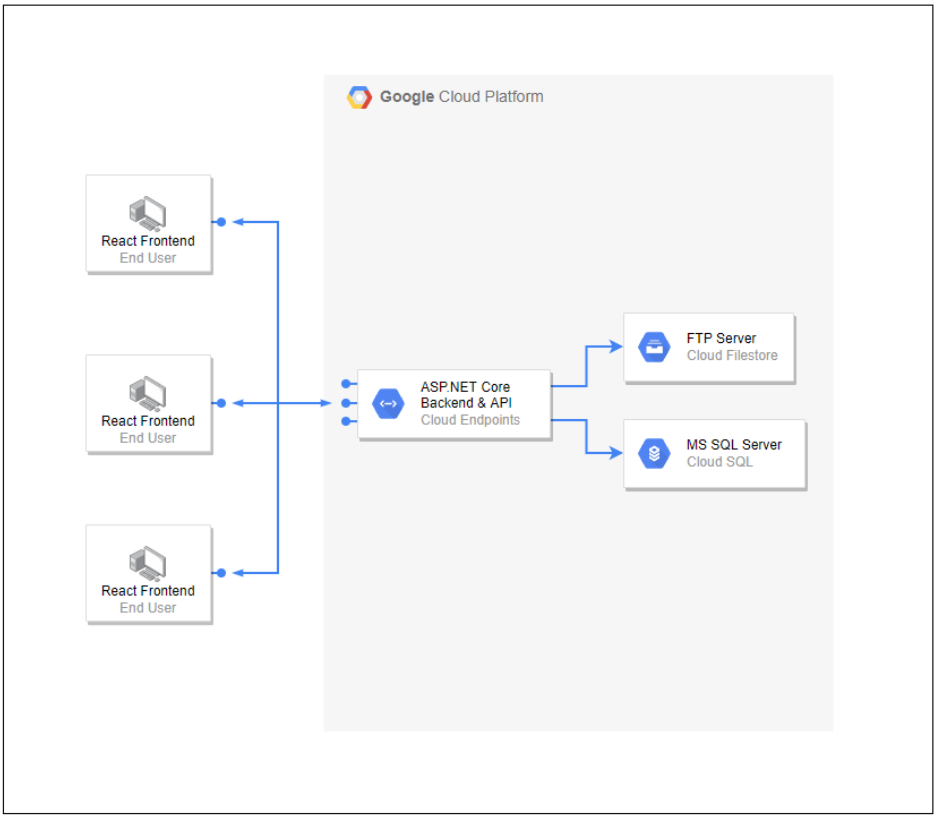

Figure 1 – eLearning Application Architecture

Similarly, the backend was chosen so that it was equally decoupled from the frontend. We chose ASP.NET Core as the backend technology since using it along with the API-first design pattern gives the backend development team the flexibility to write the backend independently of the frontend. Furthermore, C#'s strong typing, and the rigidness that comes with it, balances the flexibility and potential chaos of the weak-typed JavaScript/React frontend. This allows the backend developers to model the database structure and data objects in a convenient manner while still being able to pass data that is required by the frontend.

An example helps clarify the advantages. Let us examine the Add Course workflow for this purpose. The backend's CourseController class's AddCourse method simply requires an incoming Course object. The backend can assume the object is a complete object that does not require manipulation (although the capability to manipulate that object is there for complicated cases). It simply receives the data, ensures the user is properly authenticated, and makes the call to the SQL Server to add the new course. The frontend counterpart, the Add Course page, is similarly decoupled in that it does not have to handle any special interactions with the backend, it simply receives inputs from the user, sets those inputs as properties on an object it defines as a Course object, and passes that object to the AddCourse API method. As long as the documentation defines the Course object ahead of time, the two pieces can be written entirely separate from each other, tested separate from each other, and brought together when both are completed. This prevents delays where a frontend feature is delayed because it is waiting on a backend feature, or vice versa. Avoiding these types of development roadblocks is the main strength of this architecture.

Similarly, the SQL Server instance and FTP server instance follow the same containerization principles. They are existing images of a SQL Server and an FTP server which provide an interface for interacting

with them. The SQL Server instance uses SQL Server authentication and common SQL commands for interaction, whereas the FTP server follows the common authentication and file transfer interfaces available on all FTP servers. The backend can then use a SQL library of its choice and an FTP library of its choice to interact with these servers, avoiding all potential roadblocks by using common interfaces and contracts instead of coupled proprietary interfaces.

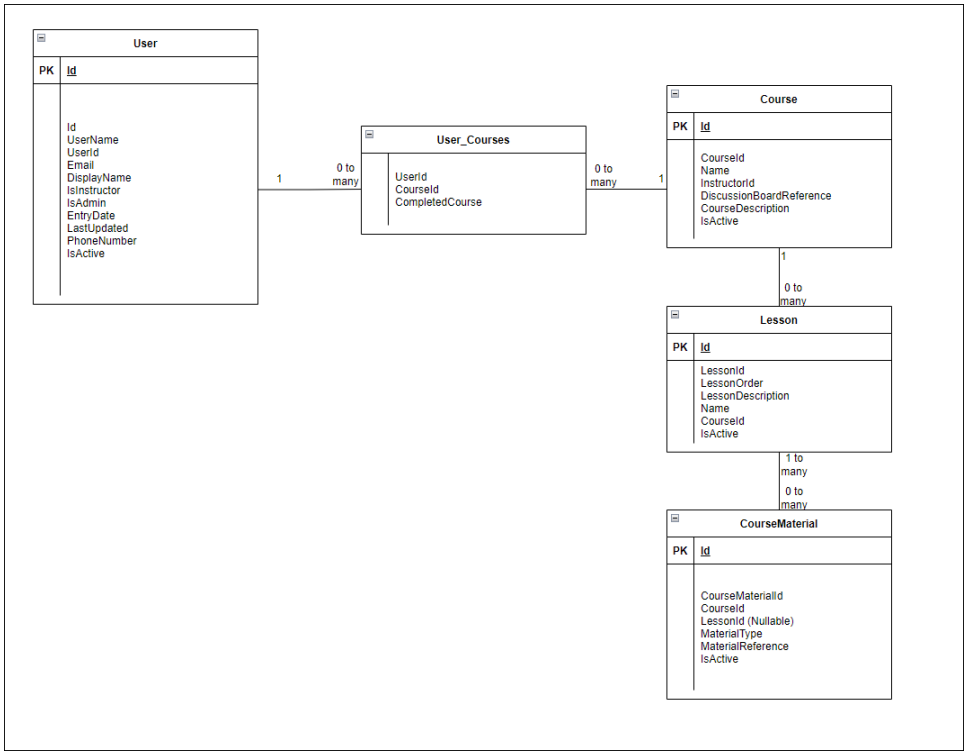

Figure 2 – eLearning Data Model

So, as can be seen in Figure 2, each slice of the architecture can be created independently of each other. The models serve as the interface. The database must contain tables and columns to describe the models. The backend must give classes for the models, have API calls to send and receive them to the frontend, and methods to perform FTP and database CRUD operations based on those models. The FTP server must simply be an FTP server – capable of receiving, storing, and sending files. And finally, the frontend must provide a way to get and create these models based on user input and display. So, the task of designing the system is suddenly simplified to the process of adding slices of the workflows described in Section 2.2, and the advantages of the architectural decisions are apparent.

2.2 Application Information flows

- Register a new account
  - Home Page -> Click **Register** -> Enter email, password, and confirm password -> Click **Register**
- Log in to an existing account
  - Home Page -> Click **Login** -> Enter email and password -> Click **Log in**
- Log out
  - At the top of any page, click **Logout**
- View all courses
  - Sign in -> User Links -> Available Courses

- Register for a course
    - Sign in -> User Links -> Available Courses -> Find the course you want to register for -> Click **Register**
- View courses you are currently registered for
    - Sign in -> User Links -> Registered Courses
- View a course page
    - Sign in -> User Links -> Registered Courses -> Find the desired course -> Click **Course Page**
- View a lesson page
    - Sign in -> User Links -> Registered Courses -> Find the desired course -> Click **Course Page** -> Find the desired lesson -> Click **View Lesson**
- View course discussion board
    - Sign in -> User Links -> Registered Courses -> Find the desired course -> Click **Course Page** -> Discussion Board
- Begin a new thread on the discussion board
    - Sign in -> User Links -> Registered Courses -> Find the desired course -> Click **Course Page** -> Discussion Board -> Fill out the **Thread Title** and **Thread Text** fields -> Click **Submit Thread**
- Reply to an existing thread
    - Sign in -> User Links -> Registered Courses -> Find the desired course -> Click **Course Page** -> Discussion Board -> Find the desired thread -> Fill out the **Submit a Post** field -> Click **Submit Post**
- Change user settings
    - Sign in -> User Links -> Edit User Settings -> Make the desired changes -> Click **Save Changes**
- (Instructor only) View courses you have created
    - Sign in to an Instructor account -> Instructor Links -> Instructor Courses
- (Instructor only) Create a course
    - Sign in to an Instructor account -> Instructor Links -> Instructor Courses -> **Create New Course** -> Fill in the necessary information -> **Create Course**
- (Instructor only) Edit a course information
    - Sign in to an Instructor account -> Instructor Links -> Instructor Courses -> Find the desired course -> Click **Edit Course** -> Make the desired changes -> Click **Save Changes**
- (Instructor only) Add a lesson to a course
    - Sign in to an Instructor account -> Instructor Links -> Instructor Courses -> Edit Course -> Add Lesson to Course -> Fill out the necessary information -> Click **Create Lesson**
- (Instructor only) Add a YouTube video to a lesson
    - Sign in to an Instructor account -> Instructor Links -> Instructor Courses -> Find the desired course -> Click **Edit Course** -> Find the desired lesson -> Click **Edit Lesson** -> Enter the YouTube video URL into the **YouTube Video URL** field -> Click **Submit New Course Material**
- (Instructor only) Add a PDF file to a lesson
    - Sign in to an Instructor account -> Instructor Links -> Instructor Courses -> Find the desired course -> Click **Edit Course** -> Find the desired lesson -> Click **Edit Lesson** -> In the **Add PDF** section, click **Choose File** -> Browse to the location of the desired PDF file on your computer -> Select the desired PDF file -> Click **Open** -> On the **Edit Lesson** page, click **Submit PDF**

- (Instructor only) Add a video file to a lesson
    - Sign in to an Instructor account -> Instructor Links -> Instructor Courses -> Find the desired course -> Click **Edit Course** -> Find the desired lesson -> Click **Edit Lesson** -> In the **Add Video** section, click **Choose File** -> Browse to the location of the desired PDF file on your computer -> Select the desired video file -> Click **Open** -> On the **Edit Lesson** page, click **Submit Video**
- (Admin only) Edit user permissions
    - Sign in to an admin account -> Admin Links -> Admin Permissions -> Find the desired user -> Make the desired changes -> Click **Save**

2.3 Interactions with other Applications

From a user perspective, the eLearning application directly interacts with only one other existing application. It uses YouTube as a file host option for instructors who prefer to host their videos on YouTube instead of on our native ftp server platform. In terms of infrastructure, the eLearning application interacts with Gitlab for deployments, an FTP server for file hosting, and a web server for application hosting. In a production environment, the FTP server and web server are both cloud-hosted servers, so the application's infrastructure depends on the cloud solution chosen by the deployment team.

2.4 Capabilities

The application has several dependencies, most of which are related to its infrastructure. The front-end is a react application, thus it gets passed to the client from the backend and does not require any additional capabilities. The backend itself, however, does interact with 2 other infrastructure pieces.

First, it requires a connection to a SQL Server instance. The instance does not have to be a specific architecture – it can be a static instance, it can be a bare-metal instance, or ideally, it would be a cloud instance. For development purposes, we hosted our SQL Server instance as local a docker instance using Microsoft's recommended SQL Server docker image. A production instance should host it in a static and safe place such as Google Cloud Platform (GCP). This instance is required, and it must provide the backend the ability to save and retrieve several different data types. It must be able to store and retrieve User, Course, CourseMaterial, Lesson, and UserCourse objects. It also must store ApplicationUser objects, in addition to several helper tables implemented by Entity Framework's authentication system, as recommended by the maintainers of ASP.NET Core.

In addition to the SQL Server instance, it must also have access to an FTP server. In development, the FTP server was hosted as a local docker instance using the stilliard/pure-ftpd container image. This allowed us to develop and run the system locally as if it were an actual FTP server. In production, the instance should be a cloud-hosted FTP server, and it should ideally be run on the same cloud platform as your SQL Server. The backend should be able to connect to the instance to both store and retrieve 3 different file types. It must be capable of handling PDF files, mp4 files, and discussion board JSON files.

2.5 Risk Assessment and Management

According to webfx.com there are many costs associated with developing and implementing an e-learning website. These costs include Web design ($12,000), website maintenance ($400), website marketing ($2,500), website design services ($500), website maintenance services ($2,500). These initial costs would normally start at $18,400, however because our team consists of developers most of these costs are eliminated; apart from marketing; so, the initial cost for implementation of the website would be a minimum of $2,500. Hosting a "high traffic" website on AWS will cost approximately $720 a year, and

GCP has a comparable cost. These costs combined are approximately $3,220, which are team has agreed to evenly distribute amongst themselves.

3 Project Requirements

3.1 Identification of Requirements

<**GSU-GS_FA2021-1 Admin-Capability-00100**>
The project must allow administrator users to give other users the administrator role.
Implementation: Mandatory

<**GSU-GS_FA2021-1 Admin-Capability-00200**>
The project must allow administrator users to give other users the instructor role.
Implementation: Mandatory

<**GSU-GS_FA2021-1 Admin-Capability-00300**>
The project must allow administrator users to remove the administrator role from other users.
Implementation: Mandatory

<**GSU-GS_FA2021-1 Admin-Capability-00400**>
The project must allow administrator users to remove the instructor role from other users.
Implementation: Mandatory

<**GSU-GS_FA2021-1 Guest-Capability-00500**>
The project must allow guest (unregistered) users to create an account. The guest user must enter their email address, desired display name, and password, and the system must create a user account using that data.
Implementation: Mandatory

<**GSU-GS_FA2021-1 Instructor-Capability-00600**>
The project must allow instructor users to upload learning materials.
Implementation: Mandatory

<**GSU-GS_FA2021-1 Instructor-Capability-00700**>
The project must allow instructor users to monitor student progress.
Implementation: Mandatory

<**GSU-GS_FA2021-1 Instructor-Capability-00800**>
The project must allow instructor users to answer questions.
Implementation: Mandatory

<**GSU-GS_FA2021-1 Instructor-Capability-00900**>
The project must allow instructor users to upload learning materials.
Implementation: Mandatory

<**GSU-GS_FA2021-1 User-Capability-01000**>
The project must allow users to register for classes.
Implementation: Mandatory

<**GSU-GS_FA2021-1 User-Capability-01100**>
The project must allow users to view courses they registered.
Implementation: Mandatory

<**GSU-GS_FA2021-1 User-Capability-01200**>
The project must allow users to view lessons for classes they have registered.
Implementation: Mandatory

<**GSU-GS_FA2021-1 User-Capability-01300**>
The project must allow users to view materials for each lesson in courses they have registered.
Implementation: Mandatory

<**GSU-GS_FA2021-1 User-Capability-01400**>
The project must allow users to view the discussion board for their courses.
Implementation: Mandatory

<**GSU-GS_FA2021-1 User-Capability-01500**>
The project must allow users to post to discussion boards for their courses.
Implementation: Mandatory

<**GSU-GS_FA2021-1 User-Capability-01600**>
The project must allow users to edit their personal information – display name and phone number.
Implementation: Mandatory

3.2 Operations, Administration, Maintenance and Provisioning (OAM&P)

For this application, we are hosting our web, SQL, and FTP servers on a cloud service. The cloud service, as per terms of cloud service, are responsible for data backup, fault recovery, uptime, and maintenance.

Administrator users may grant or revoke Instructor or Admin status to any number of users.

3.3 Security and Fraud Prevention

Because our development team is lean, internal security issues should not be an immediate concern. However, upon the growth of our e-learning website, we will have to hire new employees to handle the workload of each piece of our project. For this issue we plan to compartmentalize each aspect of our project. This should protect user, instructor, and administrator information. To address any future internal threats, we will issue temporary accounts for temporary employees including but not limited to interns and seasonal employees.
The biggest external issues that we will encounter is through malicious hackers attempting to gain access to our user data for either one of two reasons, first to obtain personal information such as emails, secondly to gain access to user financial information. To combat this, we will implement a two-step authentication process to prevent unauthorized access to users' accounts.

3.4 Release and Transition Plan

The release and transition plan entirely follows the release workflow set up by Gitlab and their Git flow process. For the initial deployment, a cloud-hosted web server service is chosen – we chose Google Cloud

Platform (GCP), but Amazon Web Service (AWS) would work all the same. The GCP service would be created, and a domain would be purchased. Once that is configured, we can create a Gitlab environment in their UI which points at the URL supplied by GCP. Using this environment in Gitlab, we can choose to deploy a tag (such as v1.0.0) from our Gitlab repo, which will deploy the containerized application to the GCP instance. At that point, our customers would be able to reach our application at the domain we specify in GCP.

One of the largest advantages of following Gitlab's process is that the system for developing newer releases is exactly the same – except fewer steps since we already have our production environment defined in Gitlab. We would write and push our new feature to our main branch in Gitlab. Then we can create a new tag from that instance of our main branch, such as v1.1.0. Once that tag has been created, we can use Gitlab's environments UI to deploy our new v1.1.0 tag to our production environment. Once Gitlab has completed the deployment, the customers will now have access to our new v1.1.0 application at the same URL that previously pointed to our v1.0.0 application.

4 Project Design Description

We decided to use React.js for our front-end, we made this decision because React.js is simple to use, supports server side, loads very fast in both web browsers and phone apps, and is industry standard.

Our team decided to you a simple navigational bar at the top of the page, as can be seen in Figure 3. There are a few main reasons for this design pattern. First, consumers read and process websites the same way that they do books; their eyes start at the top right corner and scan left then move onto the next line of text or images. Second, this format is statistically one of the most commonly used design patterns. Therefore, our users will be accustomed to it.

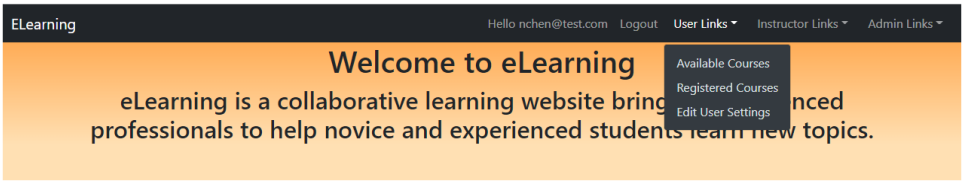

Figure 3 – eLearning Navigation Bar with all links (only available to superusers)

The next selection was to decide on a color scheme, we decided to incorporate the colors of Governors state university within our site. We used an orange linear gradient as the background color, along with black text which added to the readability. We also added a roll over feature to the buttons, when the user places the mouse over one of the buttons it changes from white to orange.

Figure 4 – Available Courses Page

On the back end we used an SQL database to store user, instructor, and administrator information.

In terms of design units, we narrowed the scope down to 4 different areas – user, instructor, administrator, and guest design units. The guest design unit is responsible for the workflows before a guest has an account with the site. The main workflow is to register for the site, which then leads into the user design unit.

The user design unit represents the average user of the site. Accounts with the user role must be able to register for courses, view those courses and their lessons and materials, view, and post on their discussion boards (Figure 5), and edit their own account's personal information.

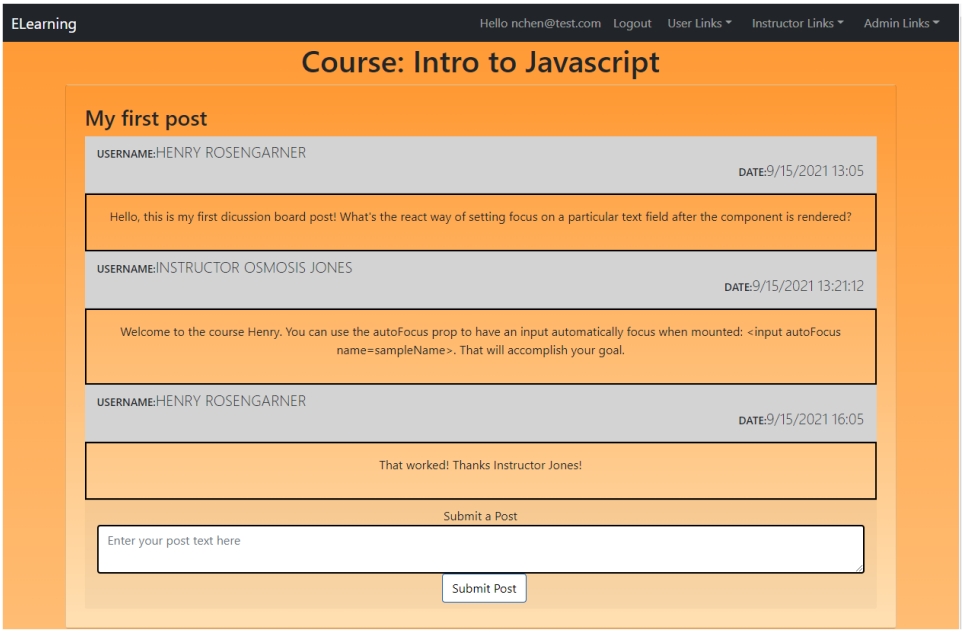

Figure 5 – Course Discussion Board Page

The instructor design unit represents accounts that can add new courses to the site. These accounts can create new courses, add lessons to their courses, add course materials to their lessons, and view discussion boards for their courses.

Finally, the last design unit is the administrator design unit. The administrator design unit represents accounts which can edit the account privileges of other existing accounts. They can add and remove both instructor and administrator privileges for other accounts. Figure 6 shows the simple-to-use interface which provides administrators with these capabilities.

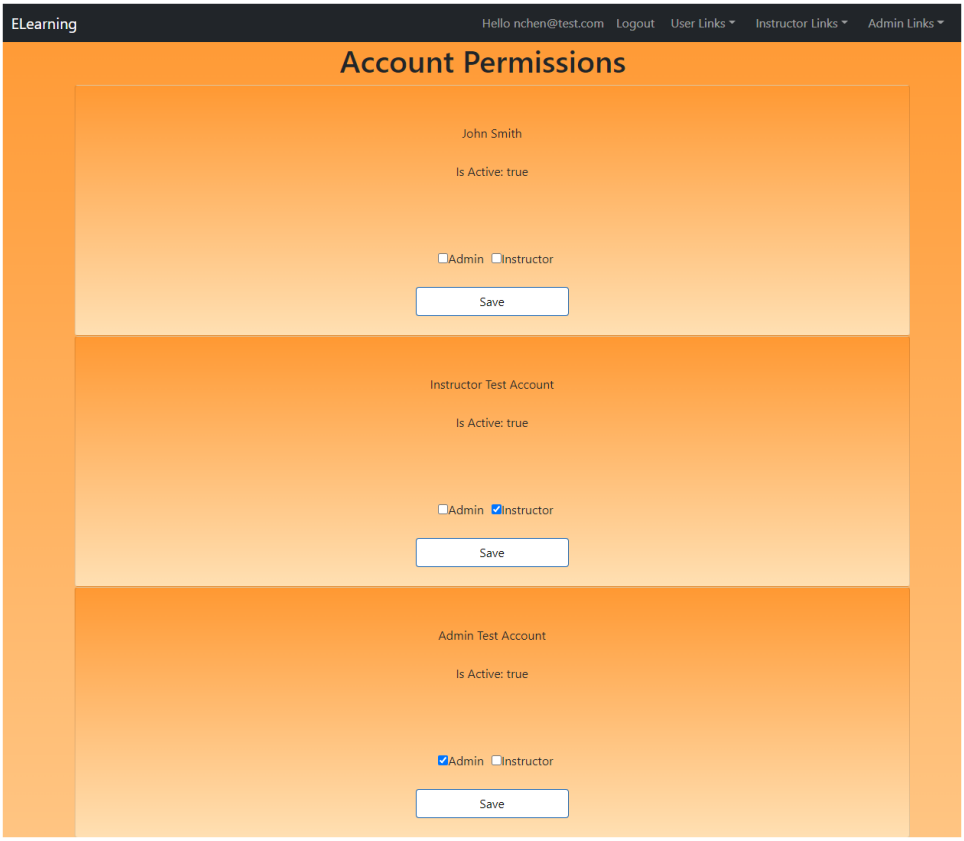

Figure 6 – Administrator's Edit Account Permissions Page

5 Internal/external Interface Impacts and Specification

The basic data structures of our application are as follows:

- User
  - UserID (int)
  - DisplayName (string)
  - IsInstructor (boolean)
  - IsAdmin (boolean)
  - EntryDate (date)
  - LastUpdated (date)
  - Phone (string)
  - IsActive (boolean)
- Course
  - CourseID (int)
  - Name (string)
  - InstructorID (int)

    - DiscussionBoardReference (string)
    - CourseDescription (string)
    - IsActive (boolean)
- Lesson
    - LessonID (int)
    - Name (string)
    - LessonOrder (int)
    - LessonDescription (string)
    - CourseID (int)
    - IsActive (boolean)
- UserCourse
    - UserID (int)
    - CourseID (int)
    - IsCompleted (boolean)
- CourseMaterial
    - CourseMaterialID (int)
    - CourseID (int)
    - LessonID (int)
    - MaterialType (string)
    - MaterialReference (string)
    - IsActive (boolean)
    - YouTubeLink (string)

All data structures affect the User, Instructor, and Administrator design units since both Instructor and Administrator are subsets of User.

Each model has a corresponding table in the eLearning SQL database, where data related to that model is stored. The application reads the data from, or writes the data to, the database as necessary.

6 Design Units Impacts

The eLearning website is composed of 4 functional design units:

1. User
2. Instructor
3. Administrator
4. Guest

6.1 Design Unit 1 - eLearning Website User

6.1.1 Functional Overview

The user functional design unit was created during this iteration. It added several new features – mainly the ability to edit personal account information, sign up for courses, post to course discussion boards, and view course materials.

6.1.2 Impacts

The main impacts are the addition of the 4 features listed above. The ability to edit personal account information allows the user to change the information associated with their account – namely, their phone number and display name. Also, the ability to sign up for courses was added, which allows a user to

register courses to their account. In addition to registering courses, the users can also post to the course's discussion board and view the course materials.

6.1.3 Requirements

<**GSU-GS_FA2021-1 User-Capability-01000**>
<**GSU-GS_FA2021-1 User-Capability-01100**>
<**GSU-GS_FA2021-1 User-Capability-01200**>
<**GSU-GS_FA2021-1 User-Capability-01300**>
<**GSU-GS_FA2021-1 User-Capability-01400**>
<**GSU-GS_FA2021-1 User-Capability-01500**>
<**GSU-GS_FA2021-1 User-Capability-01600**>

6.2 Design Unit 2 - eLearning Website Instructor

6.2.1 Functional Overview

The instructor design unit was added during this software development cycle. The instructor design unit is available to accounts with the instructor role. These accounts can create new courses, add new lessons to courses, add materials to lessons, and view discussion boards for their courses.

6.2.2 Impacts

With the addition of the instructor design unit, accounts can now be given the instructor role. The instructor role allows users to create courses. Once a course has been created with a given name and description, the instructor can then add lessons to the course. Each lesson can then be customized to include PDFs, YouTube links, and videos which then are available to users who are taking the courses. Finally, instructors can also view the number of students who have registered for and completed their courses, and they can also view the discussion board for their courses.

6.2.3 Requirements

<**GSU-GS_FA2021-1 Instructor-Capability-00600**>
<**GSU-GS_FA2021-1 Instructor-Capability-00700**>
<**GSU-GS_FA2021-1 Instructor-Capability-00800**>
<**GSU-GS_FA2021-1 Instructor-Capability-00900**>

6.3 Design Unit 3 - eLearning Website Administrator

6.3.1 Functional Overview

The 3rd design unit, the administrator design unit, was added with this development iteration. It defines the administrator account role and adds the ability to alter other account privilege levels.

6.3.2 Impacts

As part of the first development iteration, the administrator account role was added. The role is a special permission designed for system administrators. Accounts with this role can navigate to the Admin Permissions page, where they can view all existing accounts. In addition to being able to view existing accounts, they can alter the privilege levels of those accounts. Admin accounts can give other accounts

administrator and instructor privileges. Also, they can remove administrator and instructor privileges from other accounts.

6.3.3 Requirements

<**GSU-GS_FA2021-1 Admin-Capability-00100**>
<**GSU-GS_FA2021-1 Admin-Capability-00200**>
<**GSU-GS_FA2021-1 Admin-Capability-00300**>
<**GSU-GS_FA2021-1 Admin-Capability-00400**>

6.4 Design Unit 4 - eLearning Website Guest

6.4.1 Functional Overview

As part of this iteration, the guest design unit was added. The guest design unit should allow an unregistered user to use the register page to create a new account.

6.4.2 Impacts

This functional area was newly created. It allows unregistered users to navigate to the newly created register page. On the register page they can enter their email address, a password for the account, and their desired display name. Then the user clicks the register button, and the website will create their new account.

6.4.3 Requirements

<**GSU-GS_FA2021-1 Guest-Capability-00400**>

7 Acknowledgements

We would like to thank Professor Xin Chen, Governors State University, for her contribution and dedication to our project in helping us achieve our goals.

8 References

Compare plans and pricing. Coursera For Business. (n.d.). Retrieved December 1, 2021, from https://www.coursera.org/business/compare-plans/.

Docs.aws.amazon.com - AWS documentation. (n.d.). Retrieved December 1, 2021, from https://docs.aws.amazon.com/.

Google. (n.d.). Google cloud documentation. Retrieved December 1, 2021, from https://cloud.google.com/docs.

Internet marketing prices for every strategy. WebFX. (n.d.). Retrieved December 1, 2021, from https://www.webfx.com/Internet-Marketing-Pricing.html.

MashaMSFT. (n.d.). SQL Server Technical Documentation - SQL Server. SQL Server | Microsoft Docs. Retrieved December 1, 2021, from https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15.

Overview of docker compose. Docker Documentation. (2021, December 1). Retrieved December 1, 2021, from https://docs.docker.com/compose/.

Plans. Udemy Business. (2021, December 1). Retrieved December 1, 2021, from https://business.udemy.com/plans/.

Pluralsight Subscription Plans and Pricing: Monthly and annual. Pluralsight Subscription Plans and Pricing: Monthly and Annual. (n.d.). Retrieved December 1, 2021, from https://www.pluralsight.com/pricing.

React top-level API. React. (n.d.). Retrieved December 1, 2021, from https://reactjs.org/docs/react-api.html.

Rick-Anderson. (n.d.). Introduction to asp.net core. Microsoft Docs. Retrieved December 1, 2021, from https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0.

Stilliard. (n.d.). Basic example walkthrough · Stilliard/Docker-pure-ftpd wiki. GitHub. Retrieved December 1, 2021, from https://github.com/stilliard/docker-pure-ftpd/wiki/Basic-example-walk-through.

Yifat Perry, P. M. L. (2021, March 20). Google cloud pricing vs AWS: A fair comparison? NetApp Cloud Solutions. Retrieved December 1, 2021, from https://cloud.netapp.com/blog/google-cloud-pricing-vs-aws-a-fair-comparison-gcp-aws-cvo-blg.