A Simple Course Management Website

A Senior Project

Presented to

The Faculty of the Computer Engineering Department

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

Of the Requirements for the Degree

Bachelor of Computer Engineering

By

Brendan Baronia

June 2018

Table of Contents

Abstract

Over the past two quarters, I have been developing a basic course management system for use by teachers and students. The project started as an alternative to existing platforms, such as Moodle, though because of a two quarter time constraint the complexity of the system was reduced to a more reasonable level. The resulting system requirements included a simple course creation/enrollment system, with separate sections, topics, and resources including videos, files, and a basic graph problem.

This document examines the design and implementation of the system, from an initial design proposal to a final functional product. It examines the design choices for both the user interface and server operation, the technologies used to achieve the final product, the implementation process of both front and back end systems and all challenges encountered, and proposed future improvements to the system.

Introduction

System Inspiration

This course management system was designed as a teaching tool to be used in classrooms as a lecture supplement and at home as a review resource. This idea was driven by my mom, a teacher at a junior high school with few technologies available. She has access to Chromebooks for each student in her class, so I designed this webpage to assist in teaching and classroom activities. I developed the system primarily as a lecture assistance tool. Teaching at a middle school level, my mom has developed a curriculum that caters to the short attention span of her students by adding many fun activities, and I developed the system to bring these activities online.

Website Design

The resulting website uses an organization system to allow students to navigate according to topics they'd like to review. Each topic has associated activities to assist in learning and review. For the purpose of senior project, I focused on developing a course on 2 dimensional plots, specifically line graphs. One key activity I implemented is a line plotting tool that uses random, procedurally generated equations and plots them on a graph to demonstrate plotting concepts. This plotting tool is contained within a review tool, which uses procedurally generated review problems to assist students in learning. These problems include hints that the student can reveal as they progress through the problem. Other activities available for topics include embedded videos and files, allowing teachers to provide students with videos and documents.

Teachers may create courses, sections within each course, topics within each section, and resources within each topic. The teacher can view a list of all created courses and sections upon logging in, and after selecting a section the teacher may view all topics and resources contained within that section. The teacher may also protect sections from unauthorized access by adding a passcode required for enrollment.

The teacher can enable and disable topics and individual activities housed inside these topics to simplify navigation for students and to hide topics that may not have been covered in class yet. For the purpose of senior project, I included a demonstrative graphing topic with the ability to hide and display the topic dedicated to line graphs, and the ability to hide and display the individual activities within this topic.

Students may enroll in a course section, using the section ID provided by the teacher, and if required a section passcode. Once enrolled in a section, the student will see the section listed on the home page and can select the section to view all topics and resources contained within that section that the teacher has revealed to students.

Server Design

Courses, course topics, and activities are managed by a MySQL database. Node.js and various libraries contained within the runtime manage all server operations, including webpage rendering, database interaction, file storage, user authentication, and page navigation. The front end is built with CSS and HTML, using Bootstrap as a design library, as well as libraries such as jQuery to support dynamic content. Overall, the system was designed to be portable and simple to set up, requiring only Node.js and MySQL to be installed on the server. It was also designed

to be expandable, so that in future iterations other features and resource types can be added easily without major modification to the base system.

Initial System Description

Functional Specifications

This system was designed specifically as a course management system. To accomplish this task, the following requirements were developed:

- The system must allow teachers to create courses, and create separate sections for each course
- The system must allow teachers to create topics for each section to help guide students looking for a specific topic.
- The system must allow the teacher to add resources to each topic, including sample problems, embedded videos, and files such as PDFs.
- The system must allow the teacher to hide topics from students or show topics to students so that topics can be revealed as they are covered in class.
- The system must allow the teacher to hide resources from students or show resources to students so that resources can be revealed as required: e.g. after a test, a teacher can reveal solutions.
- The system must allow teachers to secure section enrollment with a passcode to prevent unauthorized access.
- The system must allow students to enroll in sections to access provided materials.
- The system must allow students to view topics and resources for each section they are enrolled to if the topics and sections have been revealed by the teacher.
- The system must have protected endpoints so that unauthorized users cannot access course materials.

In addition to these basic system functions, a problem was also developed according to these requirements:

- The problem must cover the basics of plotting a 2-dimensional line.
- The problem must include multiple steps, with hints provided for each step.
- The full process for each step must be initially hidden, but available to be revealed if further help is needed.
- The problem must involve random numbers so that each problem is individual.
- The problem must include a graphing component to demonstrate the basics of plotting the line.

The system was designed and implemented using a Raspberry Pi as the server, so the system also needed to be lightweight and easy to run. In a production environment, a system with more memory and storage is recommended to manage multiple users.

The website was designed to support the latest stable release of most browsers in a desktop environment. It supports the latest versions of Google Chrome, Mozilla Firefox, Microsoft Edge, Internet Explorer 10+, and Opera. It also supports Safari for Macs.

User Scenarios

To assist with development and functional requirement specifications, a number of user scenarios were developed. These user scenarios don't cover every system operation, but were meant to provide an overview of how the system should operate.

- Teacher creates new course
    1. Teacher accesses webpage, is presented with login

2. Teacher enters credentials and logs in
3. Teacher is presented with list of current courses
4. In upper left corner of course list, presses button to add new course
5. Teacher is presented with form to fill in for new course
6. Enters course name, course number, section number, description, locked or open, password to enroll, etc. Only course name is required. Course defaults to locked.
7. Teacher presses button at bottom of page to save course
8. Teacher is returned to list of current courses
9. Sections have associated unique numbers for students to add displayed to left of course name

- Teacher adds topic to course
    1. Teacher accesses webpage, is presented with login
    2. Teacher enters credentials and logs in
    3. Teacher is presented with list of current courses
    4. Teacher clicks course to edit
    5. Teacher is presented with list of course topics
    6. Teacher presses button in top left corner to add new course topic
    7. Teacher is presented with form to fill in for new topic
    8. Teacher enters topic name, description, hidden or viewable, etc. Only name is required, topic defaults to hidden.
    9. Teacher presses button at bottom of page to save course
    10. Teacher is returned to list of course topics

11. Teacher can click and drag course topic to move into proper order

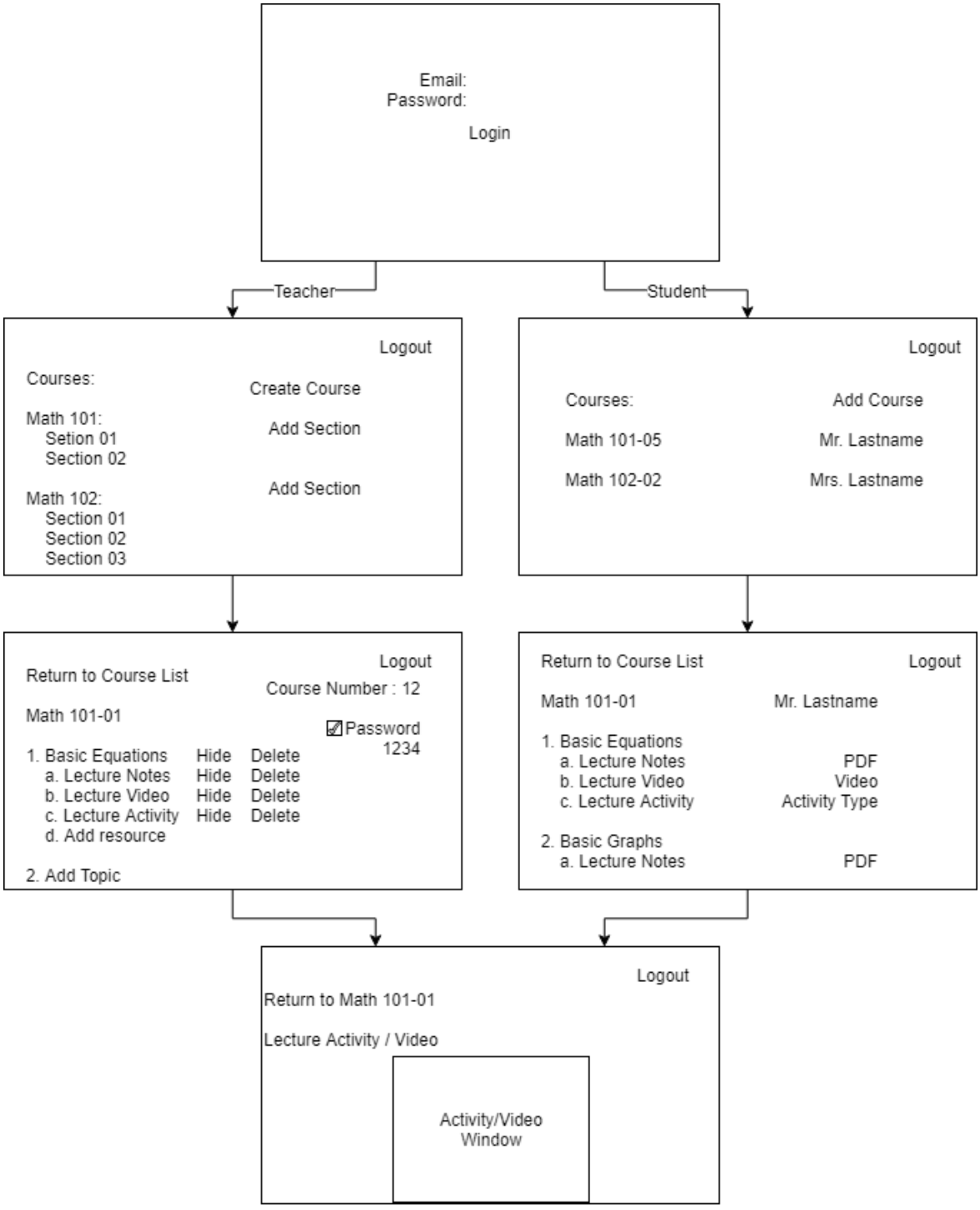
- Teacher adds lecture notes to course
    1. Teacher accesses webpage, is presented with login
    2. Teacher enters credentials and logs in
    3. Teacher is presented with list of current courses
    4. Teacher clicks course to edit
    5. Teacher is presented with list of course topics
    6. Under course topic header, teacher presses button to add new material
    7. Teacher selects type of material to add as PDF
    8. Teacher enters PDF name and selects PDF to upload from file system
    9. Teacher presses save at bottom of page
    10. PDF is added to course topic
- Teacher reveals course topic
    1. Teacher accesses webpage, is presented with login
    2. Teacher enters credentials and logs in
    3. Teacher is presented with list of current courses
    4. Teacher clicks course to edit
    5. Teacher is presented with list of course topics
    6. Hidden topics will be in grey
    7. Teacher can click on topic header to bring up edit topic info
    8. Teacher changes from hidden to viewable
    9. Teacher presses button to save at bottom of page

- Student adds new course
    1. Student accesses webpage, is presented with login
    2. Student enters credentials and logs in
    3. Student is presented with list of current courses
    4. Student clicks button on top left of page to add course
    5. Student enters teacher's login username, course number, and password if required. If no password is required, field is left blank.
    6. Student is returned to list of current courses with new course highlighted
- Student accesses course lecture
    1. Student accesses webpage, is presented with login
    2. Student enters credentials and logs in
    3. Student is presented with list of current courses
    4. Student clicks course to access
    5. Student is presented with list of course topics
    6. Under each topic header is a list of materials
    7. Student clicks lecture to view and is presented with PDF of lecture

These user scenarios were developed early in the design process, and in the production system a number of these scenarios were modified. Most of the changes were made to fit the aesthetic choices made for the website, or to streamline processes when creating or adding course components.

Storyboard

To assist in early development, I created the following storyboard. Most of the design changed to streamline the design, but the same basic page layout remained.



Design

Technologies Used

This system was developed primarily on Node.js, using Express as a web framework. There are a number of options for server technologies, including web servers like Apache and Nginx, each with their own benefits and drawbacks, as well as different development languages, like PHP and Ruby on Rails. I chose Node.js primarily because of its use of JavaScript, allowing me to develop most of the front end and back end in a single language, as well as the versatility and expandability of the system.

Within Node.js, I used a number of different packages to achieve different tasks.

I used Passport to enable local authentication and session management, with the intent of later expanding authentication to use systems like OAuth. Passport provides authentication strategies for a number of different web authenticators, so while it provides little benefit for local authentication, the expandability made it a good choice. Combined with Session, another package, it allows for persistent user sessions as well.

I used nodemon during my development as an application manager for Node. Nodemon houses a number of different features, including file monitoring and automatic system restarts on changes, so I was able to develop the system without manually restarting the system on each change. This also allows system updates to be easily executed without needing to manually restart the server.

To manage files, I used Multer. Multer allows specification of a file folder and naming scheme for uploaded files, and can automatically upload and organize files. Apart from initial

setup, Multer is automatic and can filter files according to size and type, meaning that in a future update security vulnerabilities due to file uploads can be removed.

Bcrypt was used as an encryption method for passwords. Bcrypt was designed as a slow hashing method, meaning that brute force attacks on passwords will take much longer, and given the expected low volume of the server (limited to a single school), the speed of the hashing algorithm shouldn't adversely affect server performance. Other hashing methods, such as SHA hashes, are much faster, allowing brute force attacks to be run much more quickly.

MySQL was used for database management. There are a number of different database systems available, including MongoDB, PostgreSQL, Oracle Database, and Microsoft SQL, each having their own strengths and weaknesses. I opted to use MySQL primarily because of experience with the system and its availability as open-source software.

Finally, Handlebars was used as a layout manager to integrate HTML into Node.js. Handlebars uses syntax very similar to HTML, unlike other templating tools available for web design, allowing me to write much of my website without needing to learn another syntax. It also allowed me to change a number of website components without needing to edit HTML, but rather to simply adjust injection in Node.

On the front end, the website was built primarily using Bootstrap as a component library. Bootstrap was developed as an open source library to create websites with a modern appearance, and provided a number of helpful components to develop a website with a consistent appearance throughout. Bootstrap also provided backing scripts to create responsive components, allowing me to integrate many features into a single webpage.

What I couldn't achieve with Bootstrap's provided components, I created with jQuery. Using jQuery I was able to transfer much of the load populating websites with dynamic content from the server to the client; tasks like inserting course sections and topics into the webpage are handled client-side with jQuery so that the server has a lighter workload and can serve more clients.

To ensure that properly formatted inputs were required when submitting forms to the server, I used jQuery Validator, a validation library available for jQuery. This library is open source and provides a number of preconfigured input validation methods, as well as mechanisms for creating custom validators. Using this validation library enabled me to ensure properly formatted inputs on the client system, again moving system load away from the server.

System Architecture

This system uses a client-server architecture. A single server serves all clients, allowing them to access the webpage based on authentication levels.

The process for rendering each web page follows this pattern:

1. The client requests a web page from the server
2. The server renders the webpage and sends the corresponding HTML to the client
3. The client parses the HTML and requests the proper style pages and scripts from the server
4. The server sends all requests files to the client.
5. The client applies style to the webpage.

6. The client runs the scripts associated with the page and requests any necessary data from the server.
7. The server fetches data from the database and returns it to the client.
8. The client inserts the data into the webpage.
9. Upon user input, the client sends user requests to update data to the server.
10. The server updates the database accordingly.

Database Design

To house the different structures required for the system, I created a number of different tables in MySQL. I created the following tables:

I. User
    A. User ID (primary key, integer, autoincrement)
    B. Email (unique, string)
    C. First name (string)
    D. Last name (string)
    E. Password (encrypted, string)
    F. Privilege level (student, teacher)

II. Course
    A. Course ID (primary key, integer, autoincrement)
    B. Course name (string)
    C. Owner email (string, foreign key to User email)

III. Section

A. Section ID (primary key, integer, autoincrement)

B. Course ID (integer, foreign key to Course)

C. Section Description (string)

D. Section quarter (spring, summer, fall, winter)

E. Section year (integer)

F. Section passcode (string)

IV. Enrollment

A. Enrollment ID (primary key, integer, autoincrement)

B. User email (string, foreign key to User email)

C. Section ID (integer, foreign key to Section)

V. Topic

A. Topic ID (primary key, integer, autoincrement)

B. Section ID (integer, foreign key to section)

C. Topic name (string)

D. Topic description (string)

E. Visible (boolean)

VI. Resource

A. Resource ID (primary key, integer, autoincrement)

B. Resource name (string)

C. Resource type (file, video, problem)

D. Topic ID (integer, foreign key to topic)

E. Resource location (string)

F. Visible (boolean)

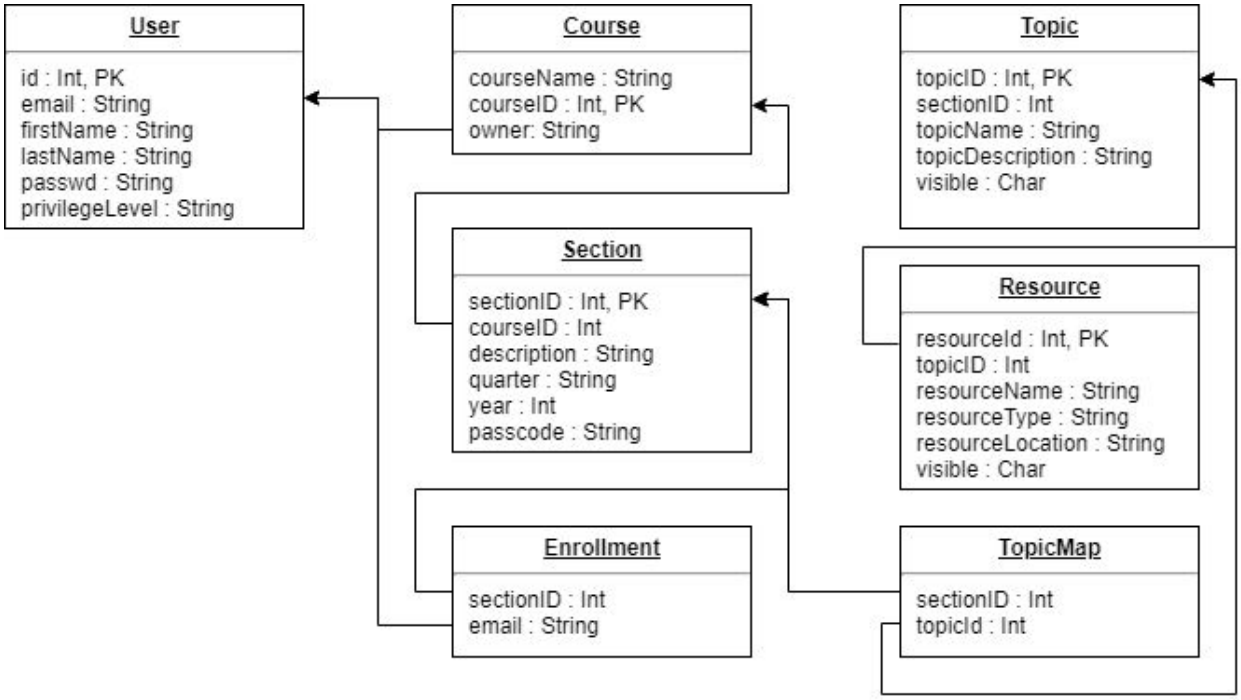
VII. Topic Map

A. Map ID (primary key, integer, autoincrement)

B. Topic ID (integer, foreign key to Topic)

C. Section ID (integer, foreign key to Section)

The tables are designed to interact according to the following diagram:



To promote good database design and avoid repeating data, two mapping tables were created; Enrollments and Topic Map. Students and Sections have a many to many relationship, so a mapping between the two was required. Topics and Sections have a many to one relationship at the moment, but future plans to allow duplicate sections mean a many to many

relationship will be created. The database was also normalized into the third normal form to prevent database anomalies.

Implementation

Implementation Process

Developing a project of this scale was far beyond my experience, so the implementation process was fairly disorganized. The implementation followed the following path:

1. Implementation began with creating the database. Because of experience with MySQL, this phase was fairly straightforward.
2. Basic navigation was configured in Node.js. Using Express as a framework for routing, I constructed simple page navigation.
3. With simple page navigation enabled, I began to create a website template using Bootstrap. I tested a number of different components early in the process to decide what I would use for the final product.
4. After finishing website layout with Bootstrap and basic HTML, I began developing scripts with jQuery to insert dynamic fields like course lists and topics into the website.
5. I moved back to Node.js, creating navigation paths to allow jQuery to request data from the server.
6. To test jQuery and the website, I created objects in JavaScript to match the database. I then sent these objects through Node.js as responses to jQuery requests and ensured that the website responded as expected to the data.
7. With the website and server functioning properly, I began integration with the MySQL database. This involved writing a number of queries to ensure that the proper data would be retrieved from the database.

8. Finally, with the website constructed and integrated with MySQL, I constructed the sample problem to demonstrate line plotting.

Challenges

When developing the system, I encountered a number of challenges. The first, and most serious challenge I faced was a lack of experience with most of the technologies I chose to use.

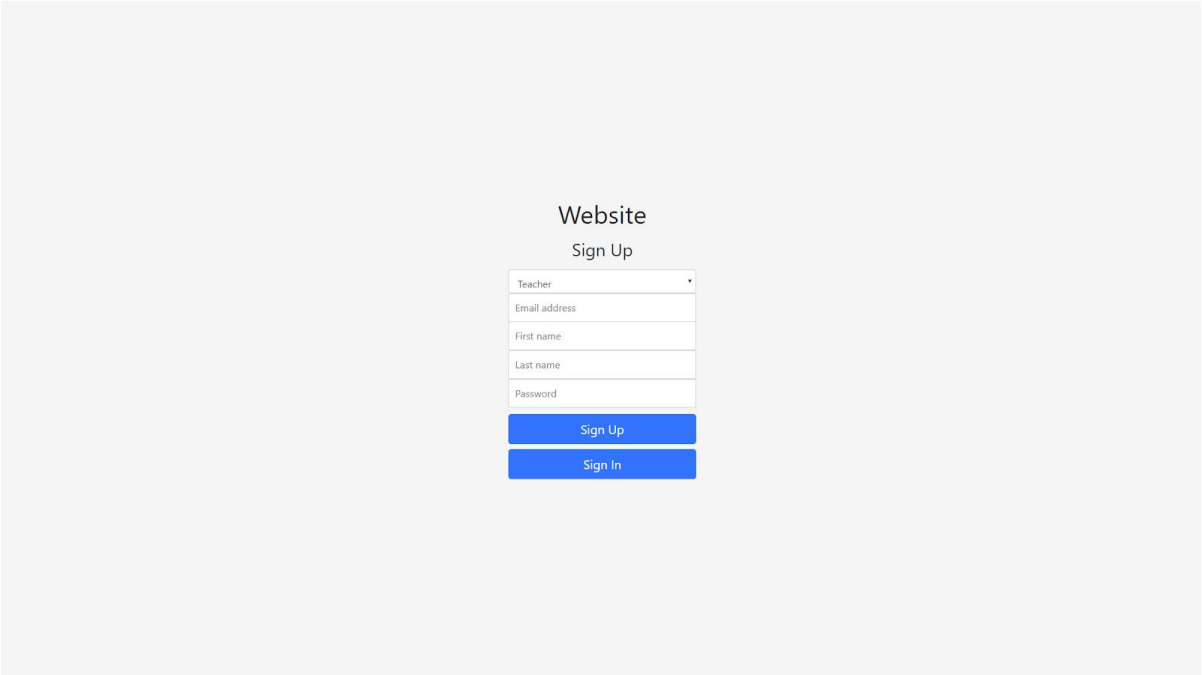
I had some experience with MySQL, JavaScript, and HTML/CSS, but had never used them to this extent before. In addition, I had very little experience with Node.js, and had no experience with jQuery, Bootstrap, Handlebars, or any of the Node.js packages I used. As a result, I spent a lot of time throughout development researching and testing different features to find my preferred technologies.

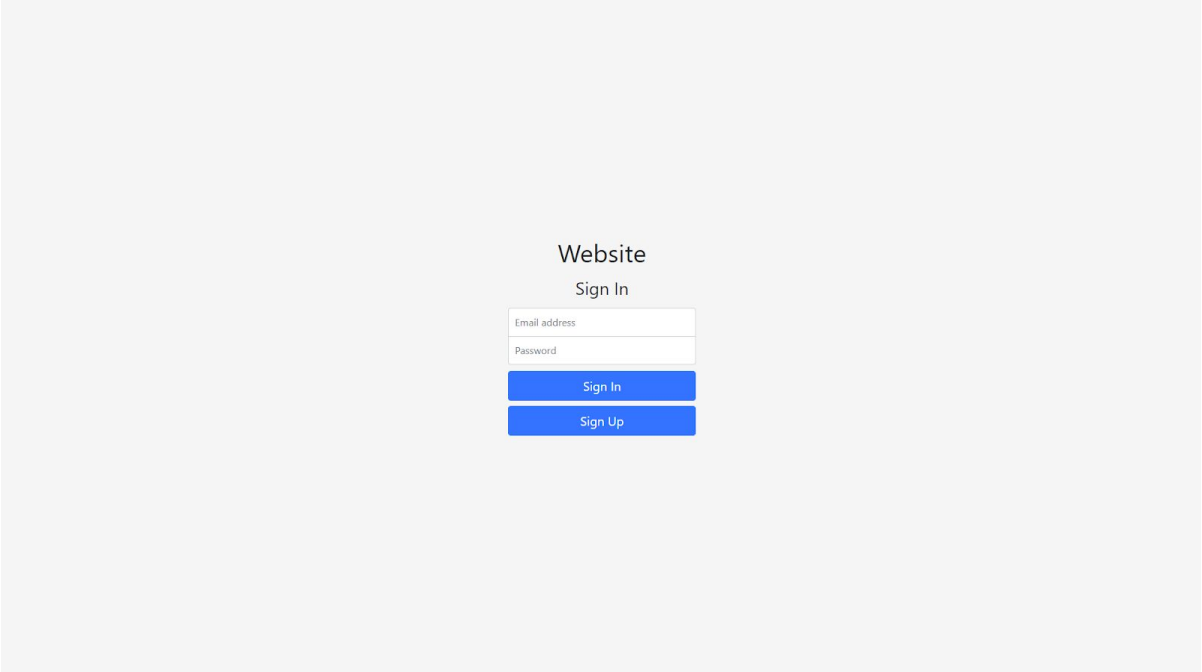
I also encountered many setbacks due to hardware problems. The system design went smoothly, but early into the implementation phase my primary development computer broke. As a result, I lost a lot of time getting the project set back up on a different system; since my first development computer was a Mac, switching systems also required moving to a different OS. Luckily, Node.js and MySQL are ready available for Mac OS, Windows, and Linux, so I was able to move the project over fairly easily.

System Operation

Login Pages

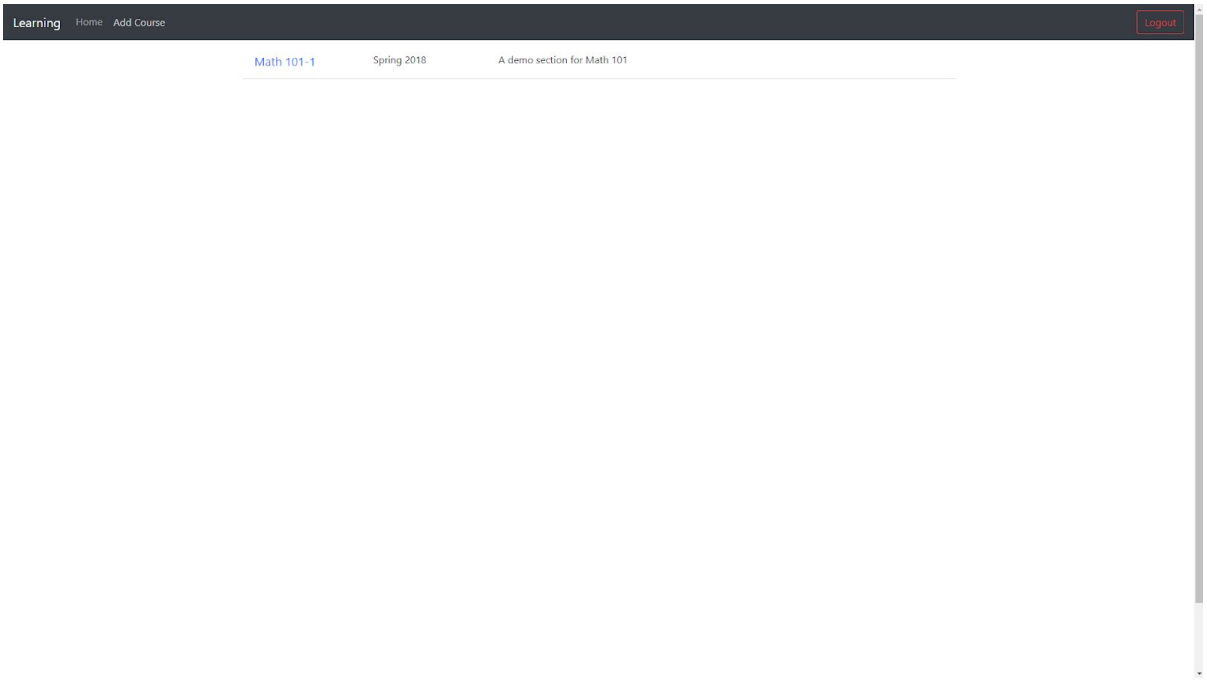
Upon visiting the website, the user is presented with the login page. The user can then either sign in with their username and password or click on ‘Sign Up’ to instead visit the sign up page and register.

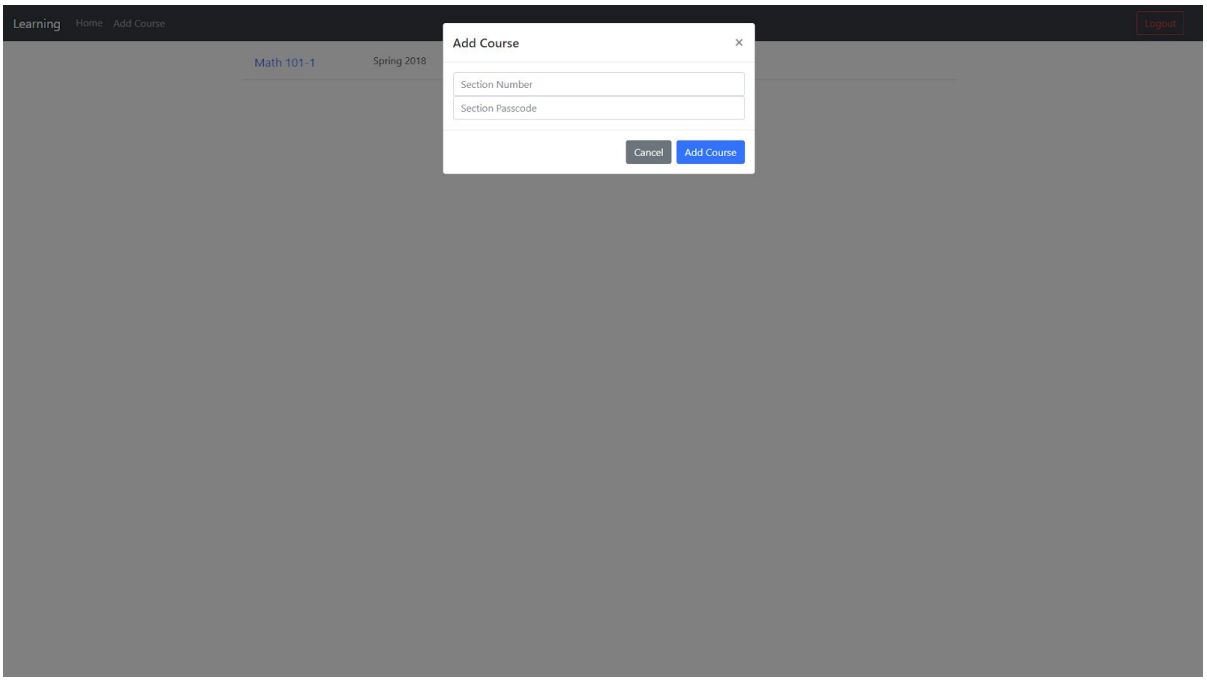




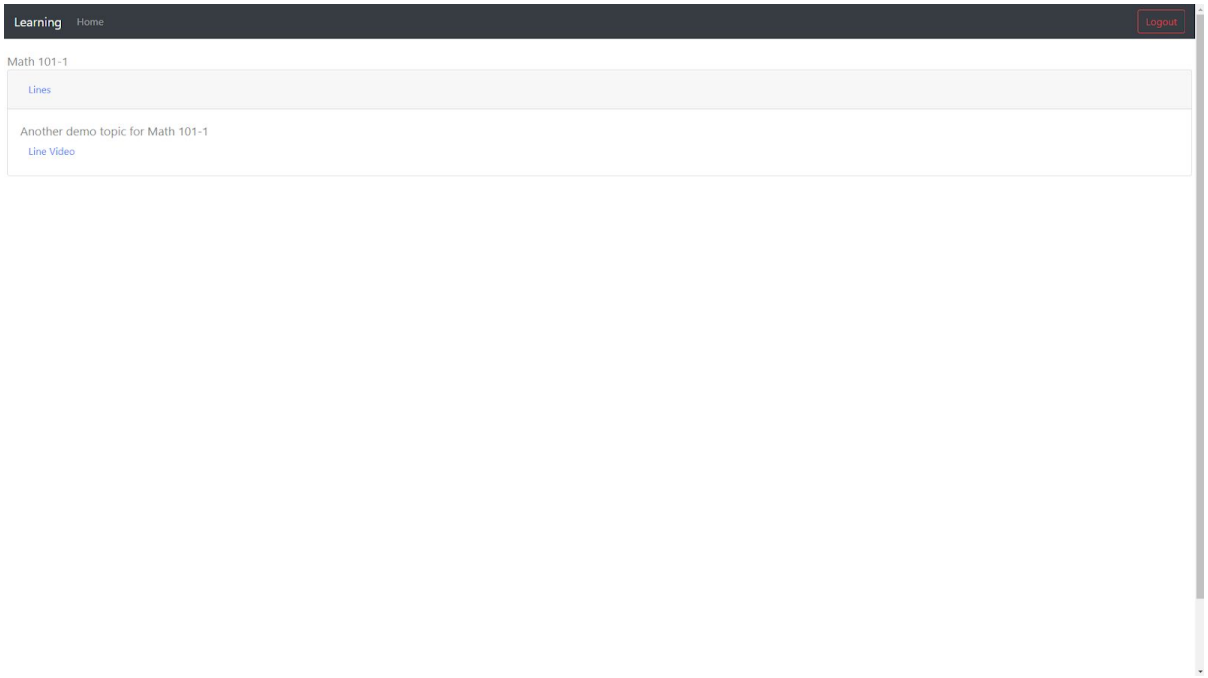
Student Pages

After logging in, students will be presented with a section list. They can either click on a section to view section resources or click ‘Add Course’ to subscribe to a new section.



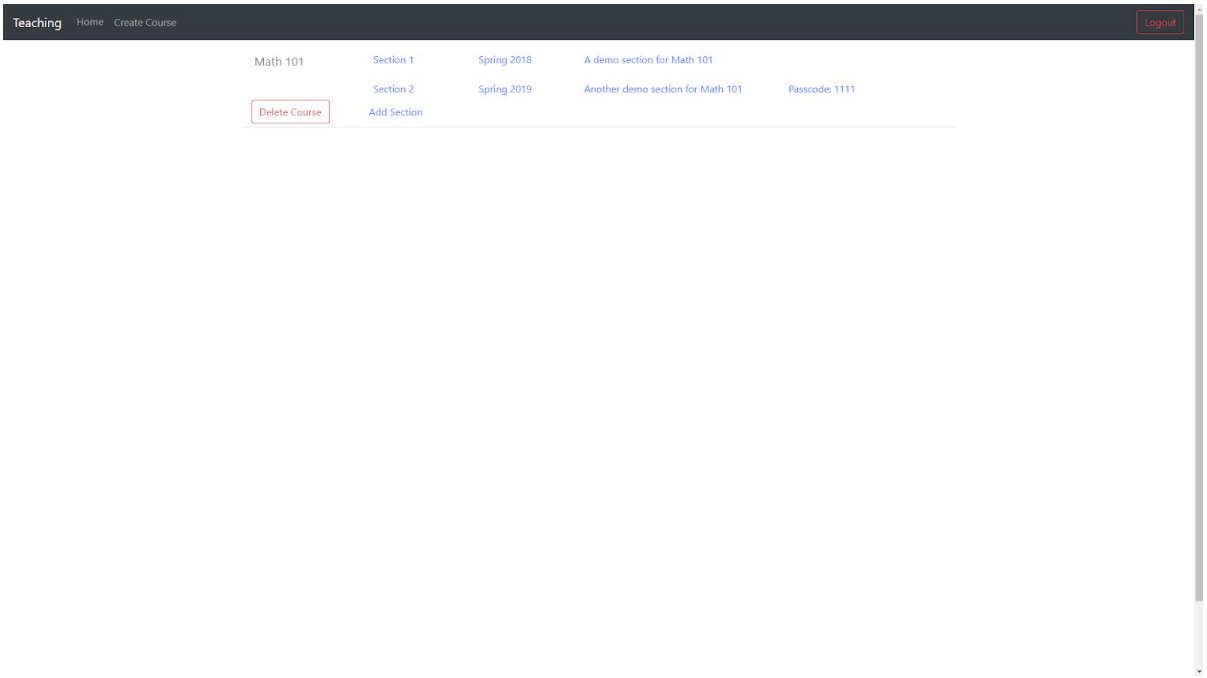


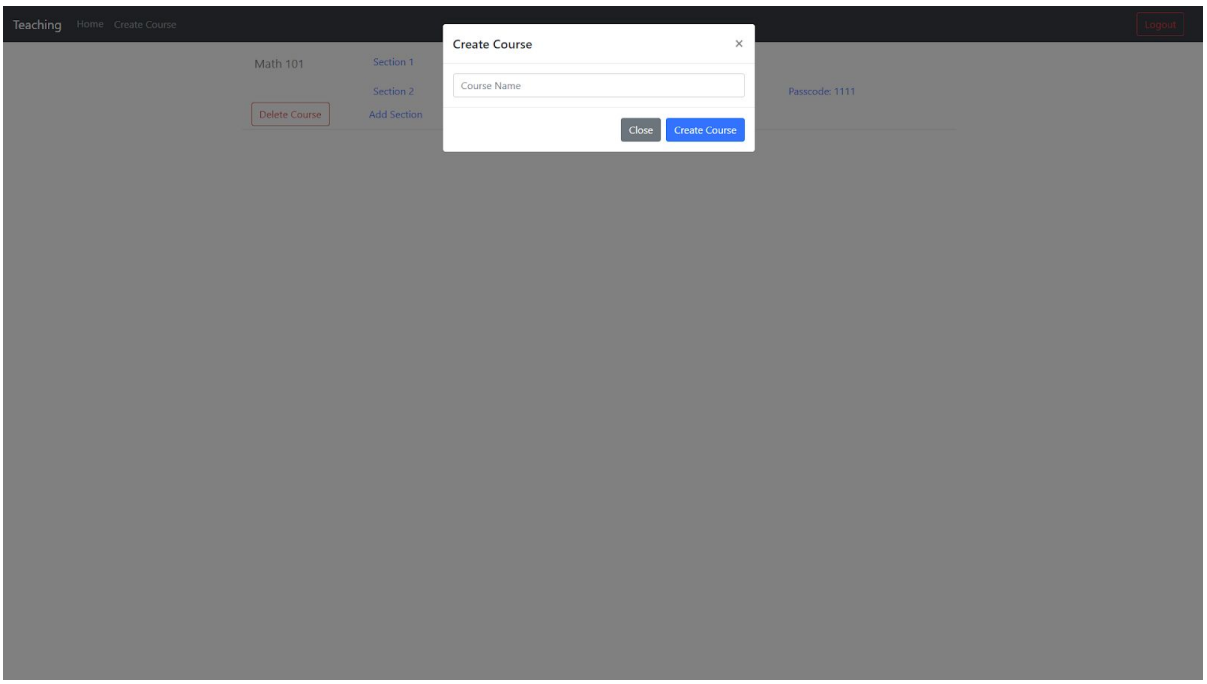
If the student clicks on a section, they will be presented with a list of all visible topics, each with a collapsible section housing all visible resources for that topic. They can click on any available resource to view it.

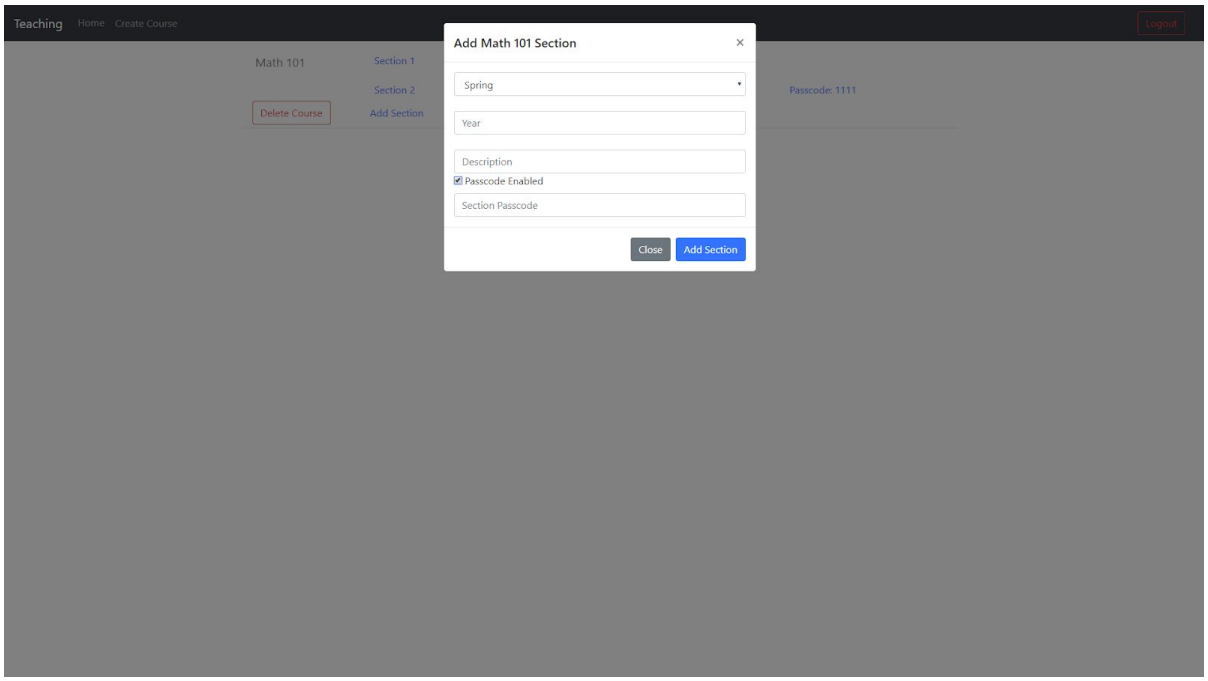


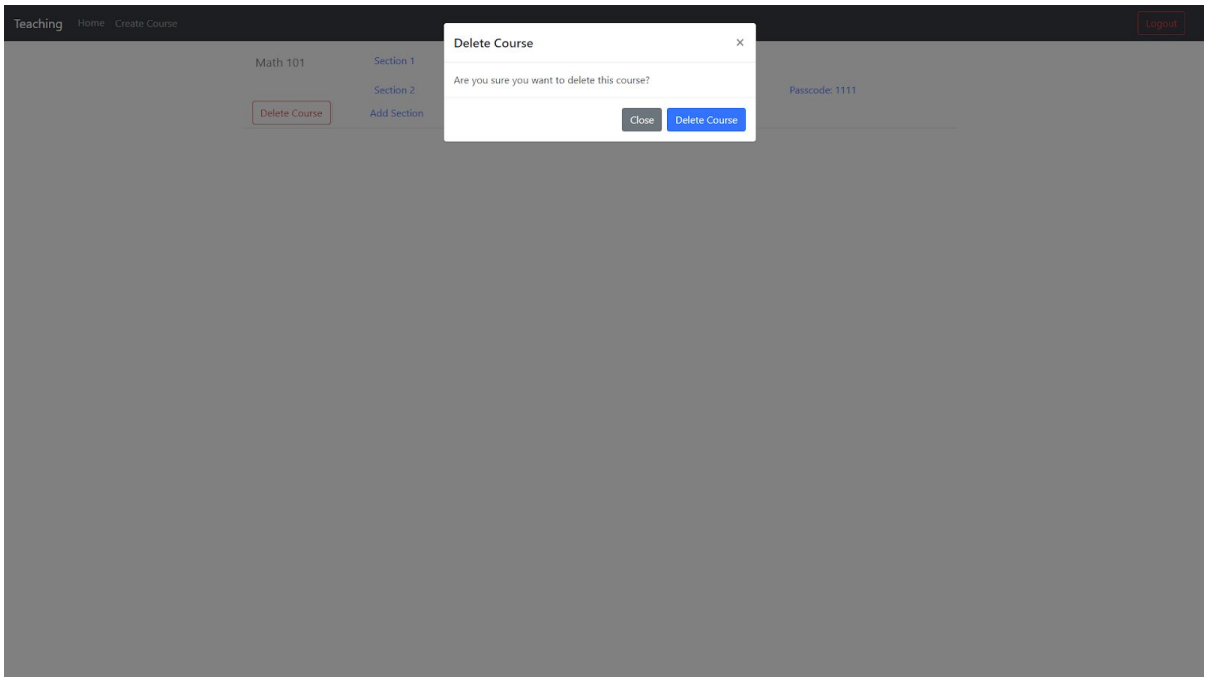
Teacher Pages

After logging in, the teacher will be presented with a course list. The teacher can click on a section to view contents, click ‘Create Course’ on the top bar to create a course, or click ‘Create Section’ under an existing course to create a new section. They can also delete a course.

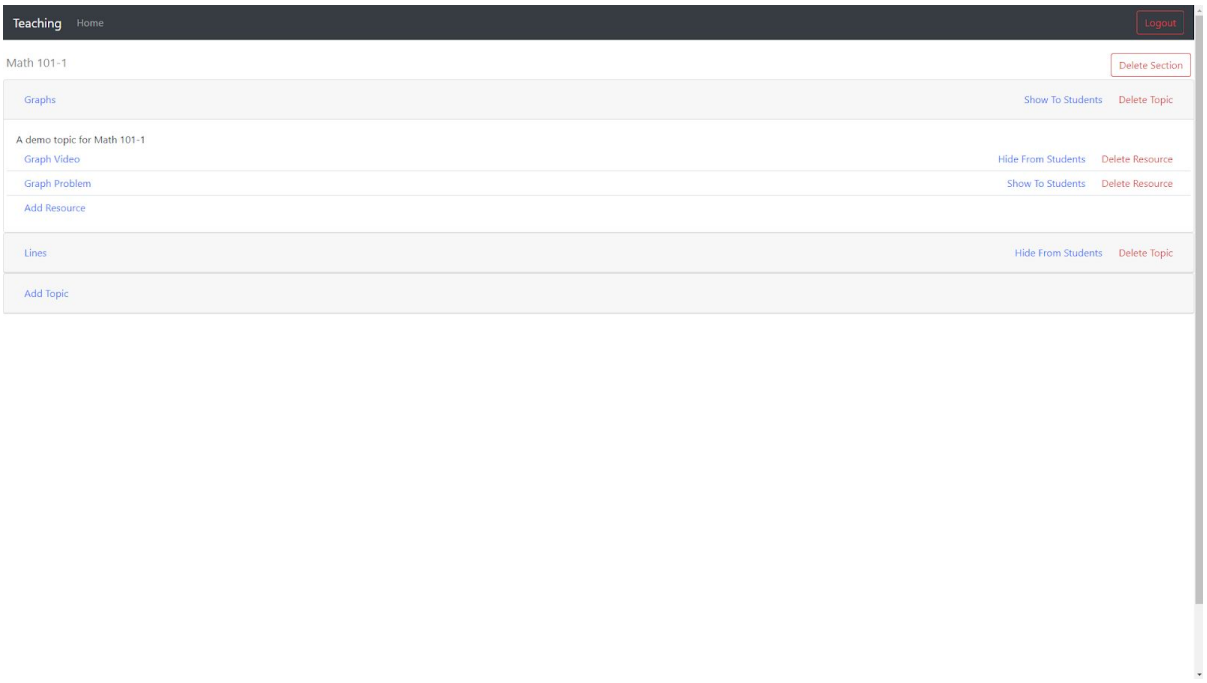


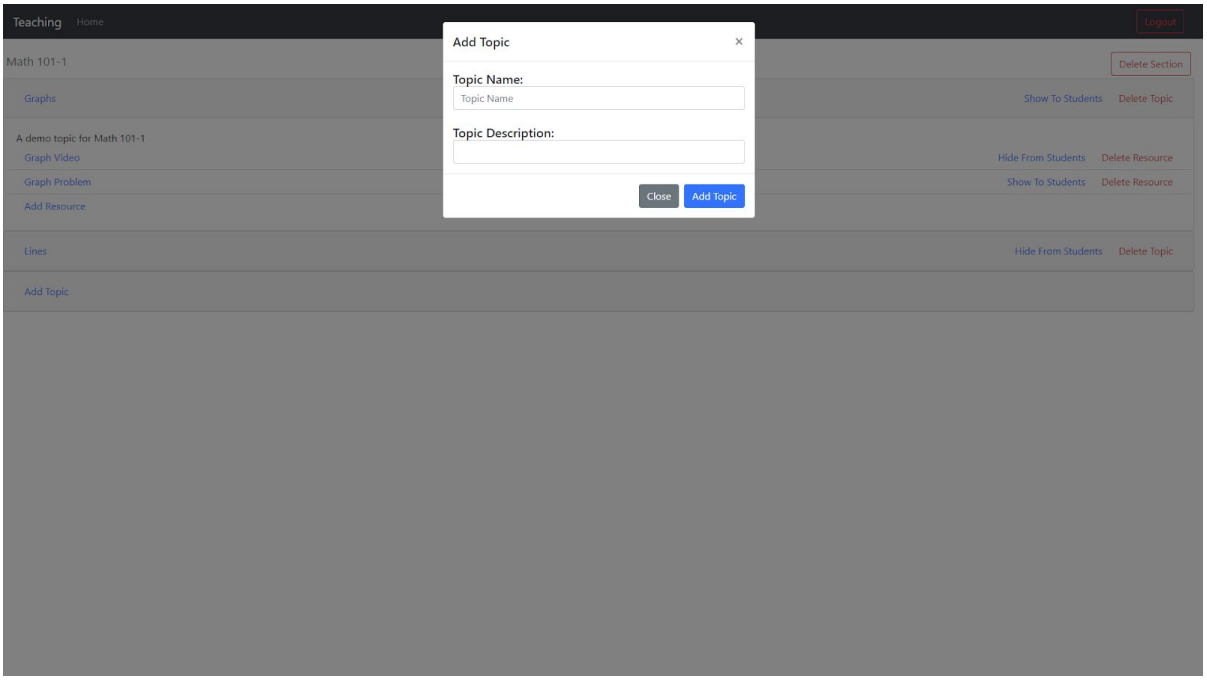


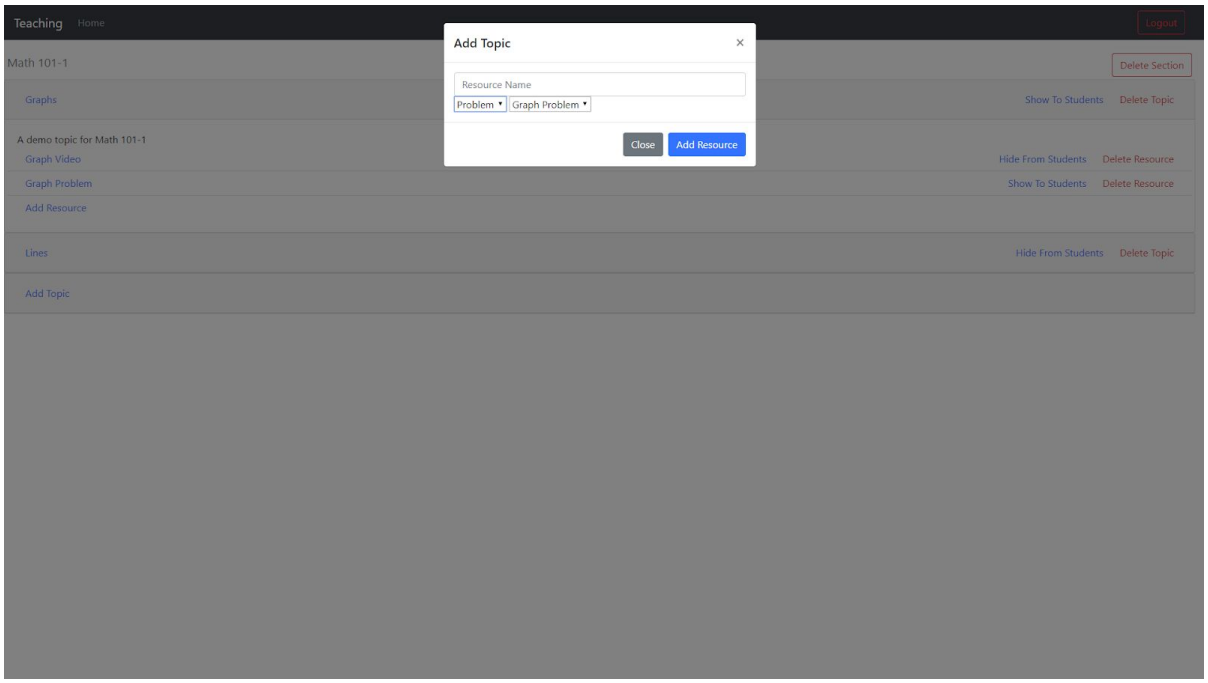

Teaching
Home
Create Course
Logout
Math 101
Section 1
Section 2
Delete Course
Add Section
Passcode: 1111
Add Math 101 Section
Spring
Year
Description
Passcode Enabled
Section Passcode
Close
Add Section

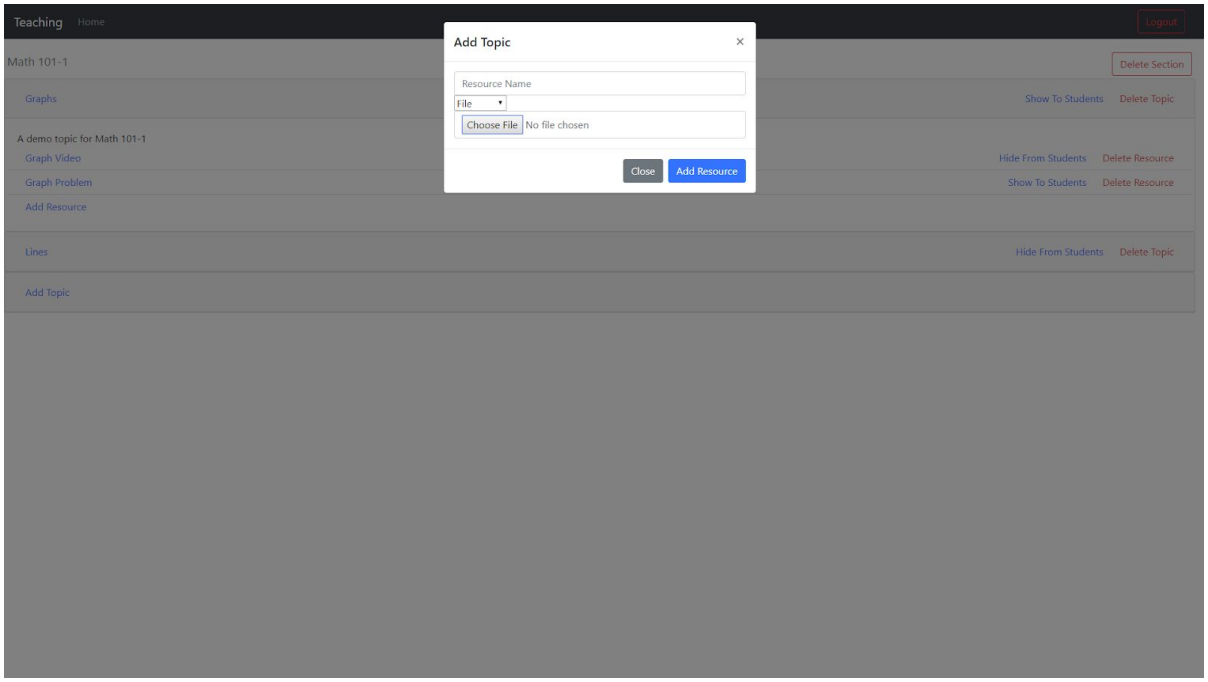

Teaching
Home
Create Course
Logout
Math 101
Section 1
Section 2
Delete Course
Add Section
Passcode: 1111
Delete Course
Are you sure you want to delete this course?
Close
Delete Course

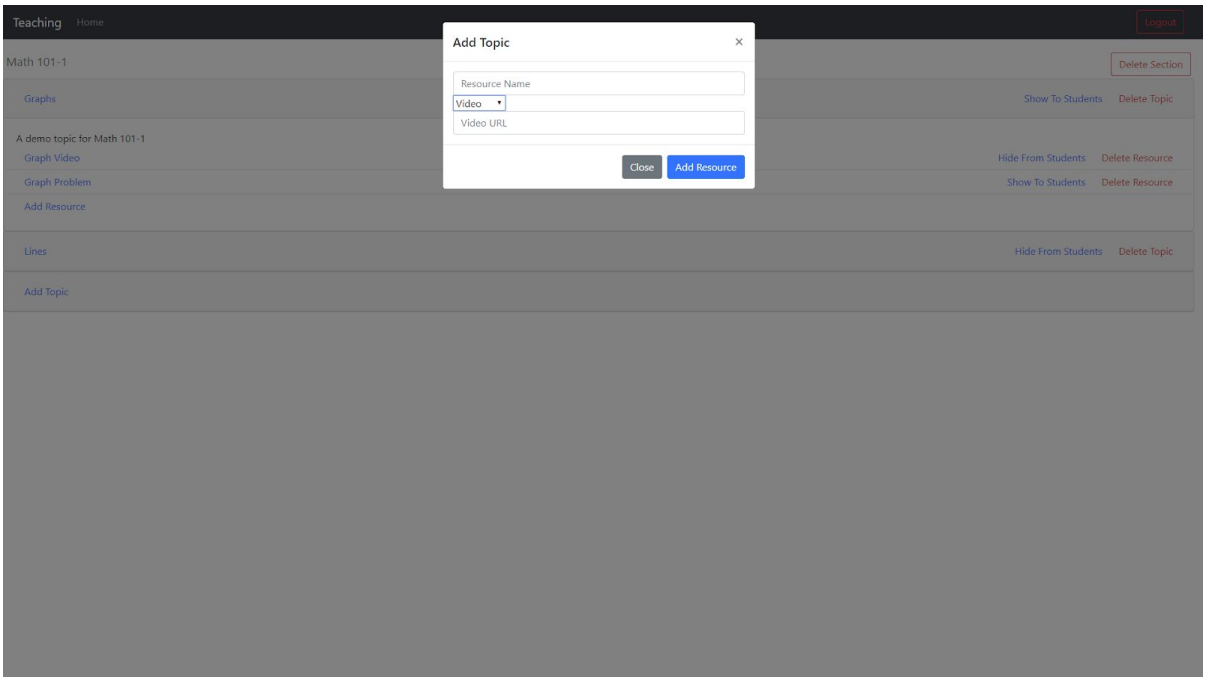
After clicking on a section, the teacher is presented with a list of section topics, with a collapsible area housing resources. The teacher can click the corresponding buttons to create a new topic, create a new resource within a topic, show or hide a topic, show or hide a resource, delete a topic, or delete a resource. They can also click on any resourced to view it.





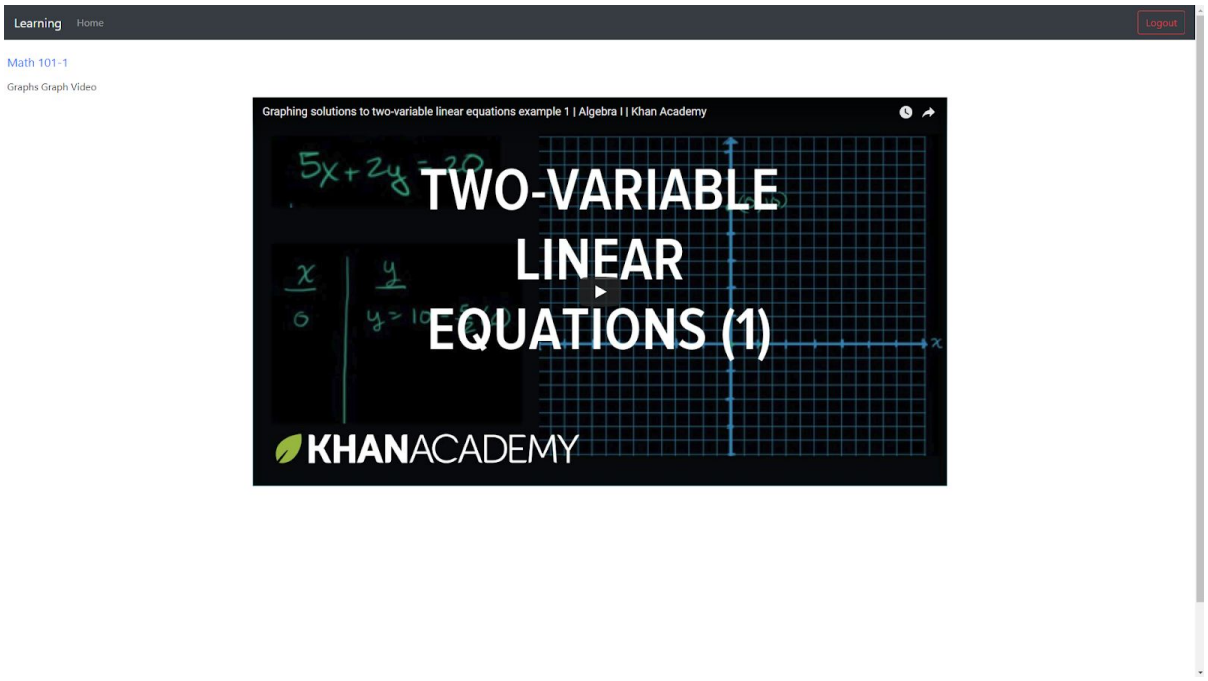

Teaching
Home
Logout
Math 101-1
Delete Section
Graphs
Show To Students
Delete Topic
A demo topic for Math 101-1
Graph Video
Hide From Students
Delete Resource
Graph Problem
Show To Students
Delete Resource
Add Resource
Lines
Hide From Students
Delete Topic
Add Topic
Add Topic
Resource Name
Problem
Graph Problem
Close
Add Resource


Teaching
Home
Logout
Math 101-1
Delete Section
Graphs
Show To Students
Delete Topic
A demo topic for Math 101-1
Graph Video
Hide From Students
Delete Resource
Graph Problem
Show To Students
Delete Resource
Add Resource
Lines
Hide From Students
Delete Topic
Add Topic
Add Topic
Resource Name
File
Choose File
No file chosen
Close
Add Resource


Teaching
Home
Logout
Math 101-1
Delete Section
Graphs
Show To Students
Delete Topic
A demo topic for Math 101-1
Graph Video
Hide From Students
Delete Resource
Graph Problem
Show To Students
Delete Resource
Add Resource
Lines
Hide From Students
Delete Topic
Add Topic
Add Topic
Resource Name
Video
Video URL
Close
Add Resource

Resource Pages

Each resource will be presented the same way to both teachers and students. Files will be sent for the browser to load or download, videos will be embedded in a webpage, and problems will be presented with a collapsible step list.



Learning Home Logout

Math 101-1

Graphs - Graph Problem

Graph the equation $y + 4 = 1(x + 2)$

Step 1: Put the equation in slope intercept form: $y=mx+b$

Step 2: Find two points by plugging 0 and 1 into the equation as x.

Step 3: Mark the two points on a graph and draw a line through them.

Learning Home Logout

Math 101-1

Graphs - Graph Problem

Graph the equation $y + 4 = 1(x + 2)$

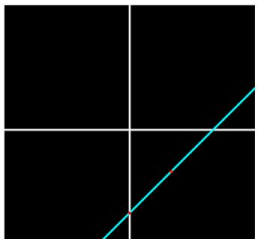
Step 1: Put the equation in slope intercept form: $y=mx+b$

$y + 4 = 1(x + 2)$
$y + 4 = 1x + 2$
$y = 1x - 2$

Step 2: Find two points by plugging 0 and 1 into the equation as x.

X	Y
0	-2
1	-1

Step 3: Mark the two points on a graph and draw a line through them.



Conclusion

This system was created to be used by teachers and students as a learning and teaching tool. Developed in Node.js, the system is lightweight and simple to deploy. It utilizes Bootstrap, jQuery, and Handlebars to present a responsive and well-designed website, and uses technologies such as Bcrypt and Passport to ensure system security. Nodemon was used both for development and production purposes, allowing future updates and patches to be deployed quickly without requiring a manual server reset.

Ultimately, the system provides a strong basic course management website, with room for future expandability.

Future Work

While the system is functional in its current form, I hope to add many features in the future:

- I intend to add more activities and resource types to be utilized by teacher, including a number of other math tools and review problems.
- I would like to add mechanisms for quickly duplicating courses and sections.
- I intend to allow topics and resources to be reordered so that they are presented to students in an organized fashion.
- I intend to add other authentication options, such as OAuth.

In the future, I would like this system to feature many of the functionalities of current course management systems, while adding features and components for previously paper-focused subjects like math.