Requirement Specifications

For

Curriculum Plan

February 21, 2021

Version 0.1

Prepared By:

Jonathan Crafton

Noah Rudolph

Jacob Paluso

Table of Contents:

Project Overview:

This project will involve the creation of a curriculum planner web application. An administrative user will be able to construct a degree plan page dynamically by having client-side course inputs stored in a database. Stored course requirements will include: Course ID, Course Name, Description, Minimum Grade Requirement, Credit Hours, Prerequisites, Corequisites, Term, Degree, and College. From the database, a plan will be generated. A normal user will be able to view these generated curriculum plans in three different perspectives: Degree, Grid, and Graph. All of this will be constructed using the MERN stack, which includes: MongoDB, Express, React.js, and Node.js.

MongoDB will serve as the database, Node.js will be used as the server-side language, Express is used as a framework for Node.js that provides features for web development, and React.js will be used as the client-side language.

Features:

Roles: User experience within the scope of separate qualifications.

Security: Authorization of administrative users within the web application.

Interface design by role: The user interface and design of the website depicted by mockups.

Capture user inputs: How and what data will be pulled from user input and sent to the database.

Databases: The databases used to store the course information and the administrator emails.

Complete all outputs: How the data will be pulled from the database to the user interface.

Roles:

There are two roles a user can take when interacting with the website. There is the normal/average user and the administrator. The difference between their interfaces will be covered in the later section. The main aspects of a normal user's interaction will consist of searching for a degree and viewing the degree through various means.

The main aspects of an administrator's interaction will consist of logging in, creating degrees, and adding courses.

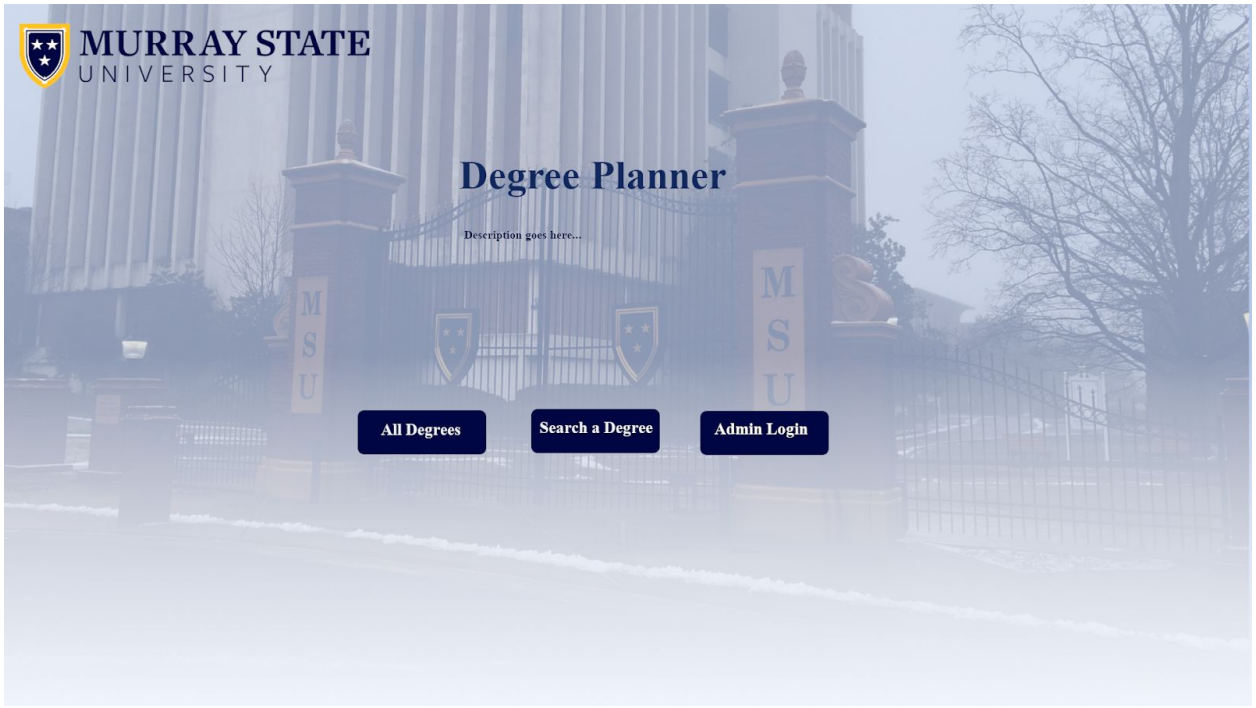
Security:

The database will not store every administrator's custom login information. Instead, login and authorization will be handled through the use of the OAuth 2.0 protocol. Since all Murray State students and personnel are given a Google email address under the murraystate.edu name, all students and personnel can have their identification verified through the use of the Google API. Through given permissions, Google can send a token of credentials that can be compared to what is stored in the database. However, this still means that the database will have to store the email of verified administrators.

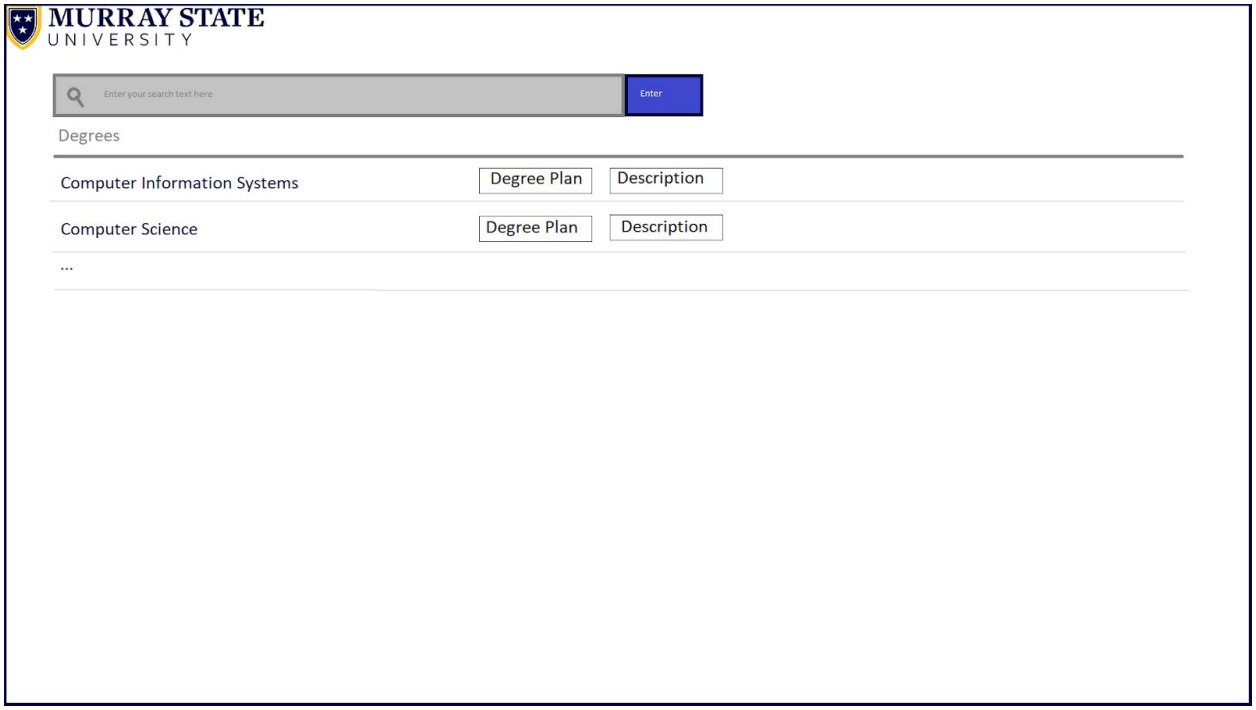
OAuth 2.0 will be implemented by requesting the user's email address through the userinfo.email scope. Once a token has been sent back, the email will be requested from Google and tested against the emails stored in the database. If the test isn't passed, the user will be informed, if it is passed, the user will be sent to the next page.

Interface Design by Role:

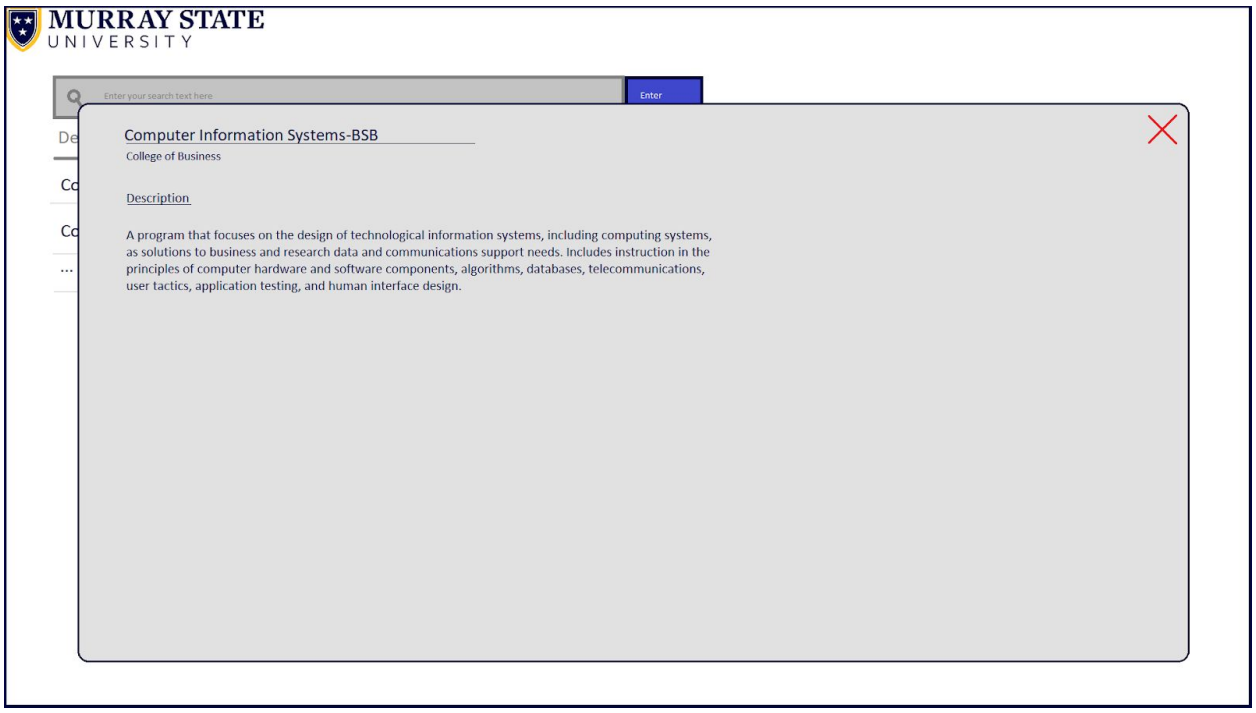
Landing/Welcome Page -The page that all users will arrive at initially when they go to the website.



Degree List Page- Lists all the degrees the user can choose from and view degree plans of.

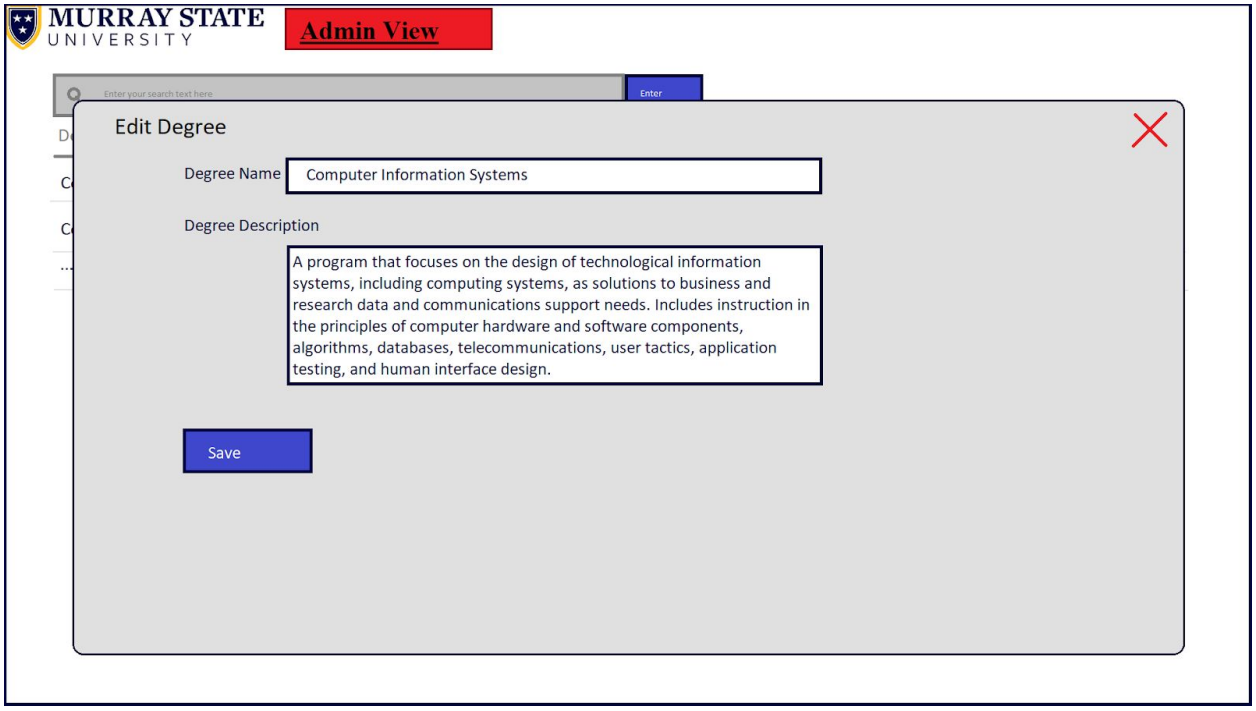


Degree Description Popup- Tells the user some general info about the degree itself.

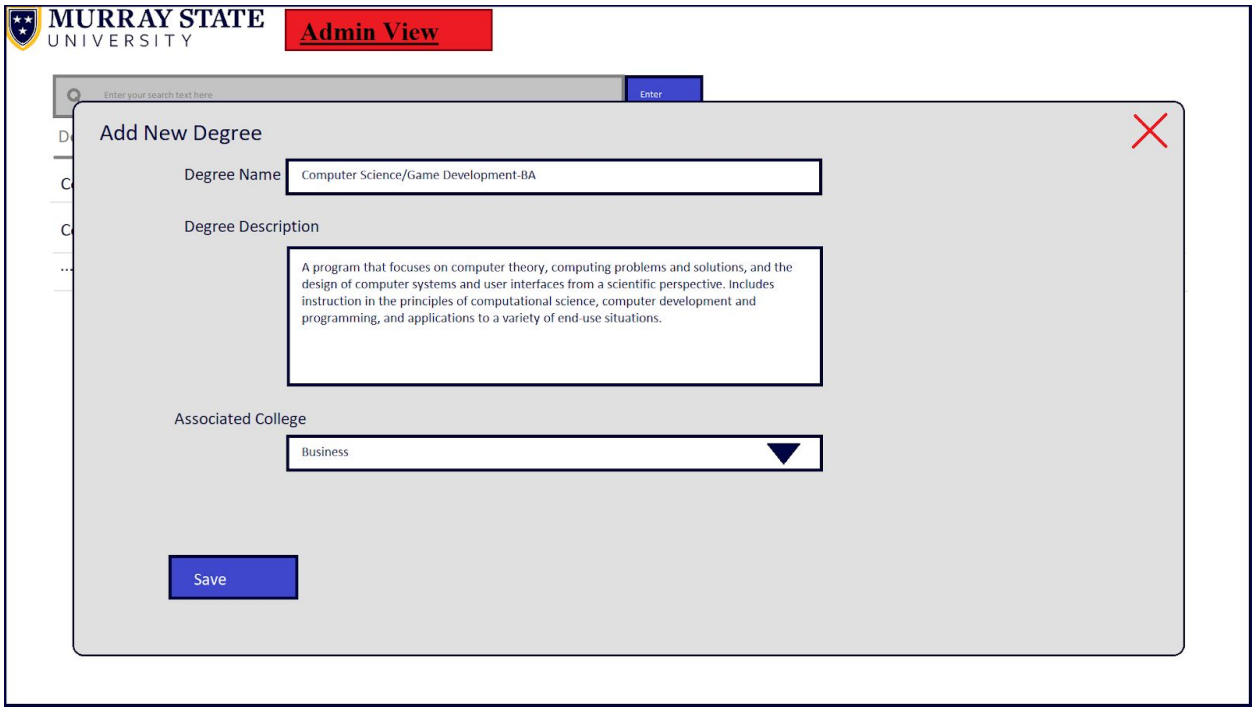


Admin Degrees List Page- List of all the degrees where the administrator can edit, delete, and add new degrees on top of the functionality of the Degrees List Page for normal users.

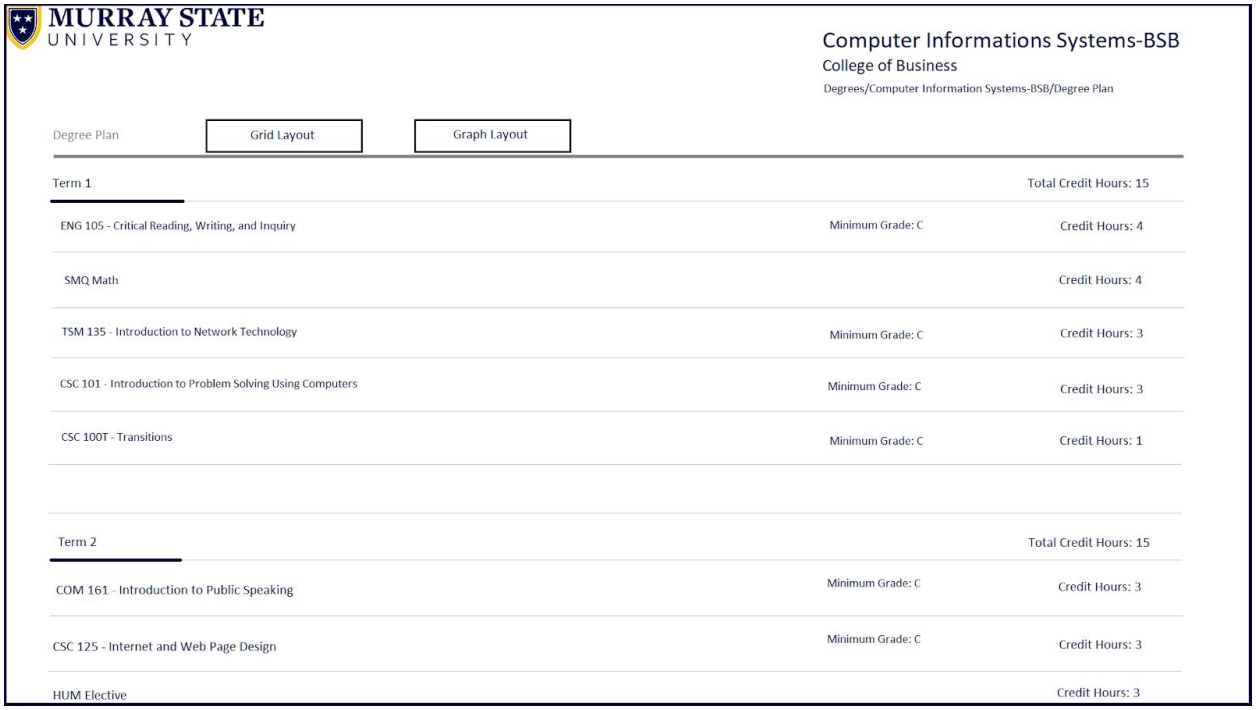
Edit Degree Popup - Where an administrator can edit the degree name and description.



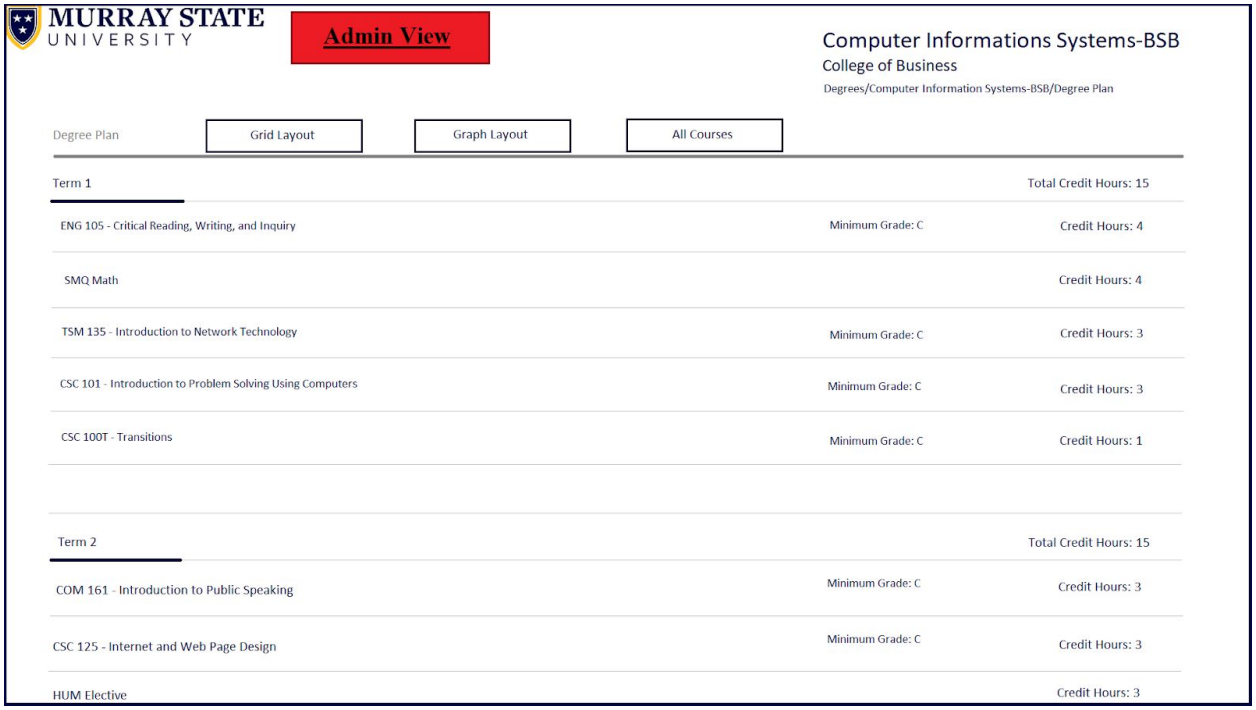
Add Degree Popup - Where an administrator can create a new degree by inserting a name, description and then selecting a college from a drop down.



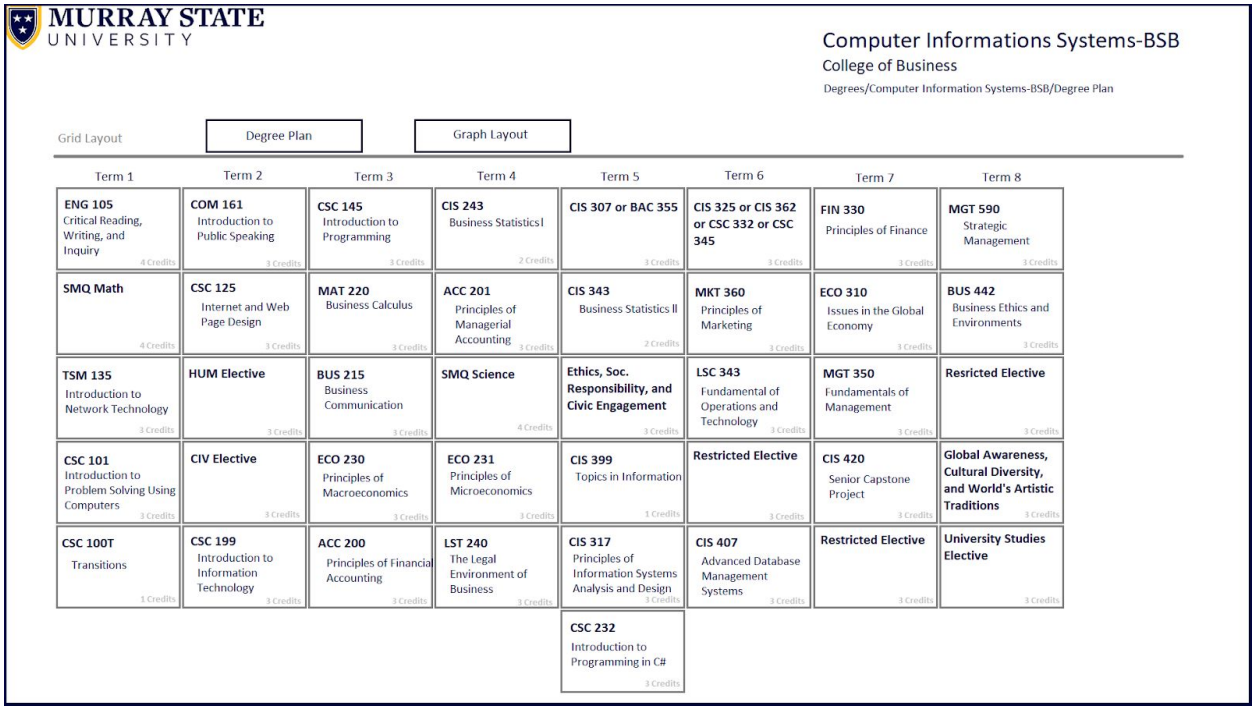
Degree Plan Page- Where the user can go to view the list of courses he or she would need to take each term to graduate in four years.

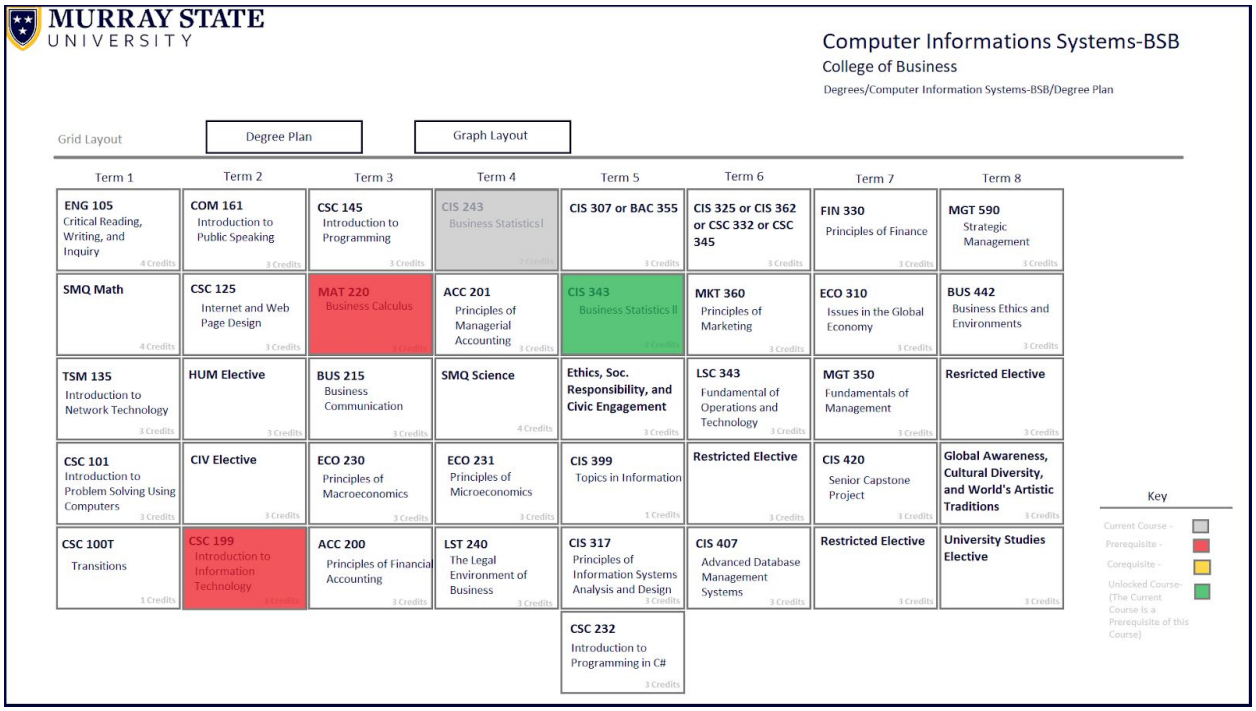


Admin Degree Plan Page - Has all the same functionality of Degree Plan Page for normal users, but there is an additional button for the All Courses Page.

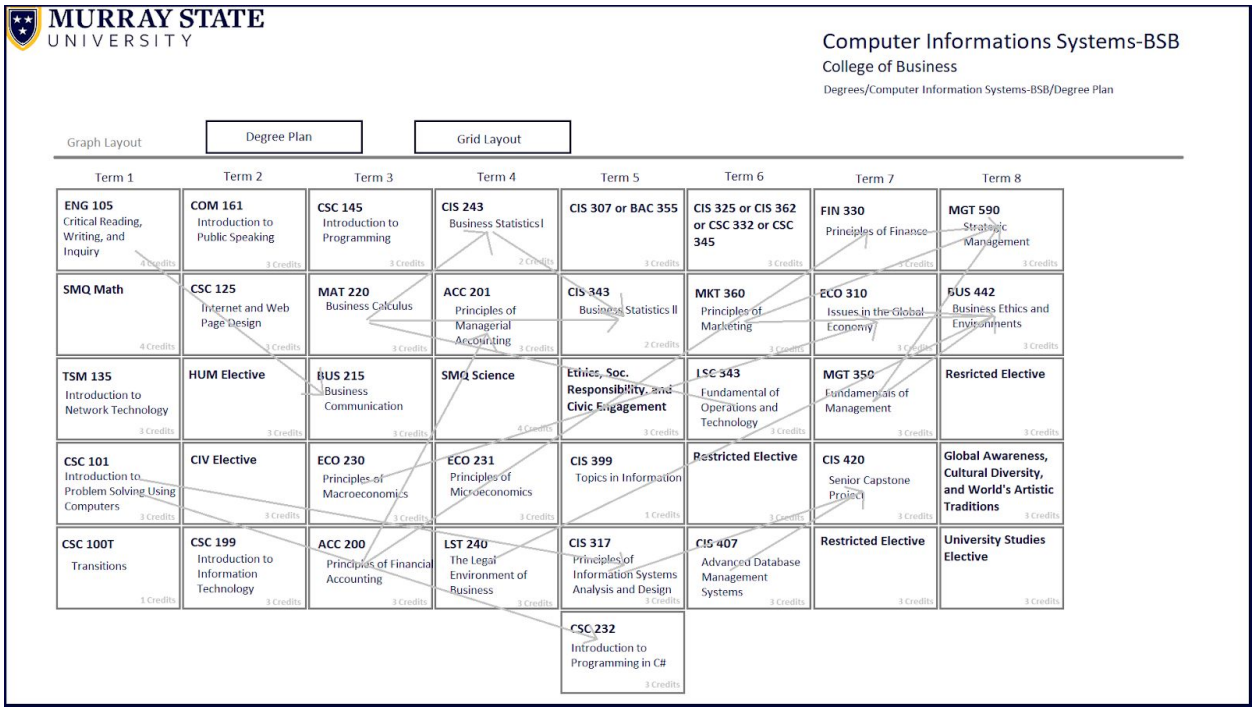


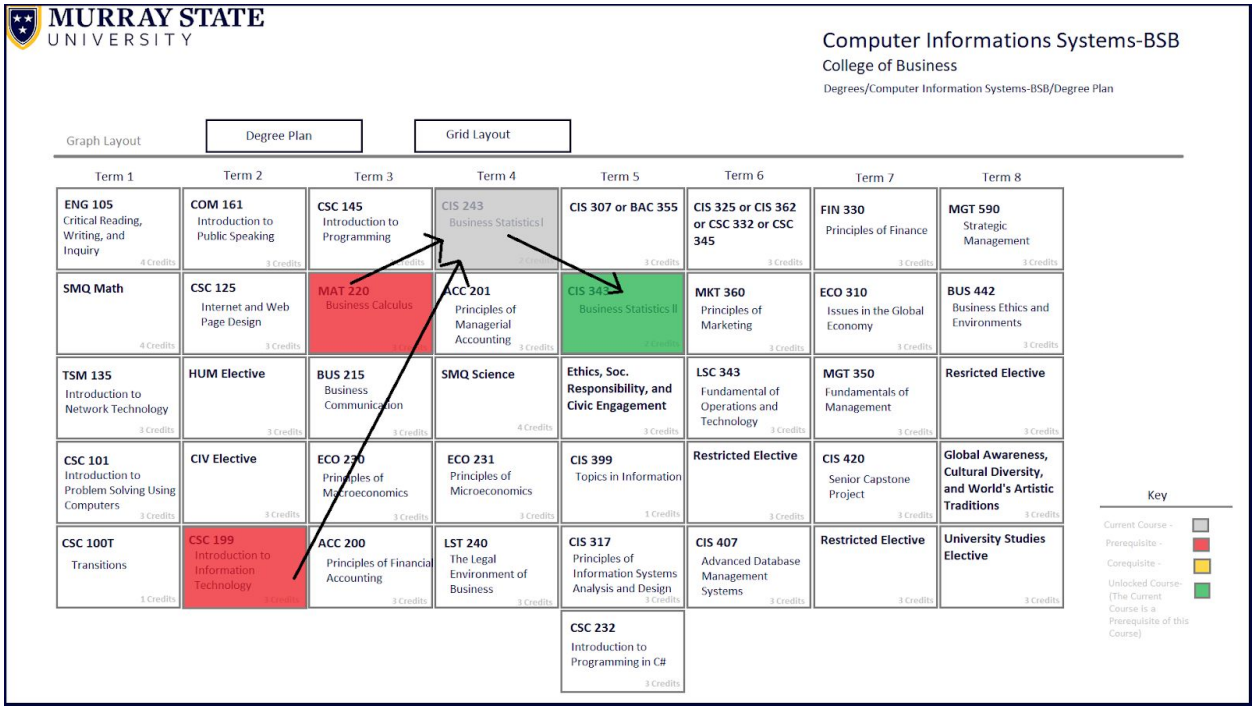
Grid Layout Page - Another view of all the courses by term he or she would need to take and where the user can view the prerequisites and corequisites of those courses. If a user hovers over a course it will show the prerequisite, corequisites, and courses unblocked from this course via different colors as well as a key at the bottom right depicting the meanings of each color.



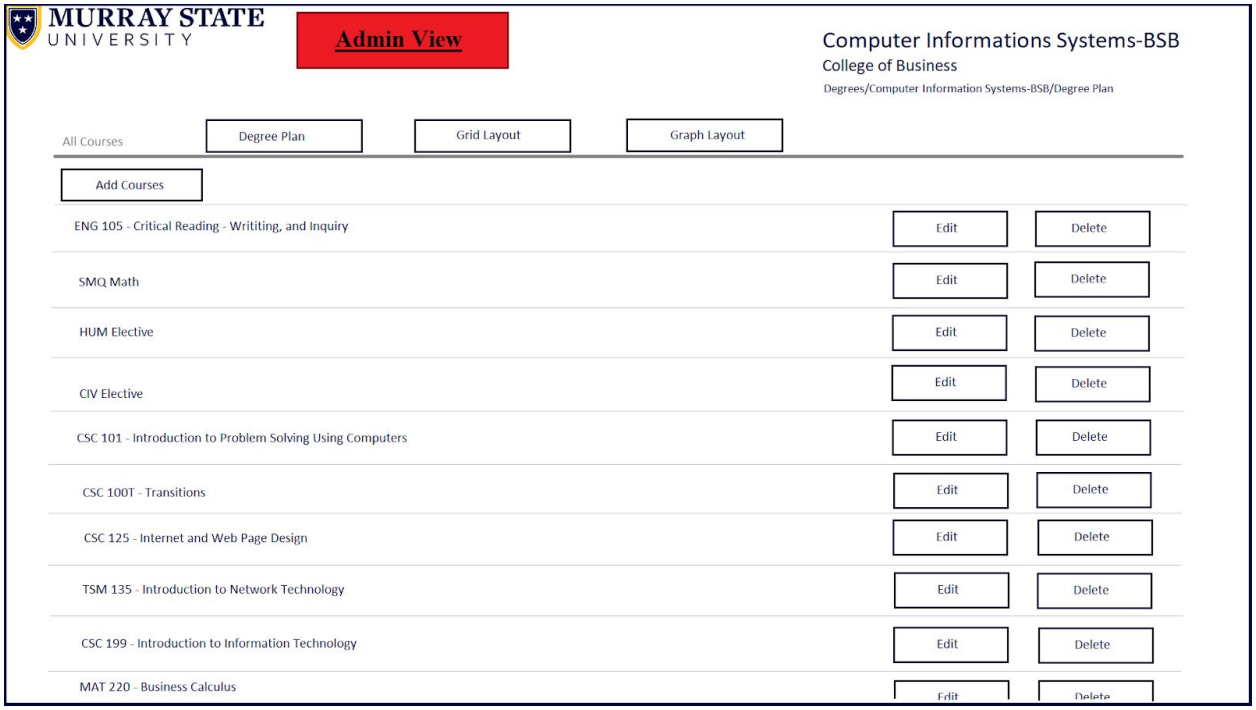


Graph Layout Page - Another view of all the courses by term he or she would need to take and where the user can view the prerequisites and corequisites of those courses. This is similar to the Grid Layout where this layout shows the same thing, but with arrows denoting the order in which classes are taken. Like the Grid layout, if a user hovers over a course it will show the prerequisite, corequisites, and courses unblocked from this course via different colors as well as a key at the bottom right depicting the meanings of each color.

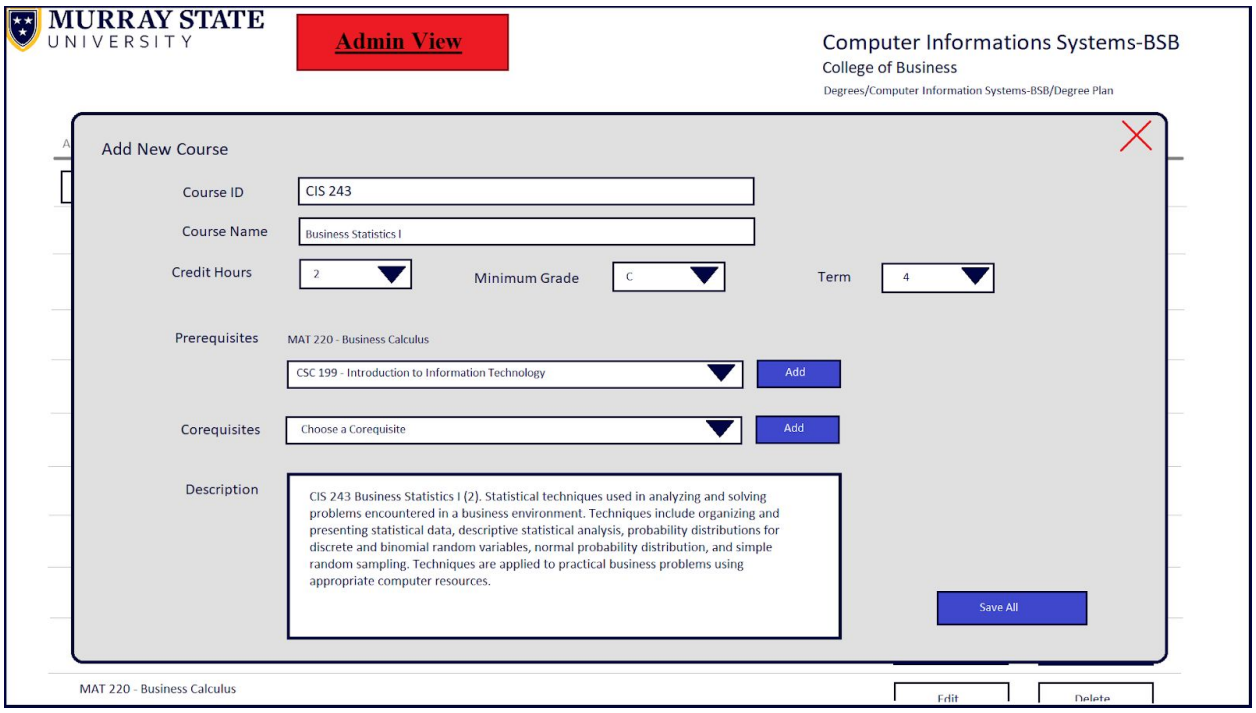




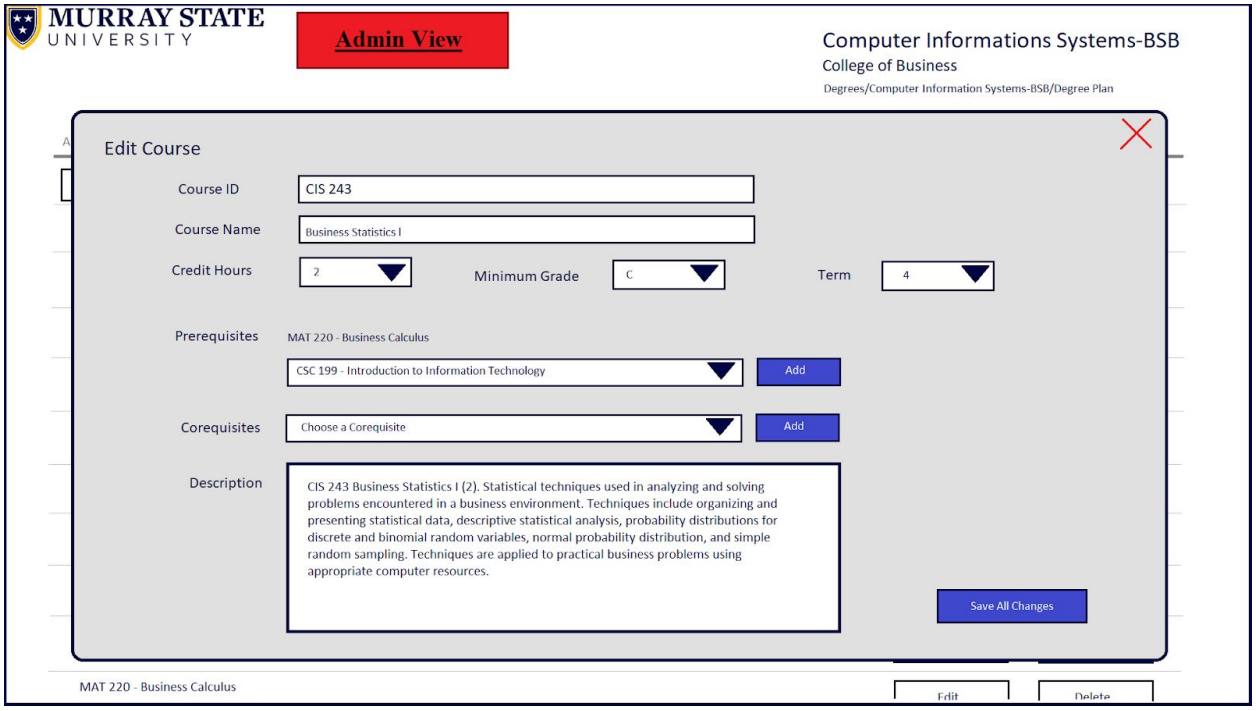
All Courses Page- Shows all the courses required for the degree. Here an admin can add, edit, and delete courses. (Admin only feature).



Add Course Popup- Allows admins to create a new course that is within the current degree.



Edit Course Popup- Allows admins to edit data of pre-existing courses within a degree.



Capture user inputs:

On the website, the only source of user input will be from Admins who choose to add, edit, or delete data. This data includes Colleges, Degrees, and Courses. The way it will work is React.js is the front end that creates the fields in which the data is entered. From there Node.js acts as a middle man and pulls the data from the fields and confirms that the data is a valid input. Afterward, Node.js sends off the data to the connected database.

Database:

Using MongoDB, we are going to create a collection of Course documents. A course document will include the following field:

- Course ID: The ID of the course (i.e. CSC 145)
- Course Name (i.e. Intro to Programming)
- Description: a brief description of the course
- Minimum Grade Requirement
- Credit Hours: the number of credit hours the course is worth
- Prerequisites: All of the courses required to take before the course
- Corequisites: Any courses required to take with the course
- Term: Which term this course will be taken
- Degree: The degree this course is for
- College: What college the course is a part of.

For the Administrator role, we are also going to create a collection of documents storing Admin info. An administrator document will include the following fields:

- Name: Name of the administrator
- Email: Administrator's email address

Complete All Desired Outputs:

Our User Interface will be created using React.js. Whenever an operation is performed to pull information from the MongoDB database, such as clicking on a desired course, Node.js will send the information to the User Interface. In Grid View, when you highlight a course, the database will be scanned for prerequisites and corequisites to be highlighted. In the Graph View, this same method will be used to dynamically create arrows to show the connections between courses.