**Project Title: GradTrak**

CMPS 4928: Senior Project II

Group Report and Code Documentation: Justin Ulloa, Tristan Bock, Shadi Abdul Razzak, and Haylee Allen

Date: May 15th, 2023

Table of Contents

[Project Overview 4](#_Toc135086210)

[Project Management 4](#_Toc135086211)

[Changes to Project 4](#_Toc135086212)

[Implementation 4](#_Toc135086213)

[Front-End 4](#_Toc135086214)

[Index Page 5](#_Toc135086215)

[Figure 1. *Index Page (PC Layout)* 5](#_Toc135086216)

[Figure 1.2. *First Time User* 6](#_Toc135086217)

[Figure 1.3. *NetID Link* 6](#_Toc135086218)

[Code (Index Page) 8](#_Toc135086219)

[Figure 1.5. *Code Snippet of Index Page* 8](#_Toc135086220)

[Design (Index Page) 9](#_Toc135086221)

[Figure 1.6. *Code Snippet of Background Design (Index Page)* 9](#_Toc135086222)

[Figure 1.7. *Code Snippet of Header and Container* 9](#_Toc135086223)

[Figure 1.8. *Code Snippet of Buttons and Footer* 10](#_Toc135086224)

[Login Page 10](#_Toc135086225)

[Figure 1.9. *Login Page* 11](#_Toc135086226)

[Student Page 11](#_Toc135086227)

[Figure 2. *Student Page (PC Layout)* 11](#_Toc135086228)

[Figure 2.1. *Course Requirements* 12](#_Toc135086229)

[Figure 2.2. *Career Resources* 13](#_Toc135086230)

[Code (Student Page) 14](#_Toc135086231)

[Figure 2.3. *Code Snippet of Navigation Bar* 14](#_Toc135086232)

[Figure 2.4. *Code Snippet of Progress Bar* 14](#_Toc135086233)

[Figure 2.5. Code Snippet of Student Information Structure 15](#_Toc135086234)

[Figure 2.6. *Code Snippet of Accordion and Collapsible Elements* 16](#_Toc135086235)

[Figure 2.7. *Code Snippet of Table Structure in Panels* 17](#_Toc135086236)

[Figure 2.8. *Code Snippet of Career Resources Section* 18](#_Toc135086237)

[About Page 18](#_Toc135086238)

[Figure 3. *About Us* 19](#_Toc135086239)

[Figure 3.1. *Developer Team* 19](#_Toc135086240)

[Figure 3.2. *Featurette Section* 20](#_Toc135086241)

[Code (About Page) 20](#_Toc135086242)

[Figure 3.3 *Code Snippet of Developer Team Structure* 21](#_Toc135086243)

[Figure 3.4. *Code Snippet of Featurette Section* 21](#_Toc135086244)

[Back-End 21](#_Toc135086245)

[Table/ER Diagram Design 22](#_Toc135086246)

[Figure 3.5. *Structure of StudentInfo Table* 22](#_Toc135086247)

[Figure 3.6. *Structure of GeneralEducation Table* 23](#_Toc135086248)

[Figure 3.7. *Structure of the MajorRequirements Table* 23](#_Toc135086249)

[Database Implementation 24](#_Toc135086250)

[Figure 3.8. *Finalized ER Diagram* 24](#_Toc135086251)

[Figure 3.9. *Create Table Statement for StudentInfo Table* 25](#_Toc135086252)

[Figure 4*. Insert Statement Query for StudentInfo* 25](#_Toc135086253)

[Figure 4.1. *Database Table of StudentInfo After Insert Query* 25](#_Toc135086254)

[Figure 4.2. *Insert Statement Into the GeneralEducation Table* 26](#_Toc135086255)

[Figure 4.3. *Database Table of GeneralEducation After Insert Query* 26](#_Toc135086256)

[Connection of Database to Website 26](#_Toc135086257)

[Figure 4.4. *PHP Code to establish connection to Database* 27](#_Toc135086258)

[Figure 4.5. *Establish Connection to the Database outside of db.php* 28](#_Toc135086259)

[Figure 4.6. *Code Needed to Login* 29](#_Toc135086260)

[Figure 4.7. *Structure of Logout Page* 30](#_Toc135086261)

[Figure 4.8. *Student Login Verification Snippet* 30](#_Toc135086262)

[Figure 4.9. *Basic Structure to Connect Database Tables to Website* 31](#_Toc135086263)

[Figure 5. *How the DistinctGenMajorCombo View Was Created* 32](#_Toc135086264)

[Figure 5.1. *How the CompUnitsPercent View Was Created* 33](#_Toc135086265)

[Figure 5.2. *Table for CompUnitsPercent* 33](#_Toc135086266)

[Figure 5.3. *How the AutoIncStudentInfo View Was Created* 34](#_Toc135086267)

[Figure 5.4. *Table for AutoIncStudentInfo* 34](#_Toc135086268)

[Figure 5.5. *Code to Connect Database to Progress Bar* 34](#_Toc135086269)

[Conclusion 34](#_Toc135086270)

[Appendix 35](#_Toc135086271)

[Works Cited 36](#_Toc135086272)

# Project Overview

This project came across from a student viewpoint to which we all related to and felt comfortable working with as we see this to be an improvement for the future and aids in expanding our knowledge of building a comprehensive program with a working database. GradTrak is aimed at all types of students admitted to universities anywhere, and since we wished for a system like this to be implemented into our school, we aimed our project to match the CSUB colors to show how it would look like if it was seriously implemented. This also creates a template for other schools to experiment with as well. The expected user base is all students active into the school’s registrar, as it pulls user information and login through there for program login and functionality. Major project features include an instant view of a progress bar towards finishing a specific degree, as well as updated GPA calculations. It also includes easy access into class history information and provides recommendations for next semester.

# Project Management

As a team, we wanted to split work evenly and especially towards each member’s experience in what they prefer/excel. Splitting the work proved to be rather easy with a such a comprehensive team. Justin Ulloa took on the role of designing and developing the front-end of the application and assisted with database design as well. Shadi Abdul Razzak aided in database creating and testing and inspired some of the application design. Tristan Bock took on a flexible role, with a focus on implementing the features of the database and front-end communication, and Haylee Allen assisted with back-end database implementation and design to the application itself. Our workplan consisted of ten different stride milestones we aimed to finish goals in a set duration of weeks, also with set teams.

# Changes to Project

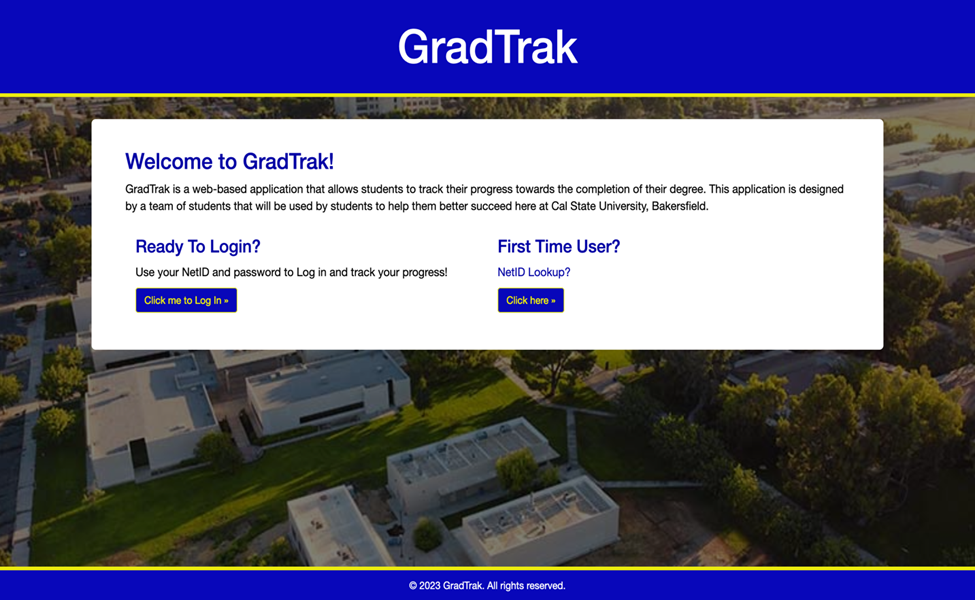
A major change we made throughout working on our project was that we decided to stick to a student side instead of a student/advisor. The sole purpose for this reason is because we discussed that it may be too complicated to include an advisor section, as it only had a goal of communicating between the student and advisor, when emails do the job quite simply good enough. It also involves creating a whole different database set that requires a different advisor log in and information, which felt extensive for a small task. What we planned for completion is working to a respectable extent, even though our target time lengths planned on each goal didn’t always comply with our time length, we still carried on and ask each other for help.

# Implementation

## Front-End

In this section, we will be going into detail about how each page for our application was created in terms of the thought process behind each page, providing what features were used, and describing how those features were applied into our project. It will include relevant images of the final product of our application as well as code snippets being described into detail showing how our product was created.

## Index Page

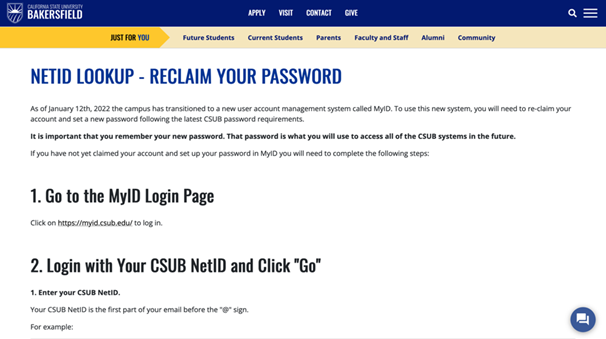


### Figure 1. *Index Page (PC Layout)*

Figure 1 is our completed design of the index page of the application. It was designed using Bootstrap[1], HTML, & CSS. As you can see, the page displays the name of our application, GradTrak, a section welcoming the users, providing a brief description of the intended purpose of our application, a ready to login in section telling the user what information they need to use our application, as well as a first-time user section that provides the user with a button that navigates them to a page that helps them understand what information is needed and what steps to take if they do know they’re information to access the application that can be seen in Figure 1.2. There is also a secondary link that takes the user to our CSUB website to a specific page where students can look up their NetID, if they are not familiar with it and how to reclaim their password that can be seen in Figure 1.3. The background for the page is an image of the CSUB campus so that the page follows the theme a common university website page. Lastly, the page contains a footer that displays a copyright message.



### Figure 1.2. *First Time User*



### Figure 1.3. *NetID Link*

Figure 1.4 showcases our application from a mobile viewpoint. We wanted to ensure that our web application could be useable using any web browser and any devices such as phones, tablets, etc. As is shown, our web page is responsive and fills its’ contents depending on the device being used. This is something we often discussed about and had to learn the best way to approach this task.

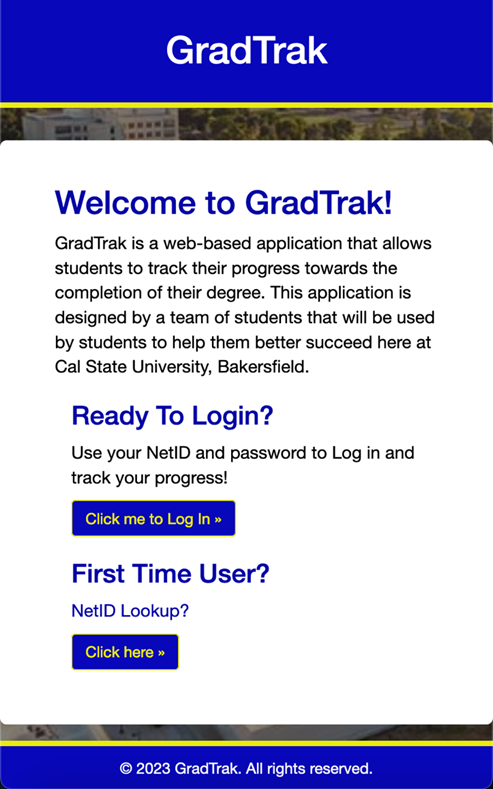
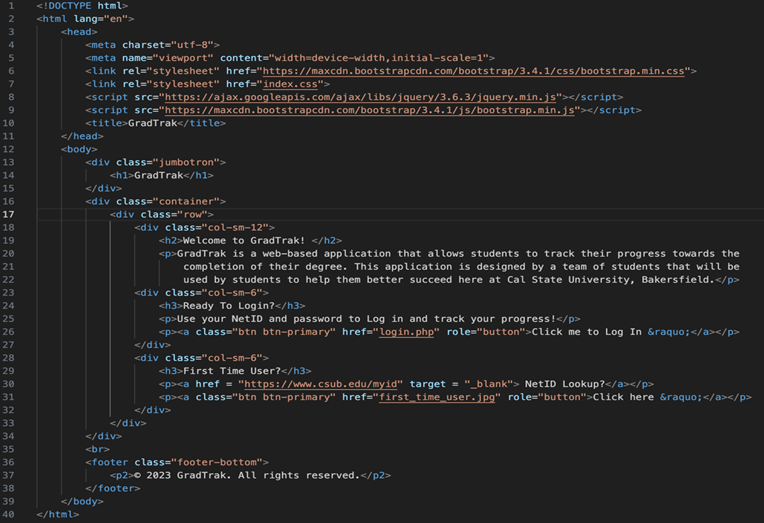


Figure 1.4. *Index Page (Mobile-View)*

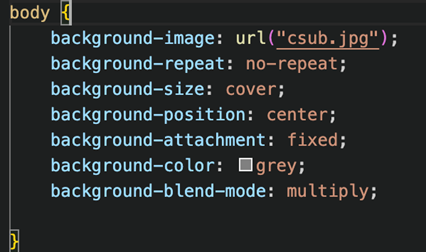
## Code (Index Page)



### Figure 1.5. *Code Snippet of Index Page*

As we mentioned earlier, the index page was created using Bootstrap, HTML, and CSS. Justin made sure to include a Bootstrap stylesheet and script, and a jQuery script to format the content to our liking. As is shown, the header of our index page was created defining a div class “jumbotron” that Bootstrap automatically formats to make the header look nice and be positioned at the top of the page. He also defined a separate class “container” that contains 3 separate columns that automatically separates the 3 different sections, “Welcome to GradTrak”, “Ready to Login?”, and “First Time User”. Bootstrap came in handy here, separating out the content for us evenly and formatted to what we were aiming for. Both the log-in button and click here button were created using Bootstrap’s scripts. A separate class was also made for the footer.

## Design (Index Page)



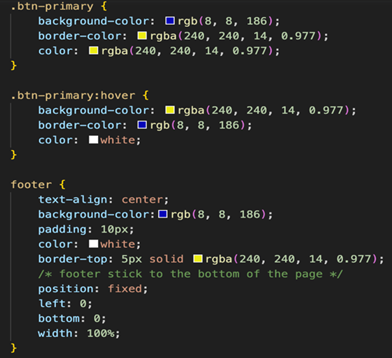
### Figure 1.6. *Code Snippet of Background Design (Index Page)*

Justin also created our own style sheet to design the webpage to look similar to the existing designs of CSUB websites. Figure 1.6 shows highlights the details of how the background of our index page was designed. He created a URL path to set the image in the background, then we made sure to that the image would not repeat itself in the background. He experienced some issues with that earlier in the project due to the size of the image being too small. Then Justin went ahead and made sure the image covered the entire background of the page, positioned it in the center, and blended the image with the color grey to decrease the brightness of the image.



### Figure 1.7. *Code Snippet of Header and Container*

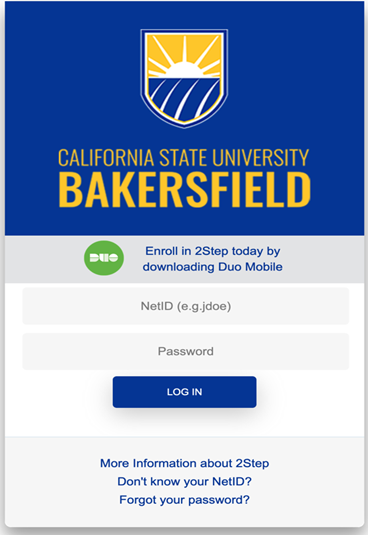
Figure 1.7 showcases the design process of the header and container welcoming the user. As is shown, we wanted to ensure that the colors being used would match CSUB’s color theme. The padding and borders were all adjusted to align the text in an appropriate format as well as adjusting the border edges to make them less sharp and rounder.



### Figure 1.8. *Code Snippet of Buttons and Footer*

Figure 1.8 displays the design of the buttons, primarily how the colors were implemented. As you can see, the color changes on the button when the cursor is hovered over a button. It’s a very simple technique that Justin has been very experienced in doing and has always found to make the page feel more interactive. The footer follows the same idea as the header in terms of the color scheme and ensure that footer sticks to the bottom of the page because the information on the index page is so little.

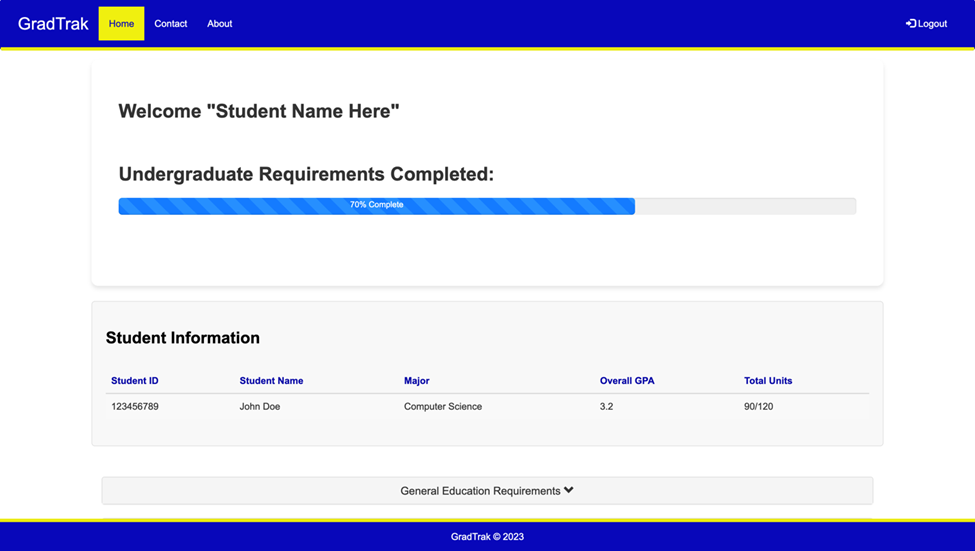
## Login Page



### Figure 1.9. *Login Page*

Initially we wanted to design our own login page and were in the process of doing so, however, after getting feedback from our instructor, it was suggested that we use the already existing CSUB login page. Our application is supposed to be used for CSUB students and made specifically for the university, so after having a couple of discussions amongst ourselves and Dr. Kaur, it was agreed upon to go ahead and implement the existing login page. It looks great and already provides all the functionality required for a traditional login page, so it was a no brainer. As for the code, it was modified to suit our needs, we went ahead and changed the page names to match that of ours so after the user does login, they will be redirected to the correct page. Which we will be discussing in the next section.

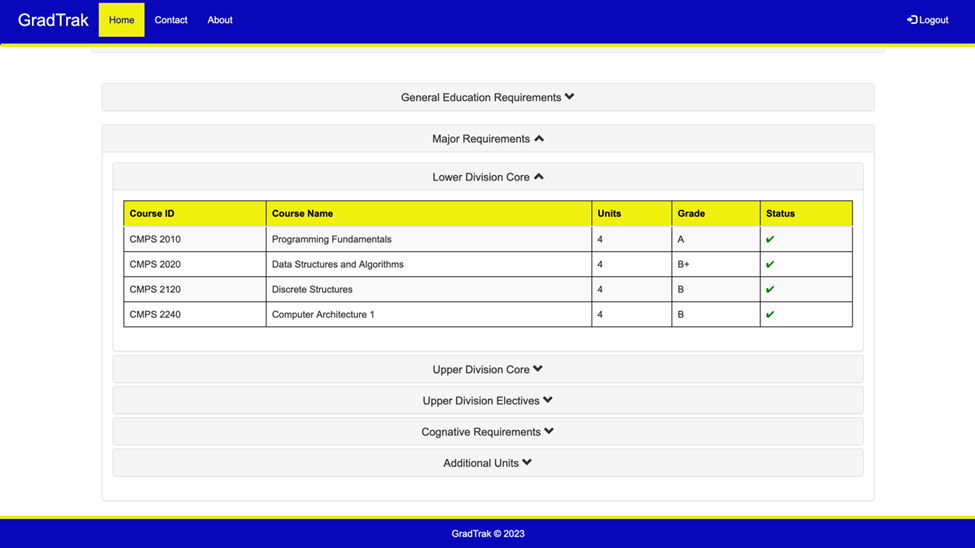
## Student Page



### Figure 2. *Student Page (PC Layout)*

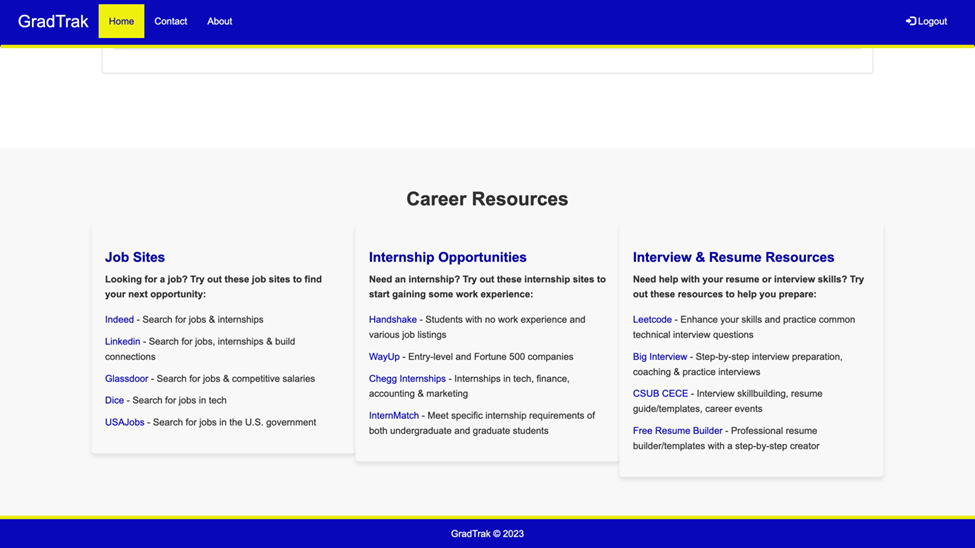
The design of the student page was a long and lengthy process. Different ideas were being shared and the layout was constantly being changed. But after spending a significant number of weeks and getting feedback from one another as well as the instructor, it became clearer as to what was expected and what features to implement. As shown in Figure 2, our student page contains a navigation bar that displays the name of our application, a link to the homepage, a contact page, an about us page, and a logout link that signs the user out of the student page and back to the index page. It also displays a message welcoming the user and displaying their name, and a labeled progress bar that shows how close students are to meeting 100 percent of their undergraduate requirements. The progress bar is dynamic and can be explained by other group members that worked on that implementation. It also displays the current user’s information from their student id, name, major, overall GPA, and the number of units that the student has completed thus far. The navigation bar also highlights in yellow the current page that the user is on. So, if they were click on the about us link for example, then that link would be highlighted in yellow instead of the home link.

Moving on to Figure 2.1, this image shows 2 different panels, the General Education Requirements, and the Major Requirements. When either panel is clicked on, it will display different subsections that exist such as foundational skills and capstone requirements, or lower and upper division requirements needed to be satisfied. As shown in Figure 2.1, when one of these subsections is clicked on, it will display a table that will display the Course ID, Course Name, the number units specific to that course, a grade if they have taken the course, and whether that status of the course has been satisfied or not. There are tables for each and every subsection that will display different information.



### Figure 2.1. *Course Requirements*

After implementing the most important features that our group had discussed in Senior Project I, we felt that it our student page was missing something. It did not feel complete is the best way we could describe it and we felt it could use more user interactivity. So, after spending a lot of time thinking and doing research, Justin went ahead and added in a “Career Resources” section. As shown in Figure 2.2, this section contains various different things. The first being the “Job Sites” section that provides the user with different websites that they can navigate to such as popular options like Indeed, LinkedIn, Glassdoor, etc. It even includes a website called Dice for people searching for jobs specifically in tech as well as another website, USAjobs, that is for students looking to work for the government. The next section, “Internship Opportunities”, that allows the user to visit different websites with different interests to search for various internship opportunities, regardless if the student has no prior work experience. The final section, “Interview & Resume Resources”, provides students free and paid resources such as the CSUB CECE or Big Interview, to help students build professional resumes to help them land interviews and help them clean up their interview skills. The career resources section is well researched to provide students with the best possible tools to guarantee them an internship or job. It saves the students time, resources, and helps them start thinking about that next big leap after graduating from college.



### Figure 2.2. *Career Resources*

## Code (Student Page)



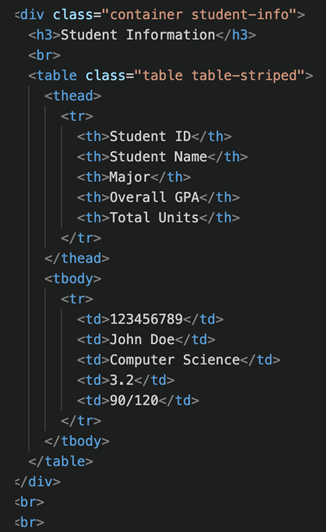
### Figure 2.3. *Code Snippet of Navigation Bar*

The implementation of the navigation bar was very basic and simple. Figure 2.3 shows how Justin simply defined a nav class using HTML and Bootstrap, put the navbar in a fluid container, created an unordered list and added relative paths to different pages within the application and separate links. Justin also used Bootstrap to add the logout icon that can be seen in Figure 2.



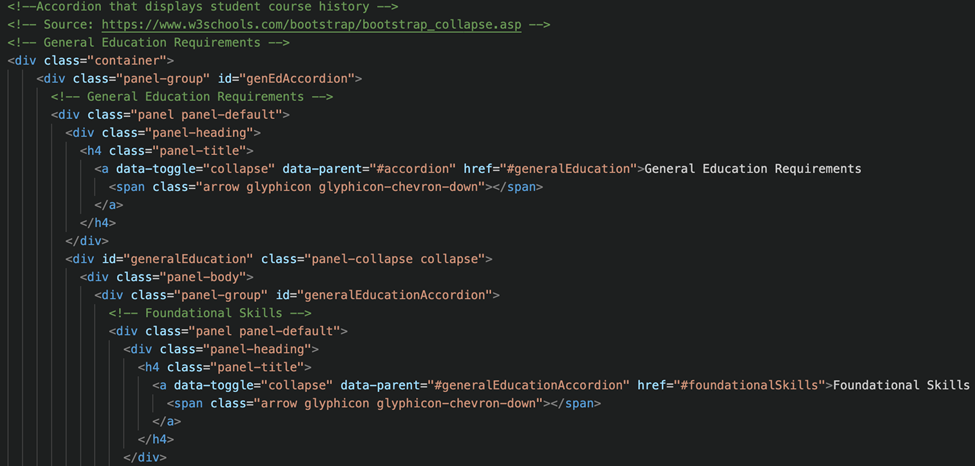
### Figure 2.4. *Code Snippet of Progress Bar*

Creating the progress bar was a very simple process, probably the easiest thing to implement out of everything in this section. Again, HTML and Bootstrap were used to create the progress bar, all Justin did was create a class called progress, and another class to define a striped progress bar to make it look more dynamic and less basic. Justin set the total value to 100 and the current value to 70 to test out how the progress bar would like once our project was completed.



### Figure 2.5. Code Snippet of Student Information Structure

Figure 2.5 Displays the structure of the student information section. It is put into a separate container, used Bootstrap to create the table formatting and HTML to fill the rows with information. As mentioned earlier, this all this information was hardcoded to get a general idea as to what our application would like once it was completed. The back-end will be explained later in this report.



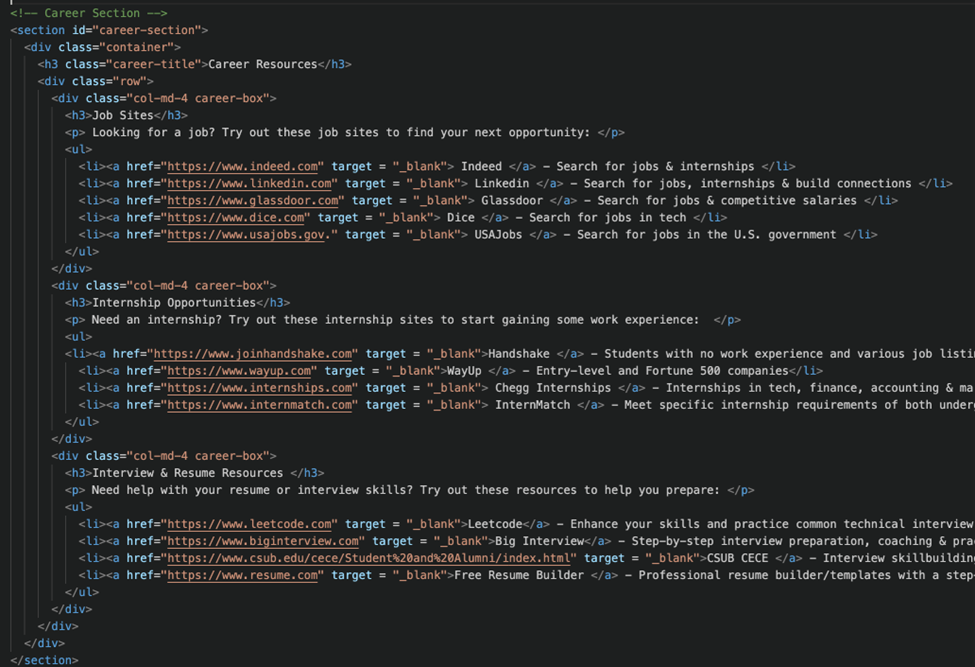
### Figure 2.6. *Code Snippet of Accordion and Collapsible Elements*

Figure 2.6 displays the general structuring for the collapsible panels that Justin used to make the General Requirements section as well as the Major Requirements section in Figure 2.1. It was a very simple process that only required him to use collapsible panels and an accordion that only displays 1 collapsed item at a time. He also created and labeled separate ID’s for both the requirements sections so that they could be their own separate entities, and so that he could make the page follow the mock image that we had envisioned. This was arguably the most important part of our application simply because this is the main feature that our targeted users would be using the most. Justin had to test it many times to ensure that there were no problems occurring when a user was engaging with the collapsible elements. As shown in Figure 2.7, Justin went ahead and created tables within each collapsible element so that user’s course history could be displayed in well-structured manner. All the data seen here was all hardcoded just so that he could adjust the content effectively for any device. The tables are responsive and does change in size if a user does access the website using a smaller screen. This was all applied using a media query to apply different styles for different screens, which can be seen in our CSS file on our GitHub. Without applying this technique, the formatting of the tables on smaller screen would have not adjusted properly and caused a huge issue.



### Figure 2.7. *Code Snippet of Table Structure in Panels*

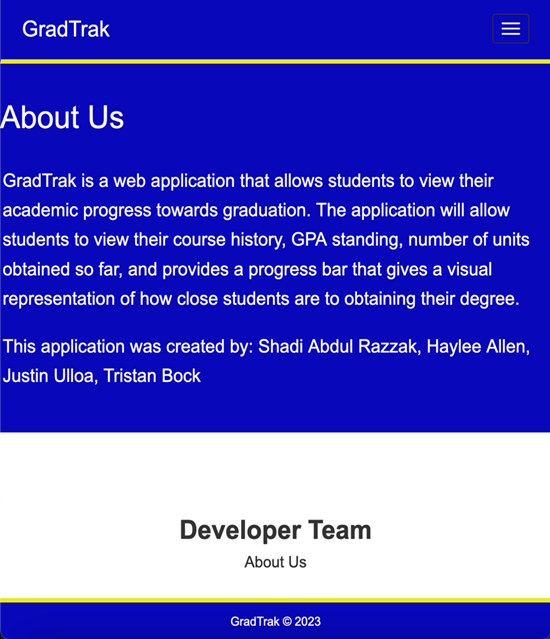
Finally, our Career Resources section was implemented by taking advantage of HTML’s section tags and creating a separate container so that all the related content within this section would remain in one spot. Figure 2.8 displays the structuring of this specific section. As is shown, Justin went ahead and created even columns for each separate section such as the jobs, internships, and resume/interview resources so that they would be displayed evenly. Justin also structured the links to specifically open a new tab with the link that was selected by the user so that they did not have leave the webpage and could simply have multiple different tabs up and be ready for them to access while remaining on the page. Justin made sure to put the links within lists so that he could add brief descriptions to each individual resource. He also edited the links and changed the colors to let the user know what links were accessed and which ones have not. As was mentioned earlier, our overall goal for the front-end design of the application was to make it as simple as possible to navigate to ensure a great quality experience.



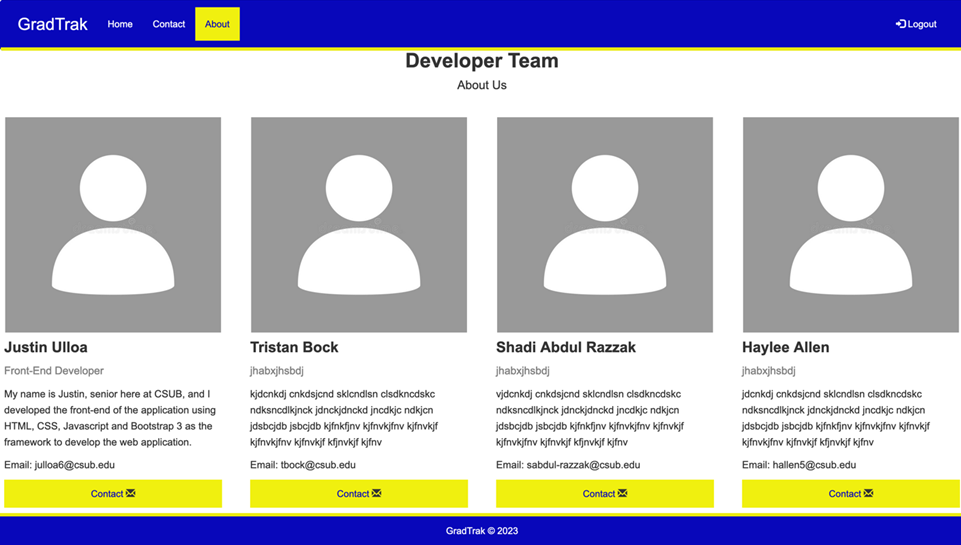
### Figure 2.8. *Code Snippet of Career Resources Section*

## About Page

After finishing with the design of the student page, we also decided to create an about us page (Figure 2.9). This page essentially provides a brief description of the goal of our project, lists the names of everyone involved with the creation of our project, a contact section with the images, names, and roles of each team member. We wanted this page to be a place where the user can meet the team behind the creation of this wonderful application, get a better understanding of how the website was created and to be able to directly contact any team member via email (Figure 3). We also decided to add in a featurette section to the page to not only enhance the quality of the page, but to show the user, especially new users, the main features of our application. We also made sure to add in images in this section so that our website could feel more visually appealing and to add some “life” to our webpage. We figured what better place to add finally add some images to our application than here (Figure 3.1).



### Figure 3. *About Us*



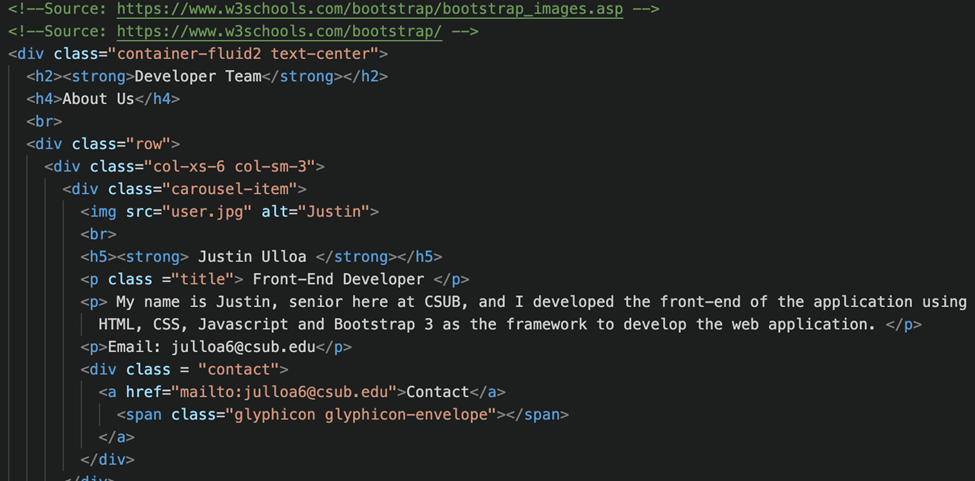
### Figure 3.1. *Developer Team*



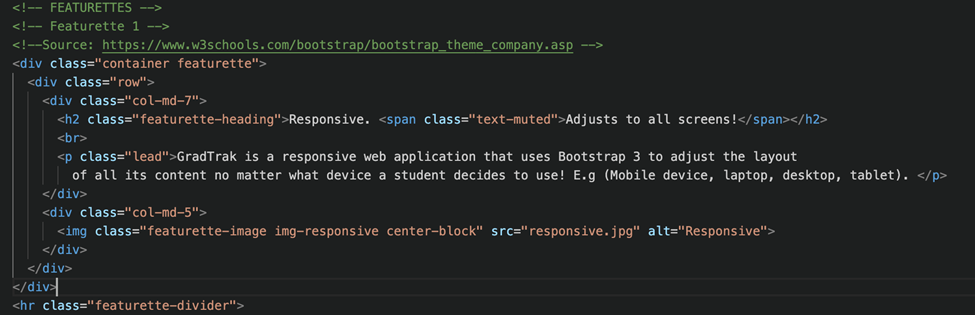
### Figure 3.2. *Featurette Section*

## Code (About Page)

Figure 3.3 showcases the implementation of the Developer team section in the about us page. Just put the content in a container, split the content into even columns using Bootstrap and the overall design of the individual sections of each group member was created setting a div class as a “carousel-item”. HE also made a separate div class, “contact”, so that separate the link from the other content and fill it with the color yellow to follow the theme our application. Justin also used Bootstrap to add an envelope icon next the link so that design could look more modern. As for the featurette implementation, he defined a class with a container that specifically creates the formatting of the featurettes. He also defined a separate span class, “text-muted” (Figure 3.4), that allowed him to have two different color schemes for the heading (Figure 3.2). Finally, Justin simply made 2 separate columns dividing the wording and image within each featurette. That is the end of our final implantation of the front-end of our application.



### Figure 3.3 *Code Snippet of Developer Team Structure*



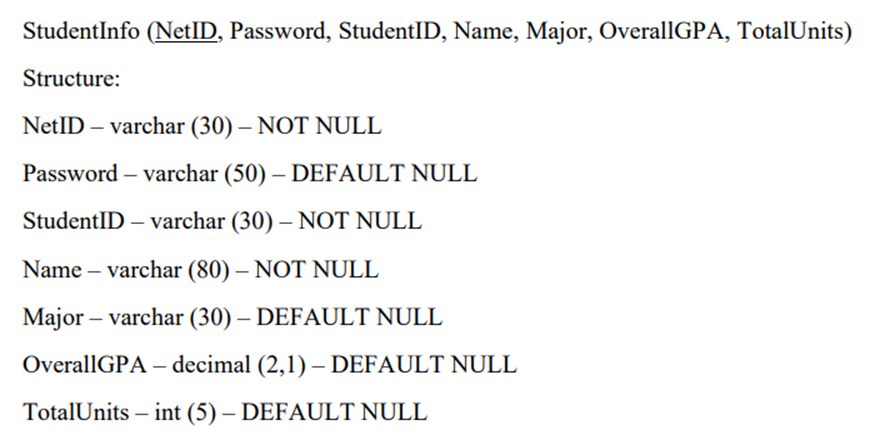
### Figure 3.4. *Code Snippet of Featurette Section*

## Back-End

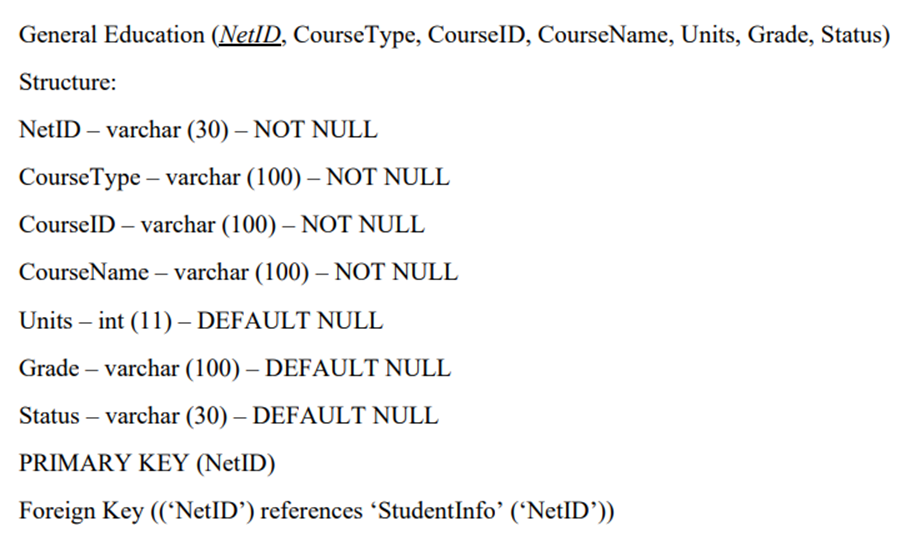
The back-end implementation of the application consisted of utilizing a couple of different types of software in order to make the application run functionally. Haylee and Shadi worked on the implementation of these features. The host of the database was via Artemis which was available via CSUB, but the actually programs used were MariaDB [2] and MySQL [3].

## Table/ER Diagram Design

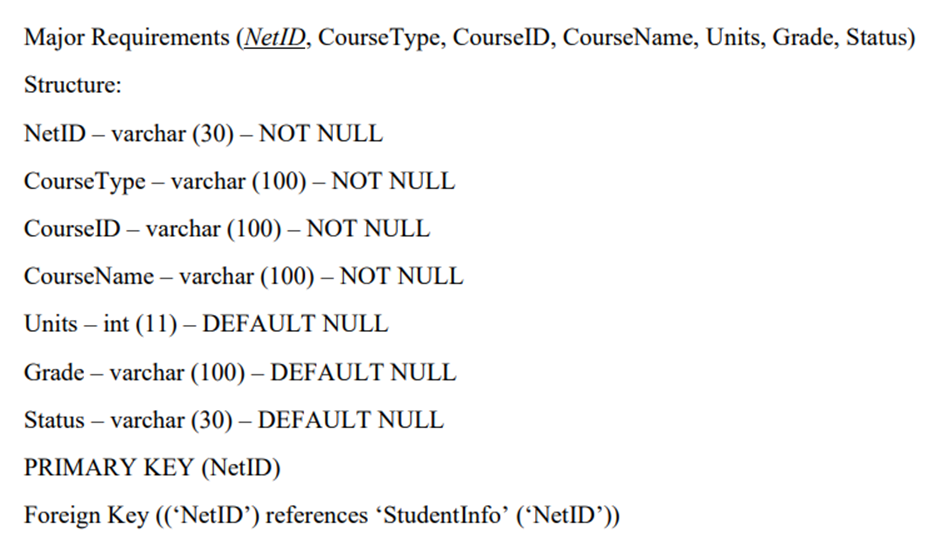
Before the database could be implemented, Haylee worked on designing the tables that would store the proper information had to be created. After a few alterations, three main tables were decided on. These three tables were a Student Info table, a General Education table, and a Major Requirements table. The Student Info table is the table that would contain all necessary information about the student which included but was not limited to a student’s name, netid, major, and studentid. Then for the General Education and Major Requirements tables, they both contained information such as but not limited to a student’s netid, the course id, course name, the student’s grade of the respective course, and the status which conveyed if a student either completed or did not complete a course. The table structure in the design portion was mapped out as such, Figure 3.5 shows the structure of the StudentInfo table, Figure 3.6 shows the structure of the GeneralEducation Table, and Figure 3.7 shows the structure of the MajorRequirements Table.



### Figure 3.5. *Structure of StudentInfo Table*



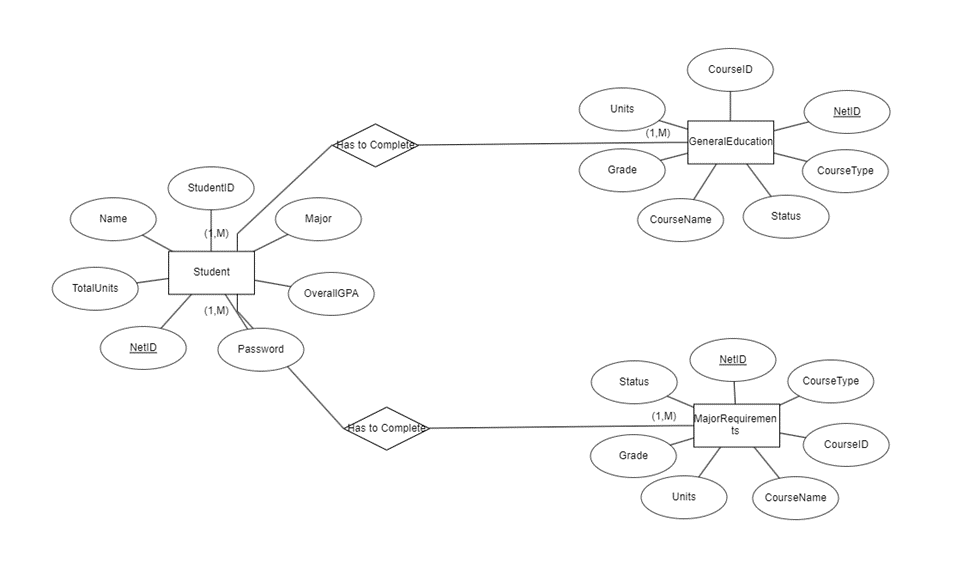
### Figure 3.6. *Structure of GeneralEducation Table*



### Figure 3.7. *Structure of the MajorRequirements Table*

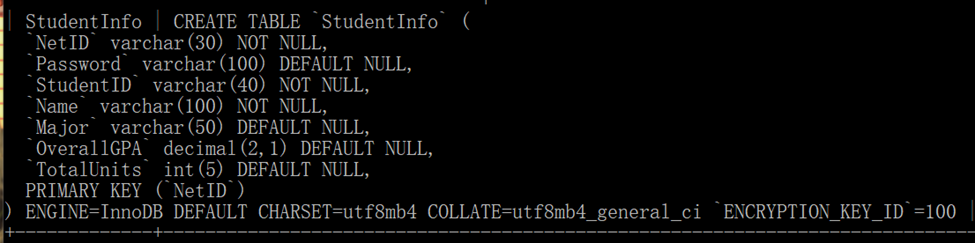
## Database Implementation

Once these tables were confirmed and finalized, the database implementation process began. For this, we finalized the ER diagram that would explain the path in which all the tables would connect. In the ER diagram, the primary keys of each table are underlined in order to emphasize that the attribute is unique/is the primary key of the table. What is not shown in the ER diagram is what the foreign keys of each table were. For the StudentInfo table, there is not a foreign key because that table is the main table in which the other two tables depend on to make the necessary connections. For the GeneralEducation and MajorRequirements tables, the foreign key is the same as the primary key. In their specific cases, NetID is both the primary key and foreign key, this is because those tables reference the StudentInfo table in order to make proper connections. Even though the diagram itself does not have a feature to highlight this, the referencing information is shown in figures 3.6 and 3.7 in the written-out visualization of the table. The finalized ER diagram is shown in Figure 3.8. To further explain some of the columns of tables that are not as clearly defined, in the StudentInfo table, the TotalUnits column is the total number of units a student has completed so far. Then in the GeneralEducation and MajorRequirements tables, the Units column is how many units a specific course is, and the Status column is what conveys if a student has either completed a course or if they have not completed it yet.



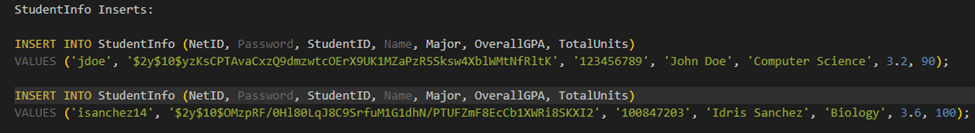
### Figure 3.8. *Finalized ER Diagram*

The first step in implementing this information to the database was to create the tables on MariaDB using MySQL. Haylee worked on implementing these tables into the program. The create table statement for the StudentInfo table is shown in Figure 3.9.

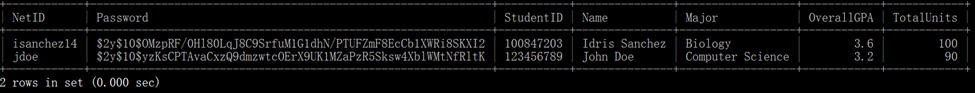


### Figure 3.9. *Create Table Statement for StudentInfo Table*

The same was done in order to create the tables for GeneralEducation and MajorRequirements, the difference of course being that those tables have some differing column names, but the general syntax is the same. The next step was to begin adding the data to the tables. For the project, two users were created to showcase how the application works. The main user created was John Doe, and the second user created was Idris Sanchez. Two users were created to ensure that only each respective user’s information showed upon accessing the website and to ensure that coding worked properly and did not show the same information for two different users. The insert statements for each user consisted of calling an Insert Into statement to the StudentInfo table, followed by the columns the information was to be stored in and then the actual information itself. The code used to implement the user information is shown in Figure 4 and following it is Figure 4.1. Figure 4.1 shows how the table appears on the actual database after the insert statement query completes.

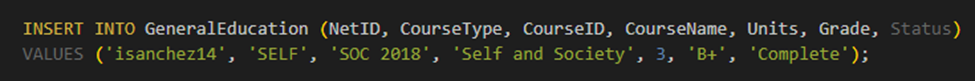


### Figure 4*. Insert Statement Query for StudentInfo*



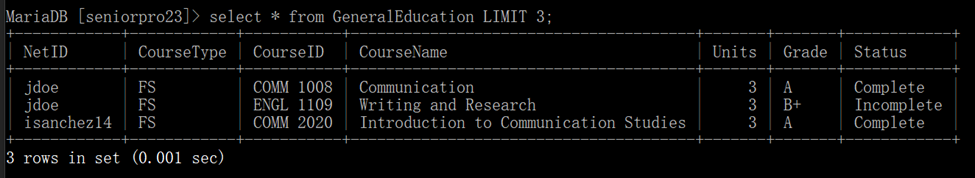
### Figure 4.1. *Database Table of StudentInfo After Insert Query*

With student information available, the next step was to implement data for the GeneralEducation table and the MajorRequirements table for each student. This process also utilized an Insert Into statement into each respective table. Figure 4.2 showcases what an insert statement into the GeneralEducation looks like. The same structure was also utilized for Inserts into the MajorRequirements table.



### Figure 4.2. *Insert Statement Into the GeneralEducation Table*

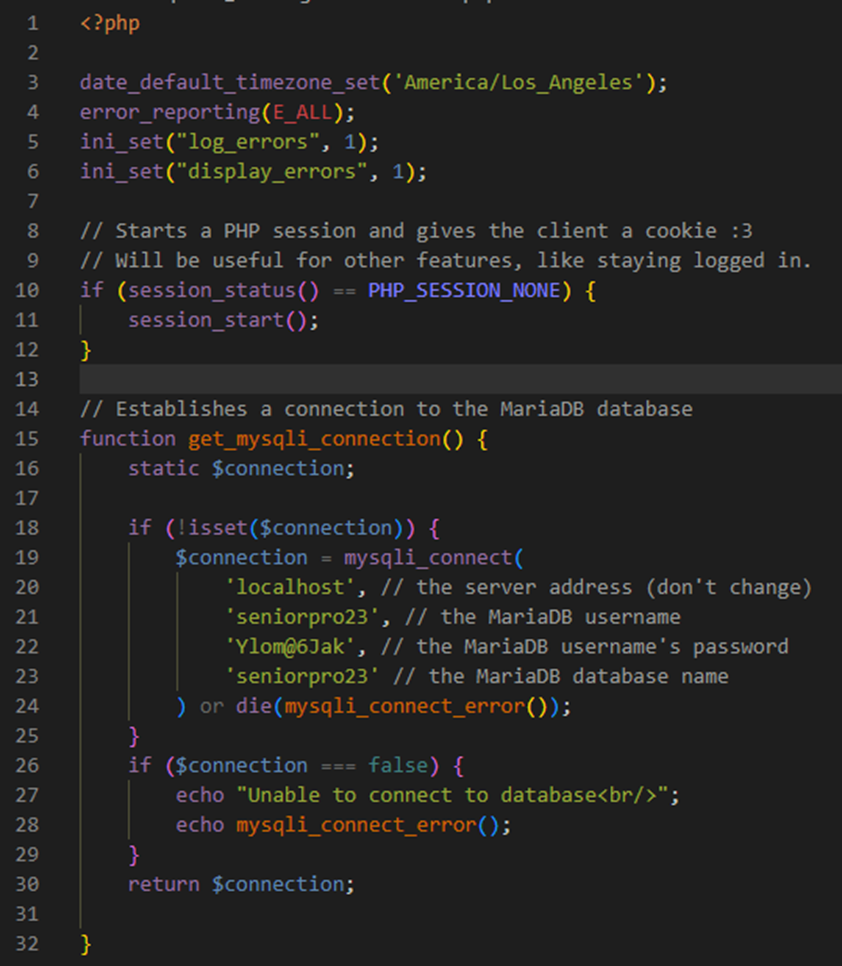
Figure 4.3 shows how the insert statement looks on the database after its completion.



### Figure 4.3. *Database Table of GeneralEducation After Insert Query*

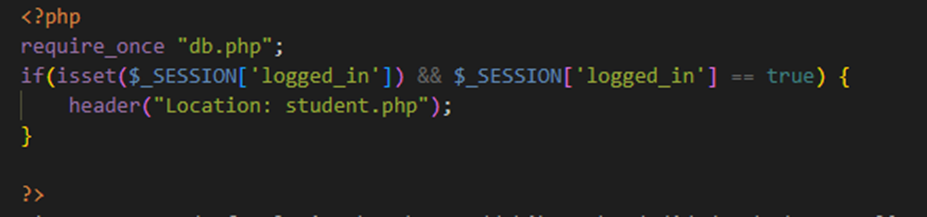
## Connection of Database to Website

Once all the data was inserted into the database, the next thing that took place was the connection of the database to the website. This is what links the back-end to the front-end of the application. To do this, php was used to establish a connection to the database. The code used to establish connection is shown in Figure 4.4.



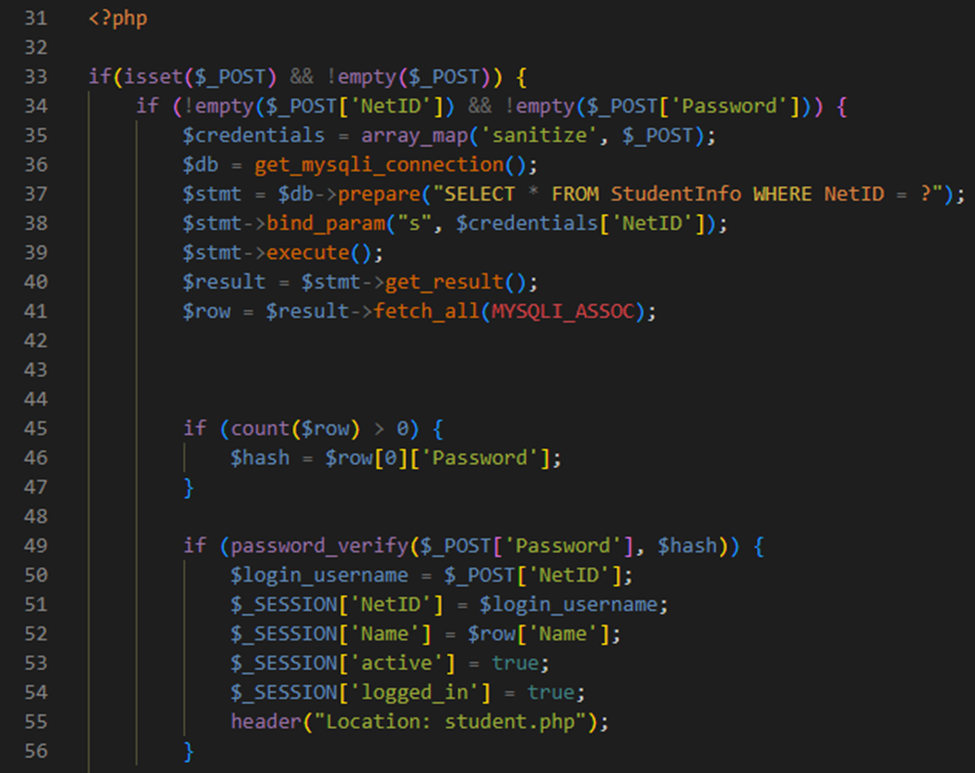
### Figure 4.4. *PHP Code to establish connection to Database*

The breakdown of the Figure 4.4 snippet is as follows. On line 4, error\_logging is enabled which is helpful if there were any issues with displaying information from the database to the website interface. The error\_logging gives a brief statement stating on which line in the php/html code the error is being caused from. Line 15 is what begins the call to the database to get connected. Then to establish the connection, the MariaDB username, Password for the respective database, and the actual database name had to be declared as shown on lines 20 – 23. On the actual website page, line 27 is there as a precaution in case the connection to the database failed to notify us that we are not connected. Otherwise, as long as the line does not echo, the connection to the database is successful and then we are able to start calling data from the database to display on the website interface. After the connection was confirmed to be successful the next task was to connect the login page to the database. There were a couple reasons this was the first task. The login page is what would enable each user to only see their data. Also, this is why there is a password column in the StudentInfo table, so each student can login and see their own progress. Firstly, this snippet shown in Figure 4.5 is what was included at the top of each page in which would enable making calls to the database.



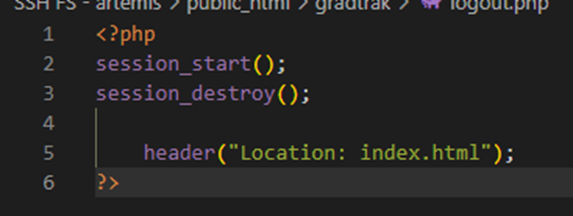
### Figure 4.5. *Establish Connection to the Database outside of db.php*

Next, in Figure 4.6, the code that establishes a properly working login is shown.



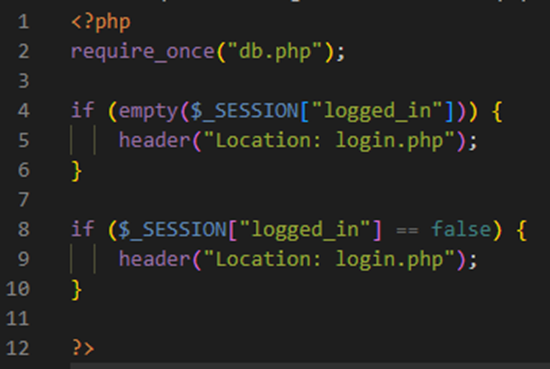
### Figure 4.6. *Code Needed to Login*

Shadi developed this working login code. What lines 33 – 41 are doing is upon the submission of the login button, the code is making sure the NetID and Password fields are not blank. If they are not blank, a database connection is established to select all columns from the StudentInfo table where the NetID entered on the login form matches a NetID established in the database. Then on lines 45 – 49 the code here is verifying that the user’s password entered password matches the hashed version of it that is present in the database tables. In order to implement increased security, a user’s plaintext password is not stored in the database, instead a hashed version of it is. So, the code here is verifying that the plaintext entered password is an exact match for the hashed one stored. In lines 51 – 55, the $\_SESSION[‘NetID’] and $\_SESSION[Name] store the user’s NetID and Name in session variables which are used to confirm that that specific user is logged in. Line 55 takes the student to the student.php page if their login is successful. The next page developed to include database functionality was the logout.php page. There is a logout button on the student.php page, so when a student is done viewing their information, they can logout. The structure of the logout page is shown in Figure 4.7.



### Figure 4.7. *Structure of Logout Page*

What this snippet is doing is destroying the user’s session, the session is what confirms that the user is logged in so when they are ready to logout their session is destroyed or cleared which in turn redirects them to the index page. With the session being destroyed, if the user tried to renavigate back to the student.php page, they would be directed to login page where they would have to login again to regain access. The biggest and final connection between the database and the website interface took place on the student.php page. Haylee worked on getting the information from the database efficiently connected to the student.php page. This page contained the student’s graduation progress, so all connections had to work properly and work correctly for each student that logged in. This included making sure that a student would only see their own information and not other student’s information. The first thing that was implemented here was the verification of a student being logged in. This was implemented first here because it ensured that if a user was successfully logged in, they would be navigated to the student.php page and if they were not logged in or if they tried to access the student.php page before logging in that they would be properly redirected to the login.php page. The code required to make this happen is shown in Figure 4.8.



### Figure 4.8. *Student Login Verification Snippet*

Lines 4 – 10 convey that if the either the $\_SESSION[logged\_in] condition was empty or false then a user would be directed to the login.php page. This successfully made it so that a user had to be logged in in order to access the student page. The next step of implementation between the database and the website interface was to get the tables that were created on the database to show on the website. Figure 4.9 shows the basic structure for how these tables were connected. For each table present on the website interface this same structure was used.

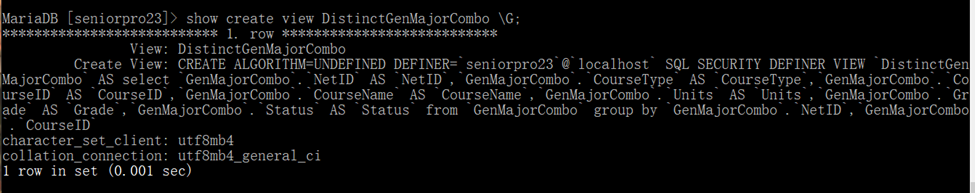


### Figure 4.9. *Basic Structure to Connect Database Tables to Website*

The layout works as follows. The first thing is the establishment of a student’s NetID, that is done on line 180. This is done first because in order to show only the logged in user’s information we get that user’s NetID from the session storage. On line 182, the query statement called is going to pull the columns of course ID, course name, units, grade, and status from, in this instance, the General Education Table that corresponds to the course type declared and the NetID. NetID is being equated to a question mark because we want the NetID of whichever student is logged in. That is where lines 183 – 186 come in hand. They fetch the NetID from the session and bind that value so that the query essentially calls for whichever NetID is logged in the current session. This same format is also used for the Major Requirements table, the only adjustment that is made is that the call would say: “FROM MajorRequirements” and the course type called would be whichever applicable Course type is needed for that specific table. On lines 188 – 197 this is where the actual results get displayed to the webpage. Line 188 first establishes the condition needed in order to begin fetching the tables, that condition is if the number of rows is greater than 0. While that is true, $result will fetch the rows associated with the query executed. Beginning on line 190 the table rows are being called to echo onto the page. The $row(s) are populating the tables on the website with the actual data that is present in the database. By that, that means that the $row[“CourseID”] is populated with course id’s that correspond to the given query. Line 193 echoes the entire table after the while loop has finished running.

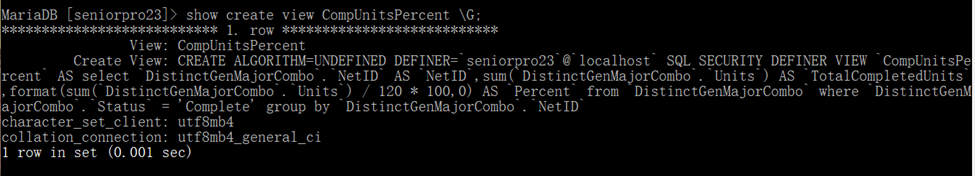
In case there is an issue with the syntax or an error that has anything to do with the tables working properly, lines 195 – 196 are there to echo a “Table Error” statement letting us know there is an issue with the code between lines 188 – 197. The last portion of the database that had to be connected to the website interface was the task of making the student progress bar auto increment as well as making sure the student’s total units would automatically increment as well. This task consisted of multiple parts to make it work. The first part was to combine the General Education table with the Major Requirements table. This was done use the implementation of MySQL Views. Views are extremely helpful because they store a query in the database catalog. Therefore, instead of having to call a specific query every time the database is accessed, the query can just be stored as a view and then the view name can be called, and those results are always there. The best part for this implementation is that views automatically update. Due to both tables having the same structure a simple join combined the two tables. Following the next task was to narrow the combined tables down so that there were no repeat courses.

For this, another view was created, this one was named DistinctGenMajorCombo which stood for distinct values of the combined General Education and Major Requirements table. The issue that this table solved is that it would make sure a course that is present in more than one category would only get units counted for it once. For instance, the capstone requirement for students is usually also a major requirement for students. This course appears in both categories, what this table does is it will only count the units for that course once. That way even if the course fulfills multiple categories, the student is shown on their end via the total unit’s section that they only receive credit one time. This table utilized a group by call to NetID and CourseID. The reason for this is because the group by call will filter for distinct results. Due to this, both NetID and CourseID were called. CourseID is called because this is what was used to determine if a duplicate course was found, then NetID was called so that even if two students took the same course, both would still be credited with those units. Without the call to NetID, only the first student found who took a course would show as having those units completed. Which would make every other student’s total completed units value be inaccurate. Figure 5 shows the structure of how the view was created for the DistinctGenMajorCombo table.



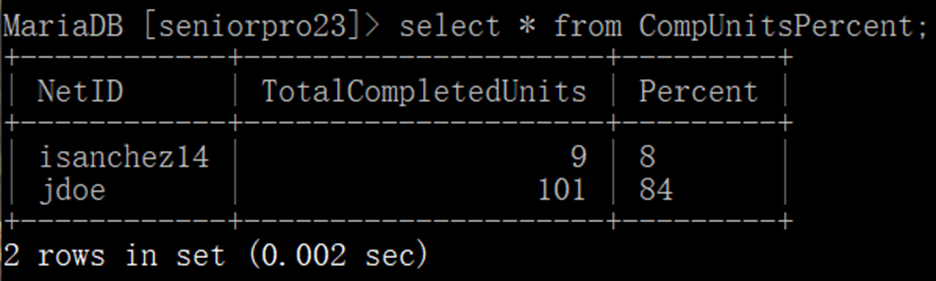
### Figure 5. *How the DistinctGenMajorCombo View Was Created*

The next step was to calculate the number of units total the student completed along with a percentage amount of those completed units out of 120. The 120 comes from the fact that you need at least 120 units to graduate so the calculated percentage had to reflect that value. This view only shows the student’s NetID, their total completed units, and the percentage completed based on those total units our of 120. Then because a student can have an incomplete course but because that course still has information pertaining to the number of units that course is, a where condition is present in the table creation statement that states that the Status must equal “Complete.” With the inclusion of this, a student’s total completed units only accounts for completed courses. The statement used to create this view is shown in Figure 5.1.



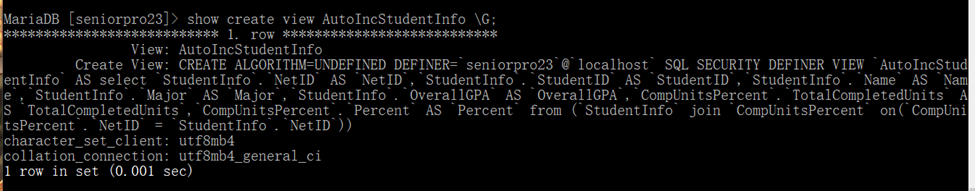
### Figure 5.1. *How the CompUnitsPercent View Was Created*

Figure 5.2 show how the actual table for CompUnitsPercent looks.

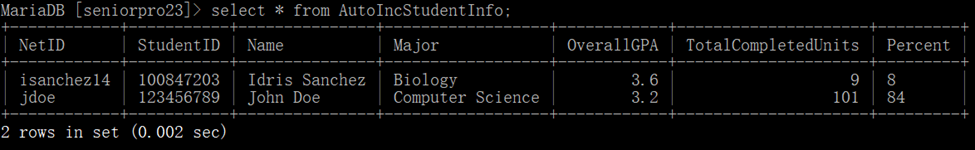


### Figure 5.2. *Table for CompUnitsPercent*

The last view created was AutoIncStudentInfo. This described how this table was the one that the data, specifically the total completed units, and the percent would automatically increment. The actual tables from this view are a combination of the StudentInfo table and the CompUnitsPercent table. How the view for the AutoIncStudentInfo table was created is shown in Figure 5.3 and how the table looks with data populating it is shown in Figure 5.4.

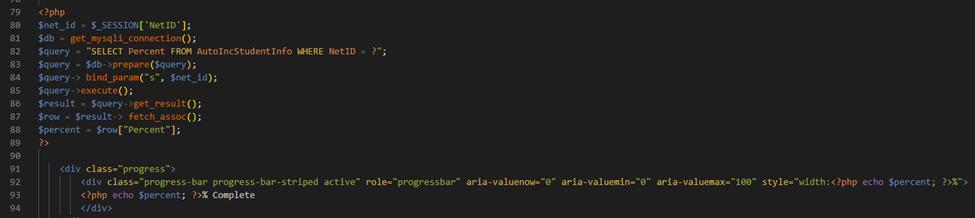


### Figure 5.3. *How the AutoIncStudentInfo View Was Created*



### Figure 5.4. *Table for AutoIncStudentInfo*

Now that the table were all implemented successfully, the last part needed was to connect the percent value with the website progress bar. The snippet needed to do this is shown in Figure 5.5.



### Figure 5.5. *Code to Connect Database to Progress Bar*

The main thing here is on line 82. Line 82 is selecting the percent from whichever student is logged in based on their NetID. Then on line 88 the row with the percent value is assigned to a $percent variable. This is then used on line 92 in which it echoes the percent value to the width of the progress bar. So, for instance if the percentage of units completed for a student is 82, then the progress bar will show a width of 82 percent. For increased clarity, on line 93 we echo the percent value once again. What this line here does it that it visually displays that amount in the middle of the progress bar. In this case it would show 82% Complete.

# Conclusion

In summary, this project has taught us all different things because of the different tasks we had to take on. It was great to present this at the SeniorExpo where we had many people comment that CSUB should include this themselves. Or the funny comments of “How is this not a thing yet?” were really encouraging as all students, current or past, had a linked feeling towards this project, just like us as well. We feel as though this is a really good starting plan for the project, because we had tried to include features that we wouldn’t need to come back and implement later on in the future. However, as every application always goes through debugging and feedback after launch, it would be no doubt this application deserves the future plans imposed by users for what they prefer is best, as this is a student application after all. As this moment, this application carries no future plans unless regarding to a new specific useful implementation that would benefit the lot of students. If we ever did have to do this again, the only thing we would change is database implementation, as I had the wrong idea in the beginning which would have help with time arrangements a lot more.

# Appendix

GitHub Link:

<https://github.com/Kersplosion/SeniorProjectCSUB2022-23>

As there is only one branch, we have collectively been working together on the main branch. On first glance, you will notice two folders and several other raw files. Through the files, our proposal and reports from old semester can be found and viewed, our SeniorExpo poster, and Group Report are posted as well. The two folders include “Database Materials” and “GradTrak”. The first folder is as the name suggests, and with the help of Shadi and Haylee, it holds the login information and functioning for the users we have in the database for effective testing for whoever is viewing. The second folder ,“GradTrak”, includes our entire project, with more database implementations. Justin and Tristan took care of many User Interface aspects that hold place in that folder.

# Works Cited

[1]W3schools, ”Bootstrap 3 Tutorial”, [Online] <https://www.w3schools.com/bootstrap/>

[2] MariaDB Foundation, "MariaDB Server: The open source relational database", [Online] <https://mariadb.org/>

[3] MySQL, “The world’s most popular open source database”, [Online] <https://www.mysql.com/>