**Contributions/Responsibilities**

We divided the work by technology, but also assisted each other where needed to complete the project. Christopher handled the logins and templates for our access and output formatting while Daniel deployed the backend SQL database and worked on the associated functions with inserting, updating, and removing database elements.

**Overall Structure**

Our web app consists of 3 web pages all tied to a database:

1. Login
2. Gradebook (index)
3. Edit

The Login is simply for the user to log into our web app. A user can view the other URLs without being logged in but they will only see the navigation bar. Once the user is logged in, they will then be directed to the gradebook URL which displays the gadebrook for the course. On the navigation bar there is an Edit tab that allows the user to edit the gradebook with functionality described below. The option to logout once logged in will then redirect the user again to the login page if they so choose to log out.

**Database Structure**

The database structure we chose for the project was rather simple, utilizing three tables as outlined below:

STUDENT

· student\_id INT(PK)

· first\_name VARCHAR

· last\_name VARCHAR

· student\_major VARCHAR

· student\_email VARCHAR

ASSIGNMENT

· assignment\_id INT (PK)

· assignment\_name VARCHAR

GRADED\_ASSIGNMENT

· assignment\_id INT (FK/PK)

· student\_id INT (FK/PK)

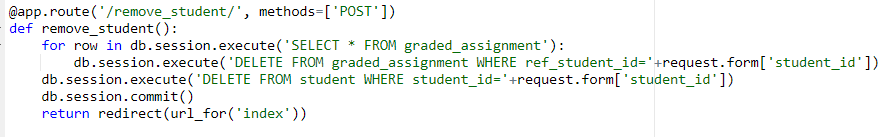
· grade DECIMAL

The Student table contains information for students in the class (first name, last name, major, and email.) The Assignment table simply contains the assignment name. The Graded\_Assignment table has a composite primary key consisting of references to the assignment\_id from the Assignment table and the student\_id from the Student table. These combine to provide a unique value to a graded assignment for each assignment and student, and then has the grade column for the actual grade received for the assignment.

**Notable Elements**

All primary keys auto-increment upon new insertion, and our one table with foreign keys, graded\_assignment, is also propagated with data representative of the new primary keys so no dependency issues arise.

We created flask routes for function references utilized on our Edit page. These include add\_student, remove\_student, edit\_grade, add\_assignment, and remove\_assignment. To ensure proper representation of data when a student or assignment is added or removed, we utilized For loops in those respective functions, again, to propagate the insertion or removal of, for example, assignments when a student was added or removed. This can be seen in a snippet from our remove\_student function below:

We structured the loop to construct an iterative SQL statement to remove not just the specified student, but all graded assignments tied to that student as well. Vice versa, we inserted ungraded (default 0) graded assignments for new students and for new assignments for all students. 

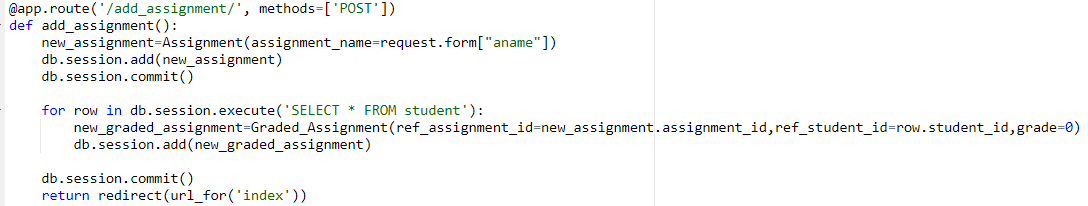
For grade editing we utilized a dual-input form to select the student and assignment a user would want to change the grade for, followed by a numeric input of the new grade:

We chose the route to reduce clutter and maintain a separation between the gradebook view and the editing page. Additionally, all edits result in a redirect to the gradebook page so any changes can be viewed instantly.

**Testing Methodology**

We mainly tested as we continued development. We would make small changes, such as a new function or new template formatting and then deploy and check how it functioned. This allowed us to troubleshoot any errors with greater speed and accuracy than if we had implemented large edits and changes before deploying.

This methodology was especially useful in testing the structure of building out and executing SQL statements in our flask functions:



We previously discussed our For loops for data propagation, however it was truly challenging to actually build out. Adding a new assignment from user input was easy enough, however utilizing that new assignment ID and name when creating all the graded assignments required a lot of trial and error. Ultimately we came out with the above session execution utilizing the new\_assignment variable and the iterative selection of rows in the Student table to appropriately loop through each student to add the new assignment for everyone.