Moodle Grading Assistant

Department of Computer Science
Title: Online Grader Helper App
Authors: Emily Costa & Michael Sisko
Date: December 6, 2021
Advisor: Victor T. Norman

Problem

Grading is a time-consuming and redundant task, and some common complaints of Moodle's user interface are that it is clunky, slow, and frustrating. When someone wants to grade an assignment, you can enter the feedback there or pull down a CSV file, modify it and reupload. One consistent issue is redundancy, meaning similar feedback often applies to multiple students. Another issue is the difficulty graders have in modifying the table efficiently. If a grader wanted to change something globally, it would require fixing each individual entry tediously. We believe our application is crucial since it minimizes administration time and permits graders to allocate more time elsewhere.

Background

With our application, we plan to make the process quick and painless. Often students make the same error, and without our app, it is tedious for a grader to modify feedback should they change their mind. We expect our application to save graders and professors time through features that allow for efficient operation and minimize busywork wherever possible. We believe our application enables professors and graders to maximize what they enjoy without the needless hassle of the current grading software.

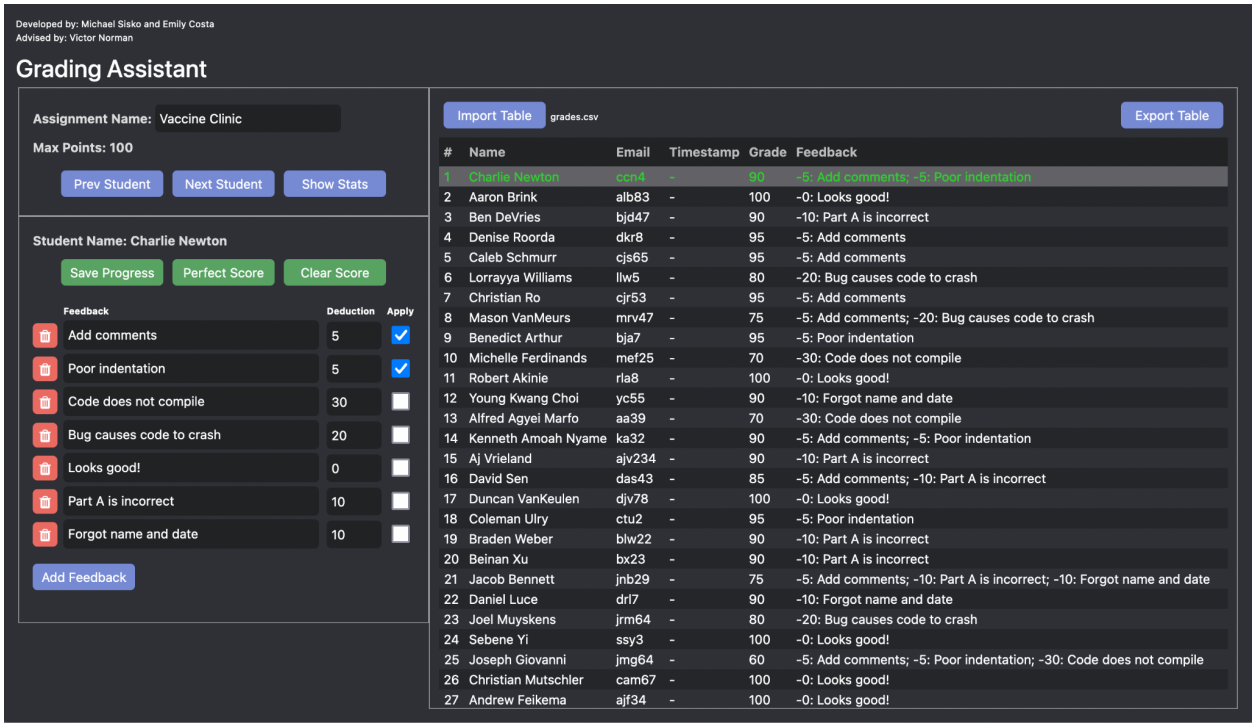
Solution Design

For the layout itself, our application had two main panels, a left panel that allows the user to add feedback, a right panel that displays a table for the user to select from. On the left panel after you import a table, you can apply feedback to a particular student by inputting a string and a deduction value, then checking a box to insert it into that cell. Graders can create, update, and delete options at any point in the process. Furthermore, we implemented feedback consistency, meaning that feedback value and wording changes will be correspondingly reflected wherever it is called. When the grader is done with the class, they can pull down the feedback into a CSV file to be reuploaded to moodle. Graders can also title assignments which will be reflected in the title of the CSV file they download along with a timestamp.

With our application, a student grader or professor would download an assignment's CSV file from Moodle, upload it, and then our app parses the file to get all of the students' information. The application displays a full chart of the students' full name, their Calvin email, their grade for the assignment, and the feedback. The CSV file does provide an image URL of each student, but we have decided to omit this information from our application to promote objective grading. The grader can proceed through the list of students, entering a string of feedback, and the value of the deduction. All of the feedback strings and their point values are stored into a saved list. Using this list, a grader can then just check off the feedback that applies to each student's submission. This eliminates the redundancy of typing in the same feedback for multiple students, through selecting the relevant custom feedback. Our application keeps track of the feedback given to each student, so if the grader modifies the custom feedback or the deduction value, if saved, the

change is reflected globally. Finally, the application displays assignment statistics: the frequency of each feedback, the average score, standard deviation, median, and mode. Once an assignment has been graded, the grader can then export the modified CSV file and upload it to Moodle.

Figure 1.0 - Potential Application State



Description: Above we see a few potential options for feedback on an assignment, and the options being applied on the table of students on the right. This displays some of the currently available functionality the application provides.

Design Norms

For the design norms that are applicable in this project, we decided that cultural appropriateness through relieving the burden of administrative time, transparency through the clarity and execution of the design, and trust for its reliability, were the most relevant given our application's purpose and use cases.

Cultural Appropriateness

We feel that our application alleviates the burden on the staff that organizes the curriculum. Teaching, guiding, and mentoring students requires plenty of time by itself, but beyond that teachers have to grade the work for every student. With all the grading that professors have to do, our application will help lessen the burden of administrative tasks. Since our application will help to reduce the load, we believe it will be conducive to the success of our students and professors alike. Similarly, we also believe it will help to expand the reach of our professors to inspire our students.

Transparency

We are also confident our application is easy to learn and requires little training. This aspect of our application is crucial as our target audience will not necessarily be tech-savvy. Our target audience is professors of any discipline, as well as teaching assistants and student-graders. Taking this into consideration, we aimed for our application to be relatively simple to prevent confusion. Also we intended for our documentation to provide ample information on the inner workings to allow others to build, modify, and expand our existing design however they see fit.

Trust

We believe our application is reliable and is not prone to failures. We have done thorough testing of our design using Cypress and various different user-tests to provide us a reasonable trust in its reliability as an application. We believe that our users can consistently and reliably use our application for the intended purpose of grading a set of students. Some helpful practices we have implemented include the incremental and thorough testing of each application feature. This allowed us to effectively eliminate many potentially application-breaking design flaws sooner than later.

Developmental Approach

In the development of our project, we used Angular, Bootstrap, Apex Charts, and GitHub. For the design of our project we implemented it as a web application built using Angular typescript. We decided to use Angular for our development because both of us were fairly familiar with the framework, and thought that it provided most of the functionality and capabilities we needed for the application. Throughout the entirety of the project we have implemented end-to-end testing (e2e) to test each of the functionalities our application introduces to ensure that each and every component is covered and error handled properly. This process of implementation followed by thorough testing allowed us to mitigate several bugs prior to pushing it into the main codebase.

Problems Encountered

One of the issues that we encountered early on in the development of the project was the best way to parse a CSV file into an easily readable, accessible, and indexable table. This we were hung up on for a couple of weeks before figuring out the best method of attack. Eventually we landed on a design that parsed the CSV file into a table that was easily read and displayed through the usage of a mutable dataservice that could handle, create, read, update, and delete functions. Similarly, we struggled with the interplay of the two tables that contain student feedback and student assignment statuses respectively. Both of these issues were solved in a couple of weeks with advice from our advisor Victor Norman and various research online.

Application Testing

Throughout the duration of our project, we implemented end-to-end development using Cypress. This allowed us as developers to streamline the testing of many anticipated user stories. Some of these include, importing and exporting the CSV file to and from the filesystem, using each of the buttons, and several other error handling type events that could potentially occur while using our application. We also intermittently did some in person user testing where we provided the application to a user and instructed them to complete certain tasks. From here we observed the difficulty required to perform each task and asked them for their objective feedback on their experience using the application.

Demonstration Video

Please view the app demonstration here. (gotta update the link after pr)

Potential Developments

Some future developments that could be implemented include persistent storage of progress. As the application currently stands, if the user wanted to close the application and return to it later, or revisit their progress they would see an untouched application state on return. Going forward saving partial progress would be ideal, this would save feedback items, state of the table, assignment title, and other relevant data or metadata. Some other minor application perks that could be added include, auto-deductions based on submission timestamp, and sorting columns alphabetically by toggle clicking the headers. Beyond this, other modifications may be in the realm of preference, a persistent settings page that could include custom styling, default feedback options, and others.

Acknowledgments

Special thanks to Professor Victor Norman for advising throughout the project, and coming up with the idea and a proposed plan of attack. Thanks to Charles Kornoelje, Noah Cummisford, and William Terpstra for volunteering as application testers, and providing feedback pertinent to the app's development.