

# Google Classroom Integration With docASSIST

By Cory Tapply and Trevor Valcourt



Google Classroom Integration With docASSIST  
An Interactive Qualifying Project Report Submitted to the Faculty of the  
WORCESTER POLYTECHNIC INSTITUTE  
in partial fulfillment of the requirements for the Degree of Bachelor of Science

By

---

Cory Tapply

---

Trevor Valcourt

Submitted on

Approved:

---

Professor Neil T. Heffernan, Advisor

---

Cristina Heffernan, Advisor

# Table of Contents

<b>List of Figures</b>	<b>4</b>
<b>Abstract</b>	<b>5</b>
<b>Acknowledgements</b>	<b>6</b>
<b>Authorship</b>	<b>7</b>
<b>Introduction</b>	<b>8</b>
<b>A Term Work</b>	<b>9</b>
Bug Fixes	9
Documentation	9
Website Development	10
<b>B Term Work</b>	<b>11</b>
PDF Draft Saving	11
Google Classroom Integration	13
Bug Fixes	14
<b>D Term Work</b>	<b>15</b>
Pushing Grades Implementation	15
Website Overhaul	17
<b>Conclusion and Future Development</b>	<b>19</b>
Future Work	19
<b>Appendix</b>	<b>20</b>

# List of Figures

Figure 1. Object model of the Rubric

Figure 2. Screenshot of old docASSIST website

Figure 3. Interface for saving a draft to a folder

Figure 4. Autocomplete when selecting draft folder

Figure 5. Draft versioning in PDF name

Figure 6. Assign with Google Classroom button

Figure 7. New interface for creating assignments to send to Google Classroom

Figure 8. Assignment pushed from docASSIST to Classroom

Figure 9. Assignment creation interface with max points field

Figure 10. New drop-down panel for pushing a grade to Google Classroom

Figure 11. Latest docASSIST homepage

# **Abstract**

docASSIST is a tool originally created by WPI students as a Major Qualifying Project (MQP) and has been in continual development as an Interdisciplinary Qualifying Project (IQP). The projects are under the advisement and mentorship of the WPI Assisments team. The purpose of docASSIST is to help teachers give feedback to their students faster and more efficiently. Currently, many teachers are overworked with the amount of grading and revising of students work they have to do. docASSIST solves this problem by giving teachers a platform to evaluate and give feedback to students efficiently. The purpose of this IQP was to continue developing docASSIST. Areas of development include refactoring to fix major bugs, draft saving, Google Classroom integration, and pushing grades to Classroom.

# Acknowledgements

We would like to first acknowledge the previous developers of docASSIST, Nick McMahon, Sam La, Jean Marc Touma, Christian Roberts, Zi Wang, Zachary Armsby, and Gianluca Tarquinio for their work on docASSIST. In addition, we would like to thank our advisors Cristina and Neil Heffernan for their guidance throughout the project as well as the Assessment's Lab at WPI for their work in promoting docASSIST. We would also like to acknowledge Theresa Inzerillo who worked on docASSIST as an independent study alongside us. Lastly, we would like to thank any future docASSIST workers for their contributions to this project.

# **Authorship**

This paper was written by Cory Tapply and Trevor Valcourt using Zachary Armsby's paper as an example.

# Introduction

docASSIST was already a well established product by the time we began development on it for our IQP. The project has changed hands multiple times over the years while under the same advisement of the Assistments team. docASSIST currently has 7,500 active users according to the Google Doc Store. Our focus with docASSIST was originally to fix some of the bugs in the program. This shifted towards an integration with Google Classroom as we learned more about our user base and realized many of them also use Classroom for managing assignments and grading student papers. We also wanted to implement features that teachers have personally requested, such as draft saving, to keep docASSIST useful to our users.

Our main goal with docASSIST was to better cater to what our users were using. We want to make their experience with our project positive by creating features that were relevant to the way they use docASSIST. This is the main motivation behind Google Classroom integration, as it allows teachers already using docASSIST to now create assignments with rubrics and add them to their Classroom right inside a document. While we weren't able to a seamless transition between docASSIST and Classroom, we were successful in creating a few basic functions and doing much of the research on how to expand this in the future.



# A Term Work

We were introduced to the project and codebase by past students Christian Roberts, Gianluca Tarquinio, and Zachary Armsby. Our division of completed work during A term could be divided into the following categories: Quality of Life Changes, Documentation, and Website development.

## Bug Fixes

To become more familiar with the workflow of docASSIST we began our IQP with small bug fixing. We fixed one bug with the link not working properly after grading a student's paper. This was frustrating for teachers because they would often want to open their gradesheet after making a grade, and without this link they would have to manually find and open the gradesheet. On top of this we also updated some of the formatting on the gradesheet to be more readable. Another bug we fixed was to make rubrics automatically attach to the document after choosing the Select button. Previously, the rubric wouldn't actually appear on the document until you clicked "Update Summary" for the first time. The last major bug we fixed was an issue with the rubric preview improperly displaying. This bug was notorious amongst past developers and caused rubric categories to disappear in the rubric preview. This was a major complaint from teachers as they liked to be able to see which rubric they were attaching in the selection screen of the rubric manager.

## Documentation

During A term we also worked on creating new documentation for future docASSIST developers. Most notably, we created a "How To" guide (See Appendix A) for publishing changes to the Google Store. We created a comprehensive step-by-step guide for any new developer to be able to make their changes live as this was something we had to figure out ourselves. We also began to create an object model for the entire system. This quickly became a frustrating and arduous task as the system was extremely complicated, with some scripts being unnecessarily large. The models exposed large amounts of copy and pasted code throughout the system and highlighted why development was often slow and difficult. We proposed a large refactoring effort to address these underlying code problems, but this is more discussed in the "Future Development" section of the paper.

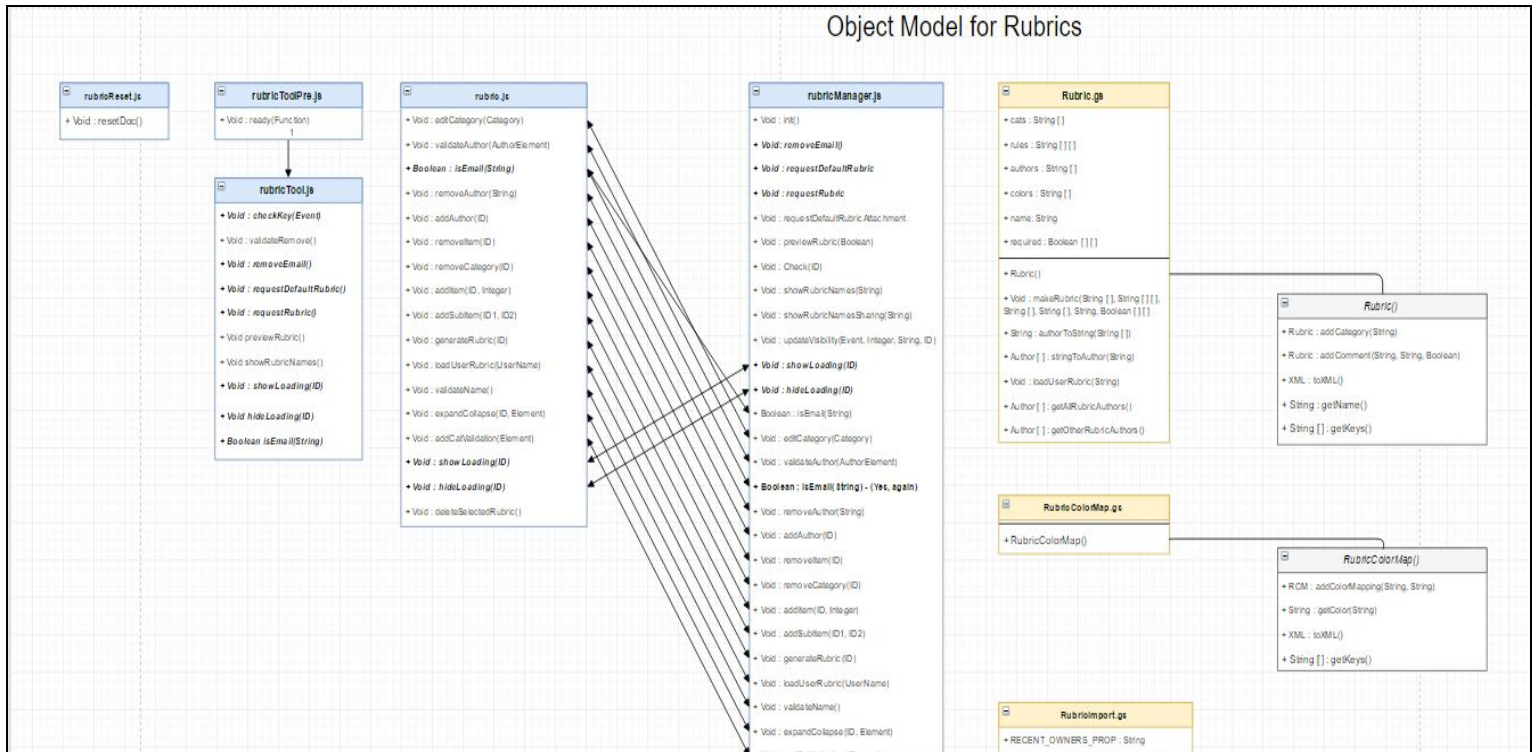
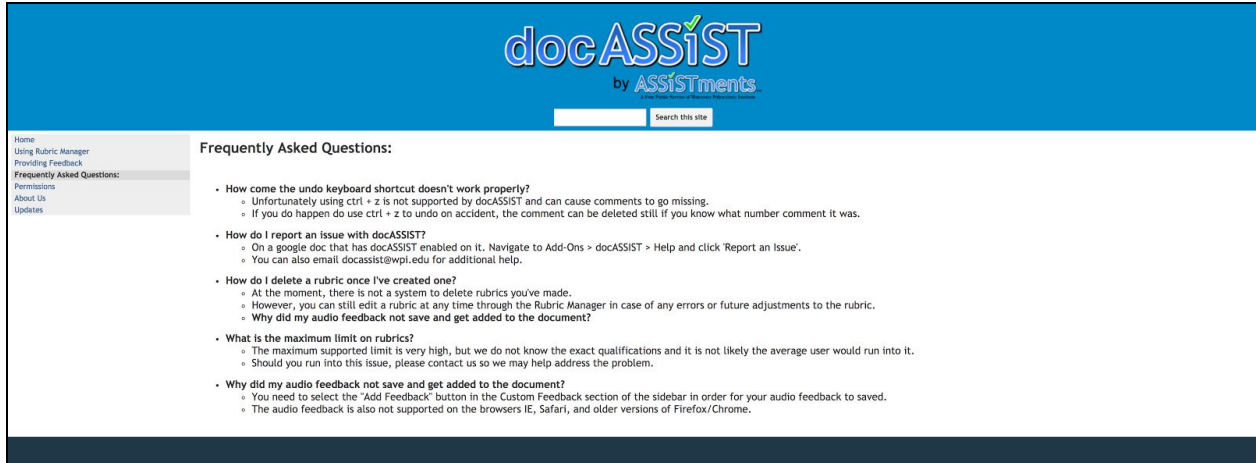


Figure 1. This represents the object model for only the Rubric related functions. It isn't exhaustive, however, the arrows between objects represent copy and pasted code, and function names in bold are functions whose names also appear in other scripts.

## Website Development

The rest of our time in A term we spent working on the docASSIST website. From receiving questions from teachers throughout the term, we created a new Frequently Asked Questions page to address questions common amongst teacher. We created an updates page with a list of our development changes from each term so if a user wanted to know exactly what we've changed they could see this. We also updated some of the older images and explanations that were of older versions of docASSIST and no longer relevant. This included the pages on the rubric manager and providing feedback.



*Figure 2. A screenshot of the Frequently Asked Questions page pages on the old docASSIST website.*

Overall, A term was a successful introduction to working on docASSIST. We were able to make real contributions and fix bugs that were frustrating our users. Upon doing documentation and sifting through the code, we were able to identify some major flaws in the overall program design. Addressing these issues was not something we were able to do during our time with docASSIST as we had pressing features to implement. Our advisor's main goal for us was the Google Classroom integration which we began to work on B term.

## B Term Work

This term we added a few key features of docASSIST that can heavily alter the workflow for teachers who use Google Classroom. Notably PDF draft saving and Google Classroom integration.

### PDF Draft Saving

This is a feature that saves the content of the document in its current state as a non-editable PDF. This was requested by a teacher from Shrewsbury High School over the summer. Gianluca had started to implement the feature but it was incomplete. At first the user was prompted to input a name for the PDF to be saved as inside of the /docASSIST/drafts folder on Google Drive. This caused the user to have all of their PDF copies in the same folder with little organization. We changed this to instead ask the user to enter the name of a folder to save the draft into.

The screenshot shows a dialog box titled 'Save a non-editable draft' with a question mark icon in the top right corner. Below the title is a text input field containing the text 'Assignment One'. At the bottom of the dialog is a button labeled 'Update and Save Draft'.

*Figure 3. Interface for saving a draft to a folder.*

This allows the user to save all drafts related to a given class or assignment in the same folder for added organization. We also made this field remember what was last entered and autofills this for the next time they want to save a draft so they do not have to remember what the name of the folder they last saved to is named. To add to this, when changing the text in the box, we show a list of existing folders so the user can see what folders already exist.

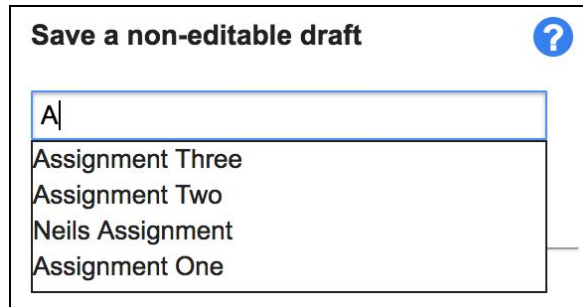


Figure 4. Example of using auto-complete when selecting a folder.

When we made this change we also changed what the name of the draft is when it's saved. We now take the name of the current document and add a version number, so if the user saved multiple copies they will not overwrite previous drafts. We think this is a good solution since often times, especially when using Google Classroom, the students name is part of the name of the document and will also be in the name of the draft as well.

My Drive > docASSIST > Drafts > Assignment One ▾







Name ↑	Owner
 Gettysburg Address Essay-v1 	me
 Gettysburg Address Essay-v2 	me
 Gettysburg Address Essay-v3 	me

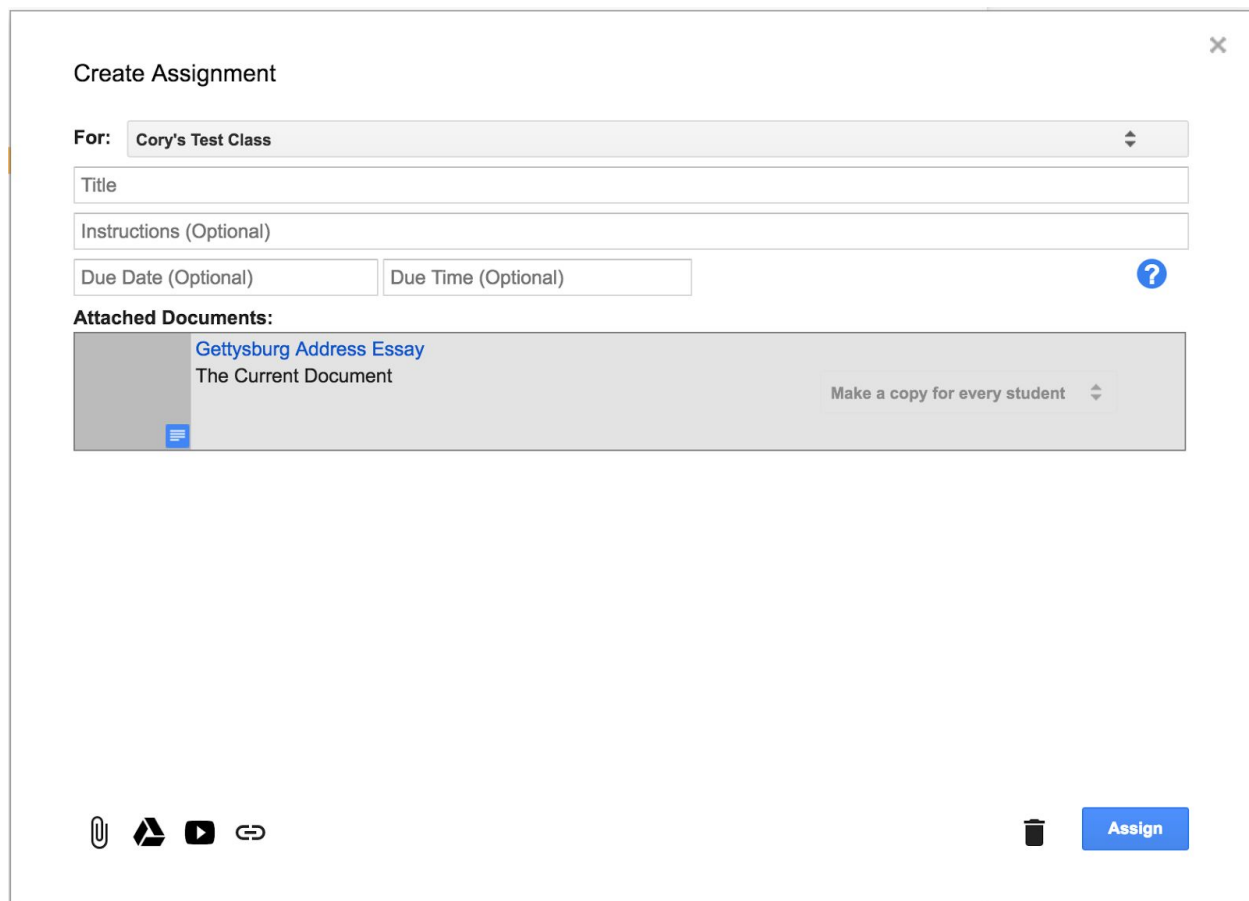
Figure 5. When you save multiple drafts of the same document, they automatically get a version number in the name of the draft.

## Google Classroom Integration

Google Classroom integration was the big goal of our IQP. The addition of this feature speeds up the workflow for teachers by a significant amount. Instead of needing to create a document and add a docASSIST rubric, then in a new tab open Google Classroom and create the assignment, you can now do both steps straight from docASSIST. In the Rubric Manager we have added a button “Select and Assign with Google Classroom” which will attach the selected rubric to the document and open a new dialogue to create an assignment for Google Classroom.



*Figure 6. New “Assign with Google Classroom” button in the Rubric Manager.*

A screenshot of a "Create Assignment" dialog box. At the top right is a close button (X). Below the title "Create Assignment" is a dropdown menu labeled "For:" with the text "Cory's Test Class". Below that are three text input fields: "Title", "Instructions (Optional)", and "Due Date (Optional)". To the right of the "Due Date" field is a "Due Time (Optional)" field and a blue question mark icon. Below these fields is a section titled "Attached Documents:" containing a single document entry: "Gettysburg Address Essay" with the subtitle "The Current Document". To the right of this entry is a dropdown menu with the text "Make a copy for every student". At the bottom left are icons for attaching files, folders, videos, and links. At the bottom right is a trash can icon and a blue "Assign" button.

*Figure 7. This is the new interface we created to allow users with Google Classroom accounts to make assignments here and attach their document with the associated rubric. We wanted to make this interface as close to the one on Google Classroom as possible so it will look familiar to them.*

We have made this very similar to how the create assignment window is on Google Classroom to make sure it is as familiar to as possible for our teachers. We automatically add the current document to

the assignment with the setting to “Create a copy for each student” which causes Google Classroom to make an exact copy of the document with the docASSIST rubric and give ownership to each student. We do this because it allows the teacher to distribute a document with the rubric already attached as a template for each student. The teacher may also add additional documents or YouTube videos with the Picker from the buttons on the bottom left of the dialogue and set these to view only, edit or make a copy for all students if applicable.

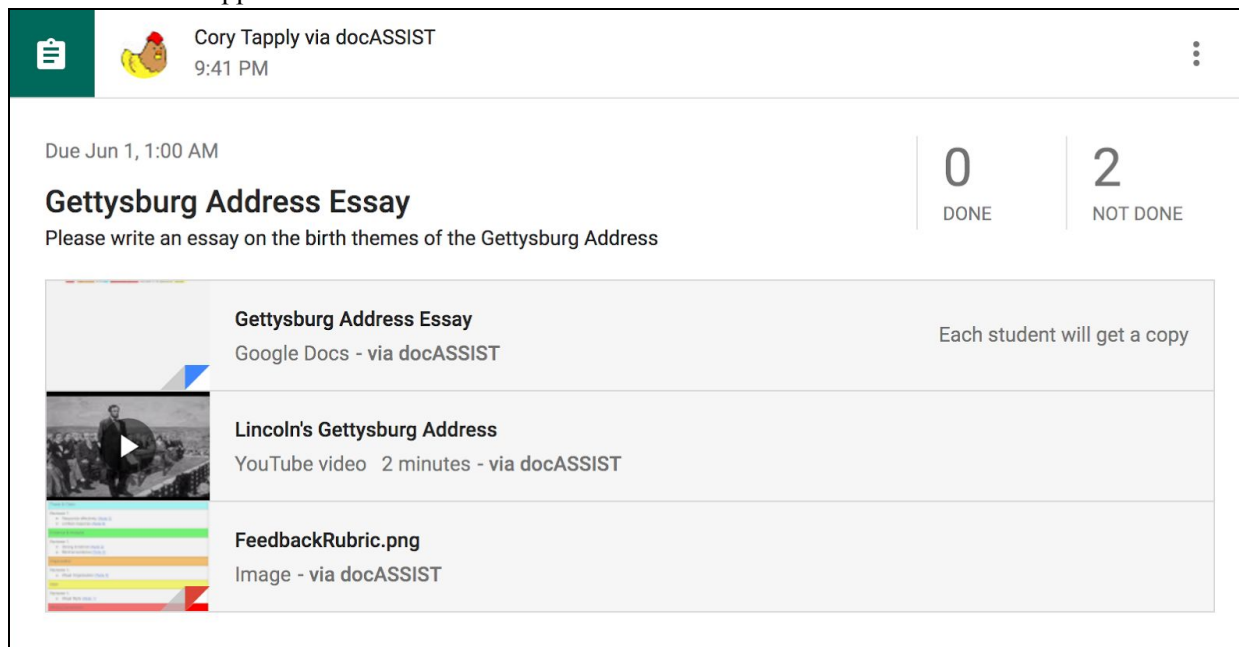


Figure 8. Example of an assignment created within docASSIST and sent to the teacher's Google Classroom.

We feel that this feature will become a core component to the workflow for Google Classroom users and this combined with the ability to send grades directly to Classroom will allow for docASSIST to have complete integration with Google Classroom.

## Bug Fixes

This term we also spent some time working on fixing some more complex bugs that we were unable to figure out while still learning the code base in A-term. The biggest being, when a user added a rubric, then changed rubrics and added another one, you would be left with multiple rubrics in the document body. This was also the case for the summaries table as well. We spent many hours trying to understand how these components could be easily replaced in the document body. We found a supposed solution to our problem within the official documentation of the Google Docs API but in reality the NamedRange elements do not work as they are documented causing us to lose many hours with no working solution. In the end, we were able to fix the issue with string matching which is not an ideal solution but it is one that solves our issue of having multiples of any of the tables created by docASSIST.

## D Term Work

After taking a break from docASSIST during C term, we returned again in D term with the main goal of pushing grades to Google Classroom. We also spent a lot of time redesigning the docASSIST website to a more modern look and feel with the \*New\* Google Sites builder. Our goal in D term was to finish the grade pushing feature and leave docASSIST in a stable state for the next set of students who will continue development.

### Pushing Grades Implementation

We spent some time in B term attempting to push grades to Google Classroom. From the documentation, we discovered that the maximum points an assignment is worth can only be set on assignment creation so this is something we need to add to our assignment creation interface. To push an individual grade, we ideally wanted a new grade field associated with rubric categories on a Rubric object in code. This proved incredibly difficult to implement and showed us just how volatile the underlying data model is. Attempting to add a new field was difficult because the majority of the code was dependent on a very specific Rubric object definition, where the order of the fields mattered when processing the object. There were many, many places (often with repeated code) where a Rubric object is processed so trying to track down each one to also process a grades field was a never ending task. We also encountered a compatibility issue with legacy Rubric objects once we did add this new field. This caused problems loading rubrics without this data from the database and effectively rendered docASSIST unusable. We knew it was not worthwhile to try to force this implementation to work, so we tried a simpler approach in D term to still obtain some functionality. We discuss our thoughts in detail about addressing this underlying data structure flaw in the Future Works section.

In D term, we started by addressing the max point field. This was a simple addition to the interface we already created to create assignments for Google Classroom.

Figure 9. The assignment creation interface now with the max points text field.

To actually send grades, we created an account based drop-down in the original Grades panel. If a user has Google Classroom authorization, then this drop-down will appear and give them the option to grade a student's assignment. If the user is not associated with Google Classroom, this interface will simply be hidden and they can grade normally using the gradesheets.

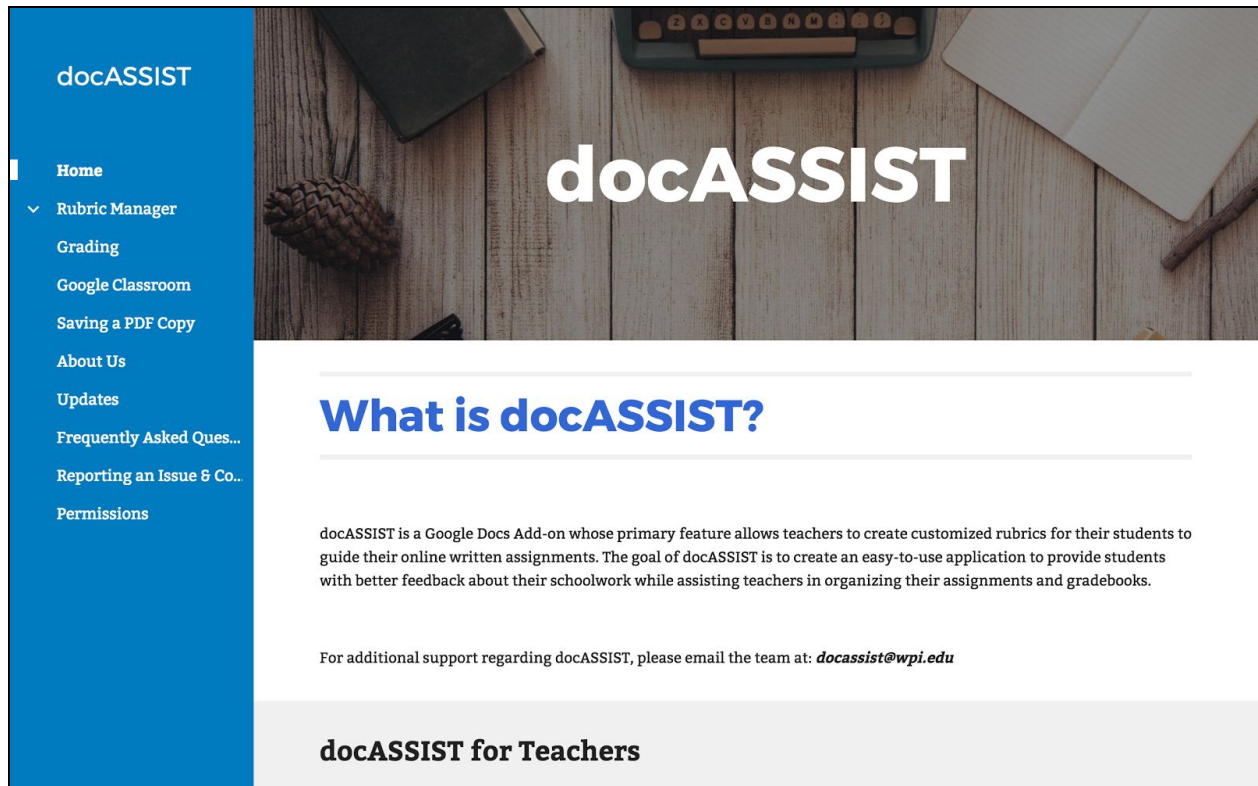
Figure 10. The new account based drop-down panel which allows the grader to select the classroom, assignment, student and enter the final grade to send to Google Classroom.



A Google Classroom user has the ability to select their course, assignment, and student from a set of drop-down boxes. The selected course and assignment are remembered from their last selections for convenience to the user. The last box is the actual grade to give the student. There is an automatically calculated sum of values set in the rubric category boxes above in the total box. The grade can be manually set and is listed as a fraction out of the max points associated with the selected assignment. In the future, it would be helpful if docASSIST could automatically detect which student is associated with the current paper for even faster grading. We explored doing this by embedding the student's name in the document properties, however, when Google Classroom makes a copy for the student these properties don't copy over so the name is missing. This is okay for the course and assignments because we can save these in the user properties of their account. Doing the same thing for the student's name doesn't make sense because you would simply be remembering the last student you graded and this simply isn't helpful. One other potential solution could be to parse the name of the document, because the student's name should be appended at the end. This solution could work, but it isn't ideal because the teacher could change the name of the document in which case we wouldn't know the student's name and they would still have to manually select the student. We were unable to test this solution as we simply ran out of time.

## Website Overhaul

The original docASSIST website was created with the old Google Sites website builder. While it was simple to add information and create pages, it had an old look and feel and many simple features were frustrating to use. For example, in the old builder, it was near impossible to include an image with the exact size and position you wanted. In addition, a lot of the textual information as well as screenshots were simply outdated due to development of docASSIST over time. We want the website to be helpful to our users and address as many questions they may have about using our application.



*Figure 11. A screenshot showing the latest docASSIST website homepage using the new Google Sites builder.*

As a team, we decided a major overhaul was a worthwhile investment and fortunately, Google recently released their new Google Sites builder. It was an executive decision by the project advisors to keep the website hosted by Google for simplicity. With the new Google Sites, we re-wrote almost every information page from scratch since most of it was outdated. This includes all of the Rubric Manager pages, the homepage, and pages on our new features such as Google Classroom and draft saving. One benefit of the new Google Sites builder multi-platform support. By default, websites are correctly scaled to PC, tablet, and mobile devices which is an important feature by modern day standards. In general, the new site builder has a much cleaner interface and many more customization options. A lot of the areas lacking in the old builder, such as adding images as mentioned above, are no longer issues and it is very easy to quickly create and edit sections of the website. We believe this new platform will be much more inviting to our users and easy for future developers to edit. All of the information and screenshots are now up to date with the latest version of docASSIST at the time of writing this paper.

# Conclusion and Future Development

Overall, we are happy with our additions to the docASSIST add-on. Many of the changes we were able to implement are features directly requested by some of our users. With the addition of Google Classroom, we are able to support a whole new array of features for our current users and hopefully draw in new users as well. We were able to fix many of the bugs that detracted from docASSIST and cleaned up parts of the code that were repeated or no longer being used. We've noticed significantly less messages from users reporting issues they encountered while using docASSIST since we started and we believe this is a good indication that our work has made a positive impact.

## Future Work

While docASSIST is functional in its current state, we believe that the current codebase is not sustainable for a successful project in the future. As mentioned in the "Pushing Grades" section of the paper, docASSIST suffers from some serious design flaws that hinder its progress and frustrate developers. When we first began developing docASSIST, it was apparent that much of the code was slapped together without much thought about future collaborators. This has led to many instances of copy and pasted code amongst multiple files, causing new additions to be difficult and forced solutions unless we were making brand new independent scripts. Some of this repeated code and strange program design can be seen in the relational models we began to create. There was a point where we stopped working on these models because it was very time consuming and obvious that there was a serious issue. We brought these concerns to the attention of our advisors, but for our time with docASSIST they wanted to see features completed over code re-work.

In saying this, we recommend that before more work is done with docASSIST there needs to be a major re-write of the program. In addition to this, there needs to be a better system for developing a Google Doc Add-on in a team and research in how other people do this would be beneficial. Zachary Armsby's contribution was a huge help to the previous workflow, however there needs to be well defined coding practices and a standard development environment so it is easier to work together. We believe it would be worthwhile to start the application from the beginning again now that there have been a few iterations of the program. More care should be taken into designing the system before programming even begins, because in its current state the underlying data model is flawed. The next major feature for docASSIST will most likely be a peer review system. With the current codebase, it will not be an easy task to implement this. Many of the features we have been working on were not considered during the project's infancy and this is beginning to cause problems with development. It would be a time consuming task to revamp the entire application, but with the knowledge the docASSIST team has gained from a few years of work it would greatly enhance the quality of docASSIST for both our users and future developers.

# Appendix

docASSIST Website: [docassist.assistments.org](http://docassist.assistments.org)

docASSIST Backend: [github.com/neiltheffernaniii/docASSIST](https://github.com/neiltheffernaniii/docASSIST)

docASSIST Frontend: [github.com/neiltheffernaniii/docASSISTAppsScriptProject](https://github.com/neiltheffernaniii/docASSISTAppsScriptProject)

Google Chrome Store: <https://chrome.google.com/webstore/detail/docassist>