DocASSIST: Google Docs Add-On Development

Fabio Borges
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/iqp-all

Repository Citation

This Unrestricted is brought to you for free and open access by the Interactive Qualifying Projects at Digital WPI. It has been accepted for inclusion in Interactive Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.
DocASSIST:
Google Docs Add-on Development

By Fabio Borges
DocASSIST: Google Docs Add-On Development
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

________________________________________
Fabio Borges

Submitted on

Approved:

________________________________________
Professor Neil T. Heffernan, Advisor
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents</td>
<td>3</td>
</tr>
<tr>
<td>List of Figures</td>
<td>5</td>
</tr>
<tr>
<td>Abstract</td>
<td>6</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>7</td>
</tr>
<tr>
<td>Authorship</td>
<td>8</td>
</tr>
<tr>
<td>Introduction</td>
<td>9</td>
</tr>
<tr>
<td><strong>Feature Additions</strong></td>
<td>10</td>
</tr>
<tr>
<td>Auto Update Summary</td>
<td>10</td>
</tr>
<tr>
<td>Update Summary Reminder</td>
<td>11</td>
</tr>
<tr>
<td>User Interaction with Author Revise Sidebar</td>
<td>12</td>
</tr>
<tr>
<td><strong>Google Classroom Integration</strong></td>
<td>15</td>
</tr>
<tr>
<td>Auto Rubric Attachment on Classroom Documents</td>
<td>15</td>
</tr>
<tr>
<td>Course, Assignment, and Student Detection on Grading Sidebar</td>
<td>17</td>
</tr>
<tr>
<td><strong>Bug Fixes</strong></td>
<td>20</td>
</tr>
<tr>
<td>Author Revise Audio Player</td>
<td>20</td>
</tr>
<tr>
<td>Classroom Rubric Attachment and Non-DocASSIST Assignments</td>
<td>22</td>
</tr>
<tr>
<td>Unexpected Behavior with Update Summary and Table of Contents</td>
<td>22</td>
</tr>
<tr>
<td><strong>User Feedback</strong></td>
<td>23</td>
</tr>
<tr>
<td><strong>Future Development</strong></td>
<td>23</td>
</tr>
<tr>
<td>Student and Teacher Separation</td>
<td>23</td>
</tr>
<tr>
<td>Review and Grading as Tabs</td>
<td>24</td>
</tr>
<tr>
<td>Display of Feedback in Document and Author Revise</td>
<td>25</td>
</tr>
<tr>
<td>Time Spent Loading Audio Feedback in Author Revise</td>
<td>26</td>
</tr>
<tr>
<td>Color Coding for Positive and Negative Feedback</td>
<td>27</td>
</tr>
<tr>
<td>Extending Classroom Rubric Attachment</td>
<td>27</td>
</tr>
<tr>
<td>Additional Logging in Database</td>
<td>27</td>
</tr>
<tr>
<td><strong>Conclusion</strong></td>
<td>29</td>
</tr>
<tr>
<td><strong>Appendix</strong></td>
<td>30</td>
</tr>
<tr>
<td>DocASSIST Website</td>
<td>30</td>
</tr>
<tr>
<td>Front-End Repository</td>
<td>30</td>
</tr>
</tbody>
</table>
Back-End Repository
List of Figures

Figure 1: Graph of Update Summary Execution Times With and Without Auto Update

Figure 2: Red Reminder for User to Update their Summary to see Changes

Figure 3: Author Revise Audio Feedback Player

Figure 4: Author Revise Audio Feedback Display Bug

Figure 5: Javascript Function to Attach the Correct Rubric to a Student’s Submitted Document

Figure 6: Javascript Function to Determine the Assignment Associated With a Document

Figure 7: Javascript Function to Determine the Course Associated With a Document

Figure 8: Grading Dropdown Menus for Course, Assignment, and Student

Figure 9: Grading Automatic Selection for Course, Assignment, and Student

Figure 10: Author Revise Bugged Audio Player

Figure 11: DocASSIST Add-On Menu

Figure 12: Author Revise Execution Time for Text and Audio Feedback
Abstract

DocASSIST is a Google Docs Add-On written by various WPI students through several years as part of several IQP/MQP projects with the advisement of the WPI Assistments team. The goal of DocASSIST is to provide tools for teachers to more easily attach rubrics and feedback when grading student assignments. Grading written assignments is often time-consuming for many teachers, especially if they have multiple classes' worth of assignments to grade all at once. Thus, DocASSIST intends to cut down on the time teachers have to spend while grading on Google Docs. The purpose of this IQP is to continue the development of DocASSIST to improve the user experience for teachers and provide further Google Classroom integration.
Acknowledgements

I would like to acknowledge all the developers who contributed to DocASSIST in the past so that I could have a solid baseline with which to improve upon, especially Trevor Valcourt and Cory Tapply who aided me in understanding the codebase for DocASSIST and setting up the appropriate development environment. My advisor Neil Heffernan as well as Thanaporn Patikorn provided me a lot of guidance, specifically by discussing what needs to be improved about DocASSIST and helping evaluate the most suitable solution for a particular issue among several that I had thought of. Lastly, I would like to acknowledge Skyler Kim for continuing the development of DocASSIST starting in the A Term of 2018.
Authorship

This report was written by Fabio Borges using Zachary Armsby’s report from 2016 as an example.
Introduction

Although DocASSIST has gone through several years of development already, beginning the process of my own development was not a simple task. The documentation for DocASSIST was limited for significant portions of the front-end code and especially the back-end code. I was provided notes regarding the front-end code by Trevor and Cory but for the most part I was figuring out the back-end code on my own. Consequently, I spent a couple weeks setting up the development environment and understanding the code before I was confident in my ability to make meaningful contributions to DocASSIST. More specifically, I familiarized myself with the mechanics of Google Apps Script along with the relevant Apps Script APIs used in DocASSIST.

Once situated, I started investigating and working on improvements to DocASSIST meant to facilitate a teacher’s ability to easily and quickly add rubrics and feedback to a student document. Although DocASSIST already makes the task much easier, some features of the add-on could have been better for achieving this goal and others features were not even functional. Also, I worked on furthering the Google Classroom integration that Trevor and Cory had started since the Classroom API had specific capabilities not currently being utilized by DocASSIST that could make life easier for teachers using Google Classroom. In addition, with the help of Neil and Thanaporn, I was able to identify other areas of DocASSIST that could be improved upon in future development.
Feature Additions

Auto Update Summary

After attaching a rubric to a document, teachers have the ability to highlight text in the document and color code it according to several categories on their rubric using DocASSIST’s Review sidebar. After highlighting all the text they want, the teacher can click the Update Summary button at the bottom of the sidebar. This will update the automatically generated table at the bottom of the document with all the feedback the teacher had just added (an example of which can be seen in Figure 2). A potential problem with this feature is that teacher feedback will not be added to their summary the moment they add that piece of feedback to the document. Also, scrolling to the bottom of the sidebar is required in order to notice the Update Summary button. Therefore, it may not be obvious to user what they need to do to update their rubric summary.

As a potential solution to this problem, I removed the Update Summary button entirely and had DocASSIST automatically update the summary every time the teacher added any piece of feedback. Like this, teachers could add feedback as usual but they would no longer have to worry about updating the summary or how to update it. Although this feature functioned as intended, I noticed a crippling disadvantage to this approach that I had not considered before implementing the automatic updates into DocASSIST.

The below graph in Figure 1 plots the number of feedback notes that have currently been added to a document (from 0 to 30) against the time taken in seconds for DocASSIST to add one new feedback note to the document (from 0 to 4 seconds) for two different implementations of the automatic updates and the previous implementation with no automatic updates. The time taken to add new feedback with manual updates stayed relatively constant at just above one second, however, the time taken for automatic updates trended upward up to three and a half seconds once the document contained 30 feedback notes. This means that eventually the automatic updates would become bothersome for some teachers since they would be spending three seconds or more adding another feedback note. Although that does not sound like a lot of time, it stacks up when teachers have many assignments to grade or a lot of feedback to give. Consequently, this feature was rolled back since the convenience was not worth the gradually increasing execution time of adding new feedback.
Update Summary Reminder

In response to automatic summary updates not being optimal, I took a different approach by having DocASSIST notifying the user with red text in the heading of the rubric summary whenever they have outstanding feedback notes that are not currently reflected in the summary. The red text will remind the user that they need to click the Update Summary button in order to see their new feedback. Although not as convenient as automatic updates, this approach was enough to address the problem of a teacher scrolling down to the summary wondering why their new feedback does not appear. Figure 2 below shows an example of such a notification.
DocASSIST also has an Author Revise sidebar meant for students to have an easy way of visualizing the feedback that would be shown in the rubric summary without having to scroll down to the bottom of the page. Teachers also have the ability to leave audio feedback, so Author Revise is also useful for hearing that feedback without having to open a new tab by clicking a link to the audio file in the summary.

However, this sidebar was not very interactive since students were not able to easily associate a feedback note in the sidebar with the section of their assignment that it applies to. The feedback notes are also organized by rubric category as opposed to the order in which they occur in the document. To amend this, I added a feature to this sidebar that would allow a student to click any feedback note and automatically scroll to the page where that feedback is located along with a blue question mark that a student can hover over to reveal a text box containing the text that corresponds to that feedback note. For audio feedback, the audio player is automatically loaded into the sidebar to make it clear that students can listen to audio feedback straight from the sidebar. An example of these additions can be found below in Figure 3. But sometimes there is a bug with the audio player where the progress bar does not appear unless the user clicks on the speaker button twice, which can be seen in Figure 4.
Your Feedback Notes

[+] Thesis & Claim (1)

[+] Evidence & Analysis (2)

[–] Organization (2)

1. Strong organization
2. Audio feedback

Figure 3
Your Feedback Notes

[+] Thesis & Claim (1)
[+] Evidence & Analysis (2)
[¬] Organization (2)

1. Strong organization
2. Audio feedback

Figure 4
Google Classroom Integration

Auto Rubric Attachment on Classroom Documents

When deciding what rubric to attach to an assignment using DocASSIST’s rubric manager, teachers have the option to also create a new Google Classroom assignment at the same time which would send each a student a template document containing the attached rubric. This saves time for teachers who utilize Google Classroom but the problem this feature was that whenever a teacher would want to grade a student’s submitted assignment, they would need to attach that rubric in the rubric manager all over again, making the task unnecessarily tedious.

According the Apps Script API, documents can have a set of properties or key-value pairs associated with them and this is how DocASSIST “attaches” rubrics to documents. Specifically, DocASSIST save document properties representing the name, owner, color coding, and XML data for a rubric. However, the template documents given to each student after creating a new Classroom assignment are treated as separate documents and DocASSIST is not allowed to modify the properties of these documents. Therefore, the problem occurs because the necessary document properties are not carried over to the templates that the students receive.

Similar to documents, individual users can also have their own set of properties associated with them. So to solve the issue, I started by having DocASSIST save the same rubric properties to the teacher’s user properties and uniquely identifying this set of rubric properties according the ID of the Classroom assignment that was just created. Then, whenever the teacher tries to open the Review sidebar on a student’s submission, DocASSIST will pull the correct rubric data from the teacher’s properties. The course that a submission belongs to is determined by searching for a teacher group email address that matches one of the addresses in the document’s list of editors (shown in Figure 6). The assignment that a submission belongs to is determined by searching through all submission objects for a specific course recorded by the teacher’s Classroom and returning the assignment ID of the submission that has the same file ID as the ID of the document that the teacher is currently looking at (shown in Figure 7). Once these two are obtained, the student document’s properties are set according to the same properties associated with that teacher (shown in Figure 5).
/**
 * Sets the document properties for the current document if the document
 * is associated with a student submission from Google Classroom
 */
function getRubricFromAssignment() {
    var docId = DocumentApp.getActiveDocument().getId();
    var userProps = PropertiesService.getDocumentProperties();
    var docProps = PropertiesService.getDocumentProperties();
    var correctCourse = getCorrectCourse(DocumentApp.getActiveDocument().getEditors(), Classroom.Courses.list().courses);
    if(correctCourse & Classroom.Courses.Teachers.getActiveUser().getEffectiveUser().getEmail()) {
        var correctCourseWork = getCorrectCourseWork(correctCourse, docId);
        if(correctCourseWork & userProps.getProperty(correctCourseWork.id + " ASSIGNmentsRubric"){
            docProps.setProperty("ASSIGNmentsRubricName", userProps.getProperty(correctCourseWork.id + " ASSIGNmentsRubricName");
            docProps.setProperty("ASSIGNmentsRubricOwner", userProps.getProperty(correctCourseWork.id + " ASSIGNmentsRubricOwner");
            docProps.setProperty("ASSIGNmentsRubricColorMap", userProps.getProperty(correctCourseWork.id + " ASSIGNmentsRubricColorMap");
            docProps.setProperty("ASSIGNmentsRubric", userProps.getProperty(correctCourseWork.id + " ASSIGNmentsRubric");
        } 
    }
}

function getCorrectCourseWork(course, docId) {
    var courseWorks = Classroom.Courses.CourseWork.list(course.id).courseWork;
    for(j = 0; j < courseWorks.length; j++) {
        var courseWork = courseWorks[j];
        var submissions = Classroom.Courses.CourseWork.StudentSubmissions.list(course.id, courseWork.id).studentSubmissions;
        if(submissions){
            for(k = 0; k < submissions.length; k++) {
                var submission = submissions[k];
                if(submission){
                    var attachments = submission.assignmentSubmission.attachments;
                    if(attachments){
                        var file = submission.assignmentSubmission.attachments[0].driveFile;
                        if(file){
                            if(file.id == docId)
                                return courseWork;
                        }
                    }
                }
            }
        }
    }
    return null;
}
function getCorrectCourse(editors, courses) {
    for(i = 0; i < editors.length; i++) {
        for(j = 0; j < courses.length; j++) {
            if(editors[i].getEmail() === courses[j].teacherGroupEmail)
                return courses[j];
        }
    }
    return null;
}

Course, Assignment, and Student Detection on Grading Sidebar

In the Grading sidebar teachers can assign point values to each rubric category and add a grade table to the document once they are done, but they also have the option to send the grade total to Google Classroom. A teacher sends a grade to Classroom by selecting the correct course, assignment, and student from three dropdown menus shown in Figure 8. Despite seeming simple enough, the use of these three dropdowns relies on the user selecting the correct information. This leads to situations where teachers might make a mistake and send a grade to the wrong course, assignment, or student since DocASSIST provides the teacher’s entire Classroom in the dropdowns.

Borrowing from the three code snippets in the Figures above, DocASSIST is already able to determine the course and assignment associated with a document. So by adding one more function similar to the one in Figure 6, DocASSIST will be able to narrow down the correct course, assignment, and student automatically when sending a grade to Classroom without relying on input from the user. After implementing this feature, the three dropdowns were replaced three lines of bolded displaying the appropriate Classroom information for that document as shown in Figure 9.
Figure 8
Figure 9
Bug Fixes

Author Revise Audio Player

Before implementing the changes to the Author Revise sidebar shown in Figure 3, the audio player was not functional at all. As shown in Figure 10, the audio player for audio feedback contained a circular black button with white ellipses next to each audio feedback note and a black play button that would strangely be located over the top edge of the sidebar. The button with ellipses would reveal a progress bar that was unable to play the audio feedback and the black play button seemingly did nothing.

In terms of UI, I fixed this issue by replacing these two buttons with a standard HTML audio player that gets shown immediately when a user opens the Author Revise sidebar. In terms of functionality, the reason that the audio player was unable to play any audio feedback was because the player was being loaded using a URL meant for opening an audio file in Google Drive. So the solution was to instead use the URL meant for downloading Google Drive content.
Figure 10
Classroom Rubric Attachment and Non-DocASSIST Assignments

After initially implementing the automatic rubric attachment for Classroom submissions, some users were experiencing a bug where the Review sidebar would be unable to open. This problem occurred specifically for teachers who would try to open the Review sidebar on a Classroom student submission for an assignment that had not been created using DocASSIST’s rubric manager. Thus, the error was occurring because DocASSIST could not find the necessary user properties for automatic rubric attachment and those properties are only set when the teacher creates a Classroom assignment with the rubric manager. The initial implementation neglected to account for this, so now, DocASSIST will ensure that those properties exist before attempting to load them into the document.

Unexpected Behavior with Update Summary and Table of Contents

When a teacher clicks the Update Summary button in the Review sidebar, DocASSIST updates the rubric summary at the bottom of the document by first deleting the previous summary and adding the updated summary to the document. This is accomplished by searching for the location of the string “Summary of “ in the document and deleting everything between it and the beginning of the next table that was created by DocASSIST (possibly a grade table or a drafts table) or the end of the document.

This was especially a problem because the “Summary of “ was being added to the document as a heading meaning that it would appear in a Google Docs table of contents. If a teacher tries to update the summary at that point, DocASSIST will notice the “Summary of “ in the table of contents and essentially delete the entire document. To prevent any issues with a table of contents, DocASSIST now uses regular bolded text instead of headings. Despite this change, there is also a chance that “Summary of “ could appear in the document itself. Although not a perfect solution, this is now avoided by having DocASSIST add and search for diamond ASCII characters (♦♦♦) at the beginning of that string (shown in Figure 12).
User Feedback

Throughout the development of DocASSIST, the only way that I was aware of user feedback was the option for users to email the DocASSIST team directly if they ever encountered an issue. Although this type of feedback was useful for catching bugs that I would not have noticed otherwise, it did not say much about the user’s opinions regarding the strengths of DocASSIST. Therefore, after implementing the last update to DocASSIST that I had planned, I reached out to 30 users who had recent DocASSIST activity between the months of March and April 2018 asking for their opinions about DocASSIST’s current state.

Most users I reached out to did not provide feedback, but the few that chose to respond gave similar in regards to what DocASSIST does well and where it can improve. These users often remarked that DocASSIST did succeed in saving time for them when grading and was also useful for peer review. Judging by these remarks, DocASSIST does succeed in its goal to an extent, but a common complaint was that it took a long time for the users to figure out how to use all of DocASSIST’s features. DocASSIST does have a website explaining all the features but this feedback suggests that help tutorials for using DocASSIST are not well advertised by the add-on itself. The other important piece of feedback among them was the difficulty of associating highlighted text with its corresponding feedback. The users found it inconvenient to have to scroll to the bottom of the document whenever they or a student wanted to check what feedback note goes with which section of text.

Future Development

Student and Teacher Separation

Currently, DocASSIST has features meant for use by both teachers and students and these features are separated into teacher and student menus in DocASSIST’s add-on menu, shown below in Figure 11. The problem with this setup is that students will also have access to the teacher features if they were to add DocASSIST to their Google Docs. This means that a student would be able to add or remove feedback to their own submission, possibly undermining the feedback made by their teacher. The most immediate solution would be to restrict access to the teacher functions if the user
is not a teacher. Theoretically, this is perfectly doable by utilizing the Classroom API to verify whether the user is a teacher when the document is opened.

![Add-ons Help All changes saved in Drive](image)

**Figure 11**

**Review and Grading as Tabs**

DocASSIST allows teachers review and grade assignments using two separate sidebars. Teachers will often utilize both of these sidebars in one sitting, so each of these sidebars has either a “Review” or “Grade” button that will switch the user to the other sidebar. But everytime a teacher wants to switch between sidebars, they need to scroll down to that button, click it, and then wait for the other sidebar to load. Therefore, switching between Review and Grade takes longer than it necessarily should. A potential alternative is to have both Review and Grade loaded into the same sidebar where the teacher can switch between them via tabs so that they do not have to wait for a new sidebar to load every time they want to switch.
Display of Feedback in Document and Author Revise

DocASSIST currently displays feedback in the rubric summary and in the Author Revise sidebar by having the feedback notes grouped by their corresponding rubric category (seen in a rubric summary in Figure 2). Within each category, the feedback notes are ordered chronologically according to the time that the teacher added from least to most recent, but when looking at each feedback note from the top of the summary to the bottom, they are not ordered chronologically. This means that a student would not be able to easily visualize which feedback note corresponds to which section of their document. Although each feedback note contains a link to a bookmark marking the relevant section, it is still tedious for a student since they have to keep scrolling down to the bottom of their document. The Author Revise sidebar helps a lot in this regard but that is also not ideal because only students who have downloaded DocASSIST can use that sidebar. Therefore, a way of easily displaying feedback and what sections of a document it relates to without requiring a user to have DocASSIST installed would be a significant improvement to aid students of teachers who use DocASSIST.
Time Spent Loading Audio Feedback in Author Revise

The Author Revise sidebar loads every feedback note added to the document so far all at once, so naturally, the time taken to load it will increase as the number of feedback notes increase. However, the graph below in Figure 12 plots the number of feedback notes in a document (from 0 to 100) against the time taken to open the Author Revise sidebar in seconds (from 0 to 50 seconds) for two cases: one case where the document only contains text feedback and another case where the document only contains audio feedback. With text-only feedback, the sidebar never took more than about 15 seconds to load, even with 100 feedback notes. With only 40 feedback notes, audio-only feedback took about 20 seconds to load, going all the way to almost 50 seconds at 100 feedback notes. For large documents with many audio feedback notes, waiting almost a minute to load Author Revise is inconvenient. A possible solution to this could be to have the audio files loaded dynamically as the user requests them instead of preloading them all at once when the user initially opens the sidebar.

Figure 12
Color Coding for Positive and Negative Feedback

When a teacher adds a feedback note by highlighting a section of text in a document, DocASSIST automatically colors the background of that text according to the color coding of the rubric categories. Although this color coding provides an easy way for students looking at their teacher’s feedback to differentiate which sections of their assignment have to do with which criteria of the rubric, this system does not allow students to visually differentiate between positive and negative feedback. For students going through their feedback, this makes it more difficult for them to focus on the negative feedback in order to improve their work. A different approach in the future could be to not limit color coding to only rubric categories or perhaps using lighter and darker shades of each category’s color to represent positive and negative feedback.

Extending Classroom Rubric Attachment

Automatic rubric attachment for Classroom submissions saves teachers a lot of time since they will not have to keep attaching the same rubric to each submission but this feature still has one shortcoming. The automatic attachment relies on the teacher attaching a rubric before grading any submissions and creating a Classroom assignment with DocASSIST and not manually using Google Classroom which means any teacher that does not follow these steps will still experience the same inconvenience that existed before automatic attachment. For additional convenience, automatic attachment should also occur after the first time a teacher attaches a rubric to a submission. Then, they will not have to repeat the process for all other submissions belonging to the same assignment.

Additional Logging in Database

DocASSIST’s backend primarily focuses on storing rubric information and not much else. One aspect in which the backend is lacking is logging user activity. Although DocASSIST already has Google Analytics enabled, dedicated tables for logging in the database make it simple to keep track of what specific features are used the most or least or how successful new updates are with users, among other things. Currently, the backend only has logging in the form of entities called Audit Logs. These logs keep track of whenever a user either creates, edits, shares, or deletes a rubric along with the email address of the user that performed that action. Ideally, the logging should be
extended to every possible user action to give the most information about how teachers are using DocASSIST.
Conclusion

By the end of my time continuing the development of DocASSIST, I was satisfied not only with what I had accomplished in terms of improving DocASSIST but also the information that I had acquired with the help of my advisor regarding how to improve the add-on in future updates. Even though DocASSIST still has issues to be resolved, I was able to improve the Review and Author Revise sidebars to help teachers save more time with the review and grading process and also help students better analyze the feedback notes added to the document by DocASSIST. I was also able to fix several bugs that were significantly detrimental to some important features and further utilize Google Classroom to aid teachers who regularly use Classroom. Finally, I am confident that the outstanding issues discussed in Future Development will provide a solid basis for the start of DocASSIST’s continued development during the 2018-2019 school year.
Appendix

DocASSIST Website

https://sites.google.com/view/feedbackdocassist/

Front-End Repository

https://github.com/neiltheffernaniii/docASSISTAppsScriptProject

Back-End Repository

https://github.com/neiltheffernaniii/docASSIST