Code Dialog: An Interactive Programming Feedback System

Madeline Burton
mburton40@gatech.edu

Abstract—Two-way feedback is a pedagogical best practice in primary and secondary classrooms. While there exist platforms for generic feedback in the classroom, and platforms for sharing code between developers, the intersection of these two domains is empty. This paper introduces Code Dialog, a prototype web application designed to fill this void. It is specifically tailored towards intuitive two-way feedback in introductory computer science classrooms.

1 INTRODUCTION

Over the past decade, the role of a teacher in the classroom has transformed from standing and delivering knowledge to pupils to mentoring and creating opportunities for learners to use their existing knowledge to make predictions and connections to better understand new topics (Dochy and McDowell, 1997). Formative feedback has become an essential tool to educate, evaluate, and embolden student learners. Formative feedback – feedback given throughout the learning process to help guide the learner's development – serves to educate the learner by engaging them in their zone of proximal development with respectful inquiry that encourages deeper thought about a particular subject (Mirzaee and Hasrati, 2014). It is often evaluative, giving a learner a clear understanding of their current place in the development process and sharing specific steps to move forward (Cauley and McMillan, 2010). It is also an essential tool for supporting students emotional engagement helping them build self-efficacy through thoughtful, specific, or motivational feedback (Tan et al., 2019; Cauley and McMillan, 2010).

The process of giving feedback has transitioned from paper and ink to digital means. Word processing software and screen annotation enables instructors to give more frequent, specific feedback that is both convenient and easy to interact with. Learners respond to this feedback more proactively, choosing more frequently to revise their work or engage in dialogue with their teacher to discuss further meaning (Chong, 2019). Not only does this two-way feedback increase success in learning outcomes, it “elicits perceptions and judgements, and assists learners to define future actions... ensure understanding, and [supports students to] become independent learners” (Tan et al., 2019). The increase of e-feedback is thus facilitating active learner engagement and the role of the modern day instructor.

This process is not unique to education, and is frequently used to support the development of software through code reviews. Software engineers give feedback on each other’s code in order to make the product more efficient, reduce confusion in the code base, and gain insight into the work of others (Anderegg, 2020). In both contexts, the value of feedback is evident and whenever possible should be pursued to further one’s cognitive and metacognitive growth.

Deep student learning and development is best fostered by ongoing dialogue about learner created artifacts. When formative feedback is designed to support active learning, metacognitive processes can promote deeper and more efficient learning (Price et al., 2010). While it is possible for an instructor to write directed formative feedback with this purpose, the learner’s development is enhanced when the feedback is treated as an opportunity for dialogue in which the instructor can leverage techniques of respectful inquiry to strengthen this process even more (Tan et al., 2019). There exist user-friendly tools to provide ongoing dialogue on learners’ textual works, and industry tools to provide ongoing dialogue about source-controlled code, however no platform exists to facilitate this feedback process in an programming context without the overhead cognitive load of mastering a source-control system.

There is hole that exists in education technology in the context of computer programming; while tools exist to give one-way feedback at the end of an assignment or to give automated feedback based on predetermined test cases or programming best practices, there are no such tools to encourage dialogue and the deeper level thinking and iteration that it produces. Without such a tool, the learning process moves more slowly, CS courses may have higher attrition, and learners are not prepared with the higher level thinking skills necessary to solve open-ended problems. In the absence of a digital tool to support this dialogue, feedback can still occur. In this instance however, it would only exist in the classroom through oral discussion and thus be limited to the number of learners an instructor could interact with during a class period. Given the prominence of technology in the classroom, it is only logical for this process to be available digitally, to allow dialogue to occur outside classroom hours and support a greater number of students in a timely manner.

2 RELATED WORK

In order for successful two-way feedback to occur, there must exist a tool to facilitate this communication. While there are many tools that facilitate communication between students and teachers, a tool that specifically supports discussion surrounding computer code is lacking from the introductory computer programming classroom.

2.1 Educational Tools

As of 2017, Google Apps for Education (now G Suite) was used by over 70 million students and teachers to support the creation of academic artifacts and dialogue about these products (Fenton, 2017). Google Docs, a product in this suite, is a tool that provides many benefits over traditional paper and ink feedback. Its simple user interface is designed with collaboration in mind. In a study of 65 learners newly introduced to Google Drive, 93.84% felt the tool was easy or very easy to use (Kakoulli Constantinou, 2019). One of the key features of Google Docs that can be leveraged for dialogue is the ability to have threaded conversations about a particular portion of a student's written artifact. Task-specific comments such as, "can you think of another example to support this statement?" tied directly to a portion of the work encourage students to think more deeply about their work and are more likely to contribute to extrinsic motivation than either grades or praise (Cauley and McMillan, 2010). Google Doc comments can be instigated by both the instructor and the learner, creating a simple opening for dialogue.

Many Learning Management Systems (LMSs) include the ability to interact directly with a student by leaving comments on the submitted student work. One such system is Canvas's DocViewer embedded in Speed Grader, which allows teachers to view supported file types and leave annotated comments on top of the student work (Titmus, 2020). Additionally, instructors can leave point annotations – comments relevant to a particular portion of the project – that can be accessed as threaded discussions in a similar fashion to Google Docs. Unfortunately, the user interface for many LMSs is not particularly intuitive. Student frequently do not know that comments exist on an assignment, negating any possibility of dialogue. (Stuhlsatz, 2020; Colburn, 2020).

2.2 Industry Tools

GitHub, Crucible, and BitBucket are all similar platforms created for the specific purpose of hosting developers' code, which excel at managing complex code bases. They do this through a suite of features including version control, change tracking, issue tracking, and collaborative development. When leveraged in education, the issue tracking features of GitHub have been used in a number of studies for peer review and general learner feedback (Feliciano et al., 2016). This feature allows reviewers to tag a specific section of code and leave comments about this topic in a threaded format. When participating in these studies however, students note that there is a significant learning curve one must overcome in order to engage with content through GitHub (Zagalsky et al., 2015; Feliciano et al., 2016; Larsén and Glassey, 2019). These same students shared that if they ran in to issues with GitHub they could more easily solve them by downloading the repository from scratch than trying to resolve the problems in the system. This could be one reason that Glassy (2006) found version control platforms do not encourage students to iterate on designs; in fact students are more likely to procrastinate working on the assignments. These limitations are not unique to GitHub, but applies more broadly to industry source-control platforms which are generally over-engineered for educational settings (Anderegg, 2020). The steep learning curve dramatically reduces these tools' effectiveness in an introductory classroom.

2.3 Synthesis

For classes that require general writing assignments, there are many tools available; Google Docs, Microsoft Word 360, and speed-graders built into LMSs are designed to allow instructors to comment, annotate, and make in-line suggestions to student work. They have also been strategically designed to simplify student engagement with instructor feedback. In programming courses, students must submit code with specific formatting and highlighting, code which could be processed by a computer. All of these requirements make the aforementioned tools undesirable solutions.

There are other tools such as GitHub, Crucible, or BitBucket used in the Software Engineering industry to support management of and collaboration surrounding programming projects. While these tools facilitate management of large programming projects, they are not built with the goals of education in mind.

3 SOLUTION AND DESIGN

3.1 Overview

Code Dialog is a prototype web application developed to support two-way dialog between learners and instructors or peer reviewers. Code Dialog allows users to upload program files, share files with a reviewer, and communicate feedback directly on the shared file. In cases where more than one version of a file has been uploaded, a reviewer is able to enable difference highlighting, which highlights the lines that have changed since the previous version and indicates what specifically those changes are.

3.2 Components and Information Flow

The app has been built in Python using the Django backend framework and a PostgreSQL relational database for data storage. The frontend has been built using the BantamJS Javascript framework. User accounts and security are managed through integration with Google authentication. The Heroku platform is used for hosting. Information flow through the app is defined in Figure 1.

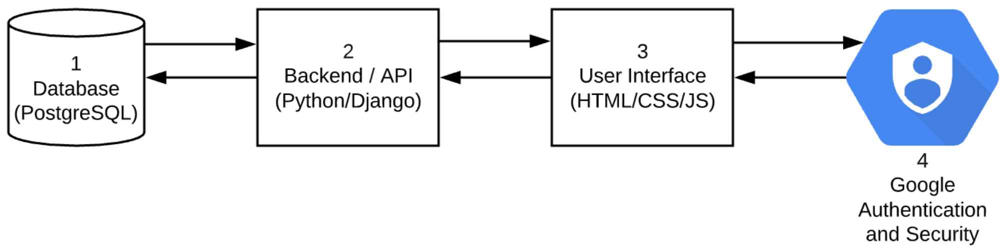


Figure 1—An information flow diagram with the following elements: (1) the project data source, a PostgreSQL relational database, (2) the backend and RESTful API endpoints, in Python using the Django web framework, (3) the interactive user interface, in HTML and CSS, using the BantamJS Javascript framework, and (4) the Google developer user authorization.

The database (1) stores users, document structure, comments, and distinct document versions (if applicable).

The backend (2) provides a model layer for interacting with the database, as well as a set of RESTful API endpoints whereby the user interface can pull information from the server and post changes back to the server.

The user interface (3) consists of three types of pages: the homepage, a user dashboard, and the file viewer visualized in Figures 3, 4 and 5 of the appendix respectively. On the homepage, users are able to view 'at a glance' the platform's features, and log on. On the dashboard, users are able to upload, access, and share files. From the file viewer users are able to download and interact in-line with a file.

Google Authentication 2.0 (4) is used to authenticate users and import a user's name and avatar.

3.3 Feature Overview

Key features of this prototype include the following items visible in Figure 5 of the appendix:

Syntax Highlighting—Based on the filename extension of the uploaded file, the platform will attempt to display the file with appropriate syntax highlighting. If a language is not specified or the extension is not recognized, the platform will attempt to auto-detect the language based on the file contents.

Line-Specific Commenting—Users are able to click on any line within the code and leave a message for the owner. Comments can be written in markdown to allow formatted text and code snippets to be included. All markdown will be rendered when saved.

Comment Threads—Users are able to respond to comments in a threaded format, facilitating two-way dialogue between author and reviewer.

Difference Highlighting—When a user uploads a second file with the same name as one already on their dashboard, this upload is considered a new version of the original file. From the file viewer, users are able to toggle between versions and compare versions using the "show differences" feature. Lines that have been removed appear highlighted in red in the older version of the file, lines that have been added appear highlighted in green in the newer version, and lines with changes are highlighted in yellow in both versions.

4 METHODOLOGY

To evaluate Code Dialog, both functionality testing and user testing were performed. Functionality testing of the platform included end-to-end testing of login, interactive features, syntax highlighting and difference tracking components. User testing included two phases. In the first phase, a small set of users were given a walk through of an early prototype, and then had the opportunity to interact with the tool. Interviews were then conducted with users to determine the features they valued, desired, or wanted to change. The interviews were conducted through open-ended questioning while watching the user navigate and interact with the tool. In the second phase, users were given no preview or support of the tool, and were asked to engage based on a short contextualization of the problem and solution. Their feedback was collected through a Google Form which included both Likert and open-ended questions.

5 RESULTS

5.1 Development

A high-fidelity prototype of Code Dialog has been deployed and is available at: Code Dialog: code-dialog.herokuapp.com

5.2 Initial User Testing

Although the survey sample space is small, the initial user testing resulted in a number of clear trends:

- Middle school students often have trouble remembering passwords and rely on teachers to remember or reset them over and over. Single sign-on with Google reduces the number of passwords a student needs to memorize, thus reducing stress for teachers and delay in the classroom.
- Middle and High school teachers are generally not familiar with Markdown and thus do not see its value. Once they are introduced to it, they become excited about the concept but also wary of the time needed to learn a new ’trick’.
- Teachers value difference highlighting very highly. This is again not a tool that any of the sample group had seen before, but they expected the speed with which they could look at iterations on a file would significantly reduce the amount of time taken to give feedback on an assignment.

5.3 Final User Feedback

Users were able to engage with a high-fidelity prototype of the Code Dialog platform that integrates intuitive two-way feedback, file versioning, and difference tracking. They participated in non-guided testing, completing a Google Form about the product's ease of use and their desired features. Overall users found the platform inviting and easy to use as visualized in Figure 3. 100% of participants felt that uploading files and writing threaded comments felt intuitive. This testing did, however, demonstrate that users were confused about how to add a second version of a file. One teacher shared, "Now that I know how to do it, it seems very simple. But I would not count on my students being able to figure it out on their own at home." This indicates a need for future work in order to simplify this process to reduce the learning curve of those using the platform.

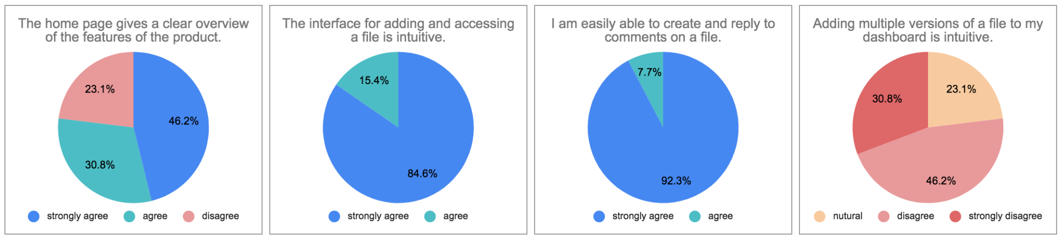


Figure 2—User feedback data regarding user friendliness of the platform.

6 LIMITATIONS

The largest limitation at this juncture is the limited user testing. This product has been developed in June and July and thus while a number of teachers have hypothesized about how they could use it in their classroom, none have actually done so. Code Dialog has not been user tested by students. It is probable that their experience with the tool will differ from that of their instructors and may highlight new use cases not addressed in the current version of Code Dialog. As indicated in the user testing, the versioning feature is not as intuitive as initially desired.

There are also some bugs with regards to difference highlighting. As of this writing, Code Dialog relies upon the Python library `difflib` to identify changes between files. Users have encountered cases where `difflib` incorrectly categorizes lines as "changed" rather than "added" which in turn affects the difference display within the app.

Furthermore there is no way for users to delete a document once it has been added to the platform, and neither edit nor delete functionality exists for comments. This can be problematic given that comments can be written using markdown and users are unable to preview the rendered version of their comments before saving, so there is a higher likelihood of a typo that causes the comment to render incorrectly than in generic text.

While addressing these issues is not within the scope of the initial project proposal, they will be addressed prior to Code Dialog's deployment in the classroom in early September.

7 CONCLUSION

In the context of a computer programming classroom, two-way feedback is essential to both learning and retention. Existing platforms for feedback dialogues are either tailored to writing artifacts (like Google Docs) or are designed primarily for codebase management and introduce unnecessary cognitive load (like GitHub and other industry source-control tools).

Code Dialog has been designed to bridge this gap as an intuitive platform that encourages learners and instructors to engage in clear, specific, two-way feedback. It allows in-line commenting in Markdown to encourage writing code segments, file versioning to promote the value of iteration, and difference highlighting to support students and instructors as they track development over time.

As acknowledged in the limitations section, this platform has not yet been tested in a classroom environment, but teachers are optimistic about its use and look forward to sharing it with their students this upcoming year.

8 FUTURE WORK

While this project created a proof of concept prototype that allowed users to engage in two-way feedback about code artifacts, there are some substantial shortcomings that should be addressed before deploying to an actual classroom setting. First, the platform lacks the ability to delete files once they have been added to the platform, as well as the ability to delete or edit comments made on a file. Both these features would give users more control over their content and ability to organize their submissions.

Second, difference highlighting has room for improvement. While the underlying code of `difflib` is unlikely to change, it would be reasonable to write a post-processing layer to check for the specific failure modes encountered and to standardize the output to an appropriate format.

Finally, there are several issues that became apparent through user testing of the existing prototype. Most notably, the current method for adding a new version of a file was not intuitive. Users have asked for a button directly on the file view to allow them to upload a new version of that file. Many testers also mentioned that they would like the ability to hide comments to simplify viewing files with long comment threads. As things currently stand, users may have to scroll through a long comment thread in order to get to the next line of code.

Despite these shortcomings, several testers expressed interest in using this platform in their classrooms this upcoming 20-21 school year, and so there are plans to update and address these issues prior to the beginning of the school year in September.

9 REFERENCE LIST

[1] Anderegg, C. (2020). The value of code reviews in industry [Private Interview].

[2] Cauley, K. M., & McMillan, J. H. (2010). Formative assessment techniques to support student motivation and achievement. *The Clearing House: A Journal of Educational Strategies, Issues and Ideas*, *83*(1), 1–6. https://doi.org/https://doi.org/10.1080/00098650903267784

[3] Chong, S. W. (2019). College students' perception of e-feedback: A grounded theory perspective. *Assessment & Evaluation in Higher Education*, *44*(7), 1090–1105. https://doi.org/https://doi.org/10.1080/02602938.2019.1572067

[4] Colburn, B. (2020). Feedback empathy research [Private Interview].

[5] Dochy, F. J., & McDowell, L. (1997). Introduction: Assessment as a tool for learning. *Studies in educational evaluation*, *23*(4), 279–98. https://doi.org/https://doi.org/10.1016/S0191-491X(97)86211-6

[6] Feliciano, J., Storey, M.-A., & Zagalsky, A. (2016). Student Experiences Using GitHub in Software Engineering Courses: A Case Study. https://ieeexplore.ieee.org/abstract/document/7883328

[7] Fenton, W. (2017). Google Classroom Could Bridge a Gap in Online Learning. https://www.pcmag.com/opinions/google-classroom-could-bridge-a-gap-in-online-learning

[8] Glassy, L. (2006). Using version control to observe student software development processes. *Journal of Computing Sciences in Colleges*, *21*(3), 99–106. https://dl.acm.org/doi/10.5555/1089182.1089195

[9] Kakoulli Constantinou, E. (2019). *Revisiting the cloud: Reintegrating the g suite for education in english for specific purposes teaching*. ERIC. https://pdfs.semanticscholar.org/f7c7/c1f5c2f1a1e581977b4641f21add1c19dcba.pdf

[10] Larsén, S., & Glassey, R. (2019). RepoBee: Developing Tool Support for Courses using Git/GitHub, In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, Aberdeen, Scotland Uk, Association for Computing Machinery. https://doi.org/10.1145/3304221.3319784

[11] Mirzaee, A., & Hasrati, M. (2014). The role of written formative feedback in inducing non-formal learning among masters students. *Teaching in Higher Education*, *19*(5), 555–564. https://doi.org/10.1080/13562517.2014.880683

[12] Price, M., Handley, K., Millar, J., & O'Donovan, B. (2010). Feedback : All that effort, but what is the effect? *Assessment & Evaluation in Higher Education*, *35*(3), 277–289. https://doi.org/https://doi.org/10.1080/02602930903541007

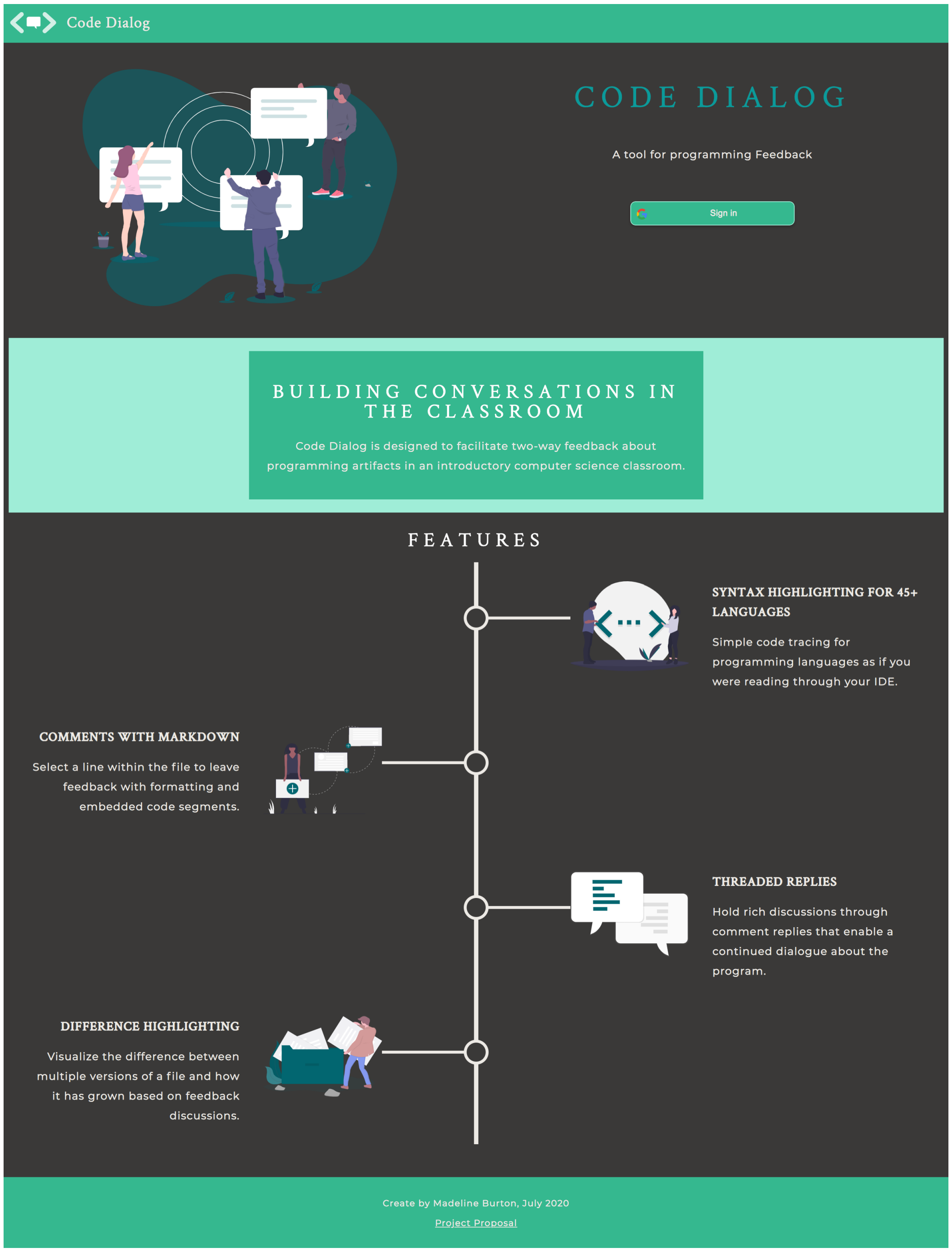
[13] Stuhlsatz, E. (2020). Feedback empathy research [Private Interview].

[14] Tan, F. D., Whipp, P. R., Gagné, M., & Van Quaquebeke, N. (2019). Students' perception of teachers' two-way feedback interactions that impact learning. *Social Psychology of Education*, *22*(1), 169–187. https://doi.org/https://doi.org/10.1007/s11218-018-9473-7

[15] Titmus, C. (2020). How do i add annotated comments in student submissions using docviewer in speedgrader? https://community.canvaslms.com/docs/DOC-26515-how-do-i-add-annotated-comments-in-student-submissions-using-docviewer-in-speedgrader

[16] Zagalsky, A., Feliciano, J., Storey, M.-A., Zhao, Y., & Wang, W. (2015). The Emergence of GitHub as a Collaborative Platform for Education, In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, Vancouver, BC, Canada, Association for Computing Machinery. https://doi.org/10.1145/2675133.2675284

10 APPENDIX A: APP USER INTERFACE DESIGN



Code Dialog

CODE DIALOG

A tool for programming Feedback

BUILDING CONVERSATIONS IN THE CLASSROOM

Code Dialog is designed to facilitate two-way feedback about programming artifacts in an introductory computer science classroom.

FEATURES

SYNTAX HIGHLIGHTING FOR 45+ LANGUAGES

Simple code tracing for programming languages as if you were reading through your IDE.

COMMENTS WITH MARKDOWN

Select a line within the file to leave feedback with formatting and embedded code segments.

THREADED REPLIES

Hold rich discussions through comment replies that enable a continued dialogue about the program.

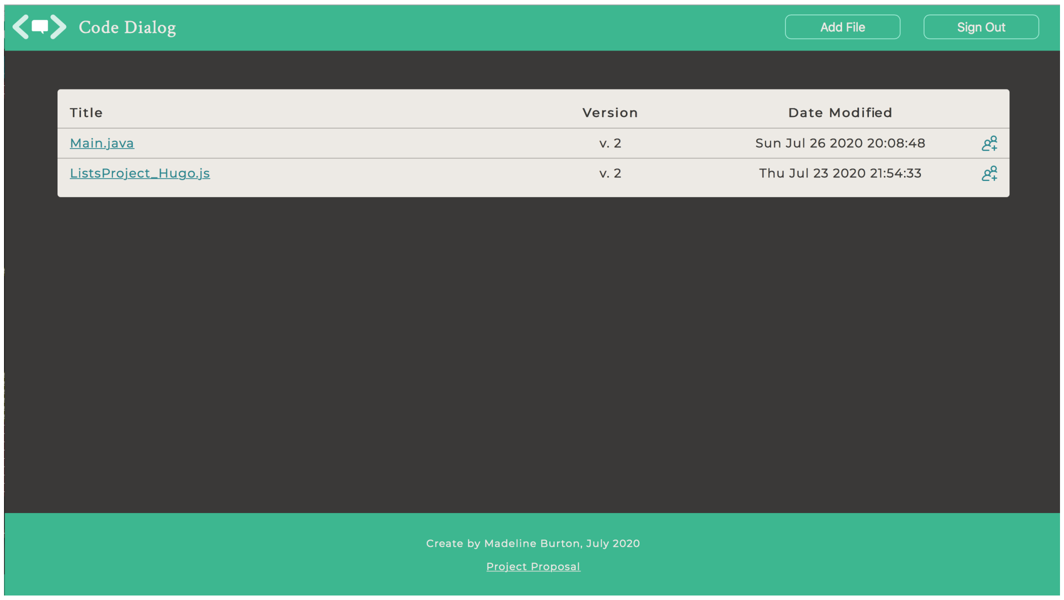
DIFFERENCE HIGHLIGHTING

Visualize the difference between multiple versions of a file and how it has grown based on feedback discussions.

Create by Madeline Burton, July 2020

Project Proposal

Figure 3—The homepage of Code Dialog

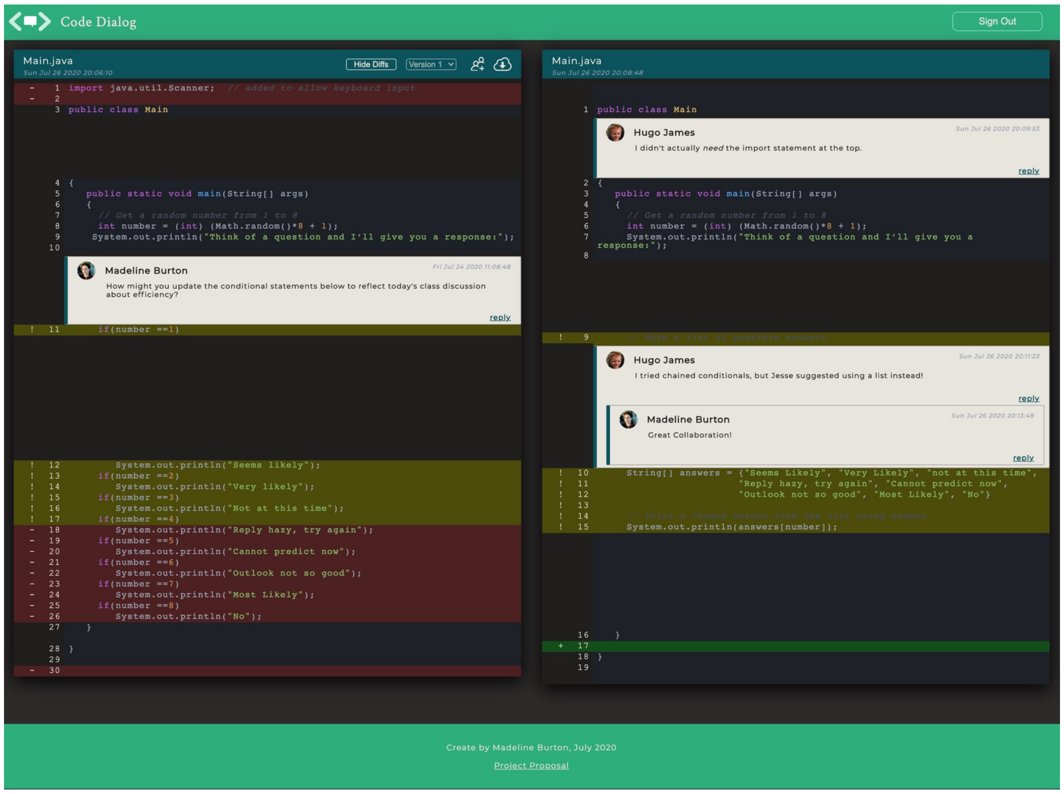


Title	Version	Date Modified
Main.java	v. 2	Sun Jul 26 2020 20:08:48
ListsProject_Hugo.js	v. 2	Thu Jul 23 2020 21:54:33

Create by Madeline Burton, July 2020

Project Proposal

Figure 4—A Sample user dashboard



Main.java
Sun Jul 26 2020 20:06:10

```
- import java.util.Scanner;  // added to allow keyboard input
- 
  public class Main
  {
     public static void main(String[] args)
     {
       // Get a random number from 1 to 8
       int number = (int) (Math.random()*8 + 1);
      System.out.println("Think of a question and I'll give you a response:");

```

Madeline Burton — Fri Jul 24 2020 11:08:48
How might you update the conditional statements below to reflect today's class discussion about efficiency?
reply

```
!      if(number ==1)
!         System.out.println("Seems likely");
!      if(number ==2)
!         System.out.println("Very likely");
!      if(number ==3)
!         System.out.println("Not at this time");
!      if(number ==4)
-         System.out.println("Reply hazy, try again");
-      if(number ==5)
-         System.out.println("Cannot predict now");
-      if(number ==6)
-         System.out.println("Outlook not so good");
-      if(number ==7)
-         System.out.println("Most Likely");
-      if(number ==8)
-         System.out.println("No");
     }

  }

- 
```

Main.java
Sun Jul 26 2020 20:08:48

```
  public class Main
```

Hugo James — Sun Jul 26 2020 20:09:53
I didn't actually *need* the import statement at the top.
reply

```
  {
     public static void main(String[] args)
     {
       // Get a random number from 1 to 8
       int number = (int) (Math.random()*8 + 1);
       System.out.println("Think of a question and I'll give you a
  response:");

!      // Make a list of possible answers
```

Hugo James — Sun Jul 26 2020 20:11:23
I tried chained conditionals, but Jesse suggested using a list instead!
reply

Madeline Burton — Sun Jul 26 2020 20:13:48
Great Collaboration!
reply

```
!      String[] answers = {"Seems Likely", "Very Likely", "not at this time",
!                          "Reply hazy, try again", "Cannot predict now",
!                          "Outlook not so good", "Most Likely", "No"}
!
!      // Print a random answer from the list using number
!      System.out.println(answers[number]);
     }
+ 
  }

```

Create by Madeline Burton, July 2020

Project Proposal

Figure 5—A sample interacting between instructor and learner.