Tracking Large Class Projects in Real-Time Using Fine-Grained Source Control

Gustavo Rodriguez-Rivera
grr@cs.purdue.edu
Purdue University
West Lafayette, IN, USA

Jeff Turkstra
turkstra@cs.purdue.edu
Purdue University
West Lafayette, IN, USA

Jordan Buckmaster
buckmast@purdue.edu
Purdue University
West Lafayette, IN, USA

Killian LeClainche
kleclain@purdue.edu
Purdue University
West Lafayette, IN, USA

Shawn Montgomery
montgo38@purdue.edu
Purdue University
West Lafayette, IN, USA

William Reed
reed226@purdue.edu
Purdue University
West Lafayette, IN, USA

Ryan Sullivan
sulli196@purdue.edu
Purdue University
West Lafayette, IN, USA

Jarett Lee
lee2363@purdue.edu
Purdue University
West Lafayette, IN, USA

ABSTRACT

Managing a class and identifying common problems becomes significantly more challenging as class sizes increase. Additionally, the increase of online learning requires better methods to track student progress remotely. In this paper, we describe a system that tracks student progress in real time. We propose a method for obtaining a fine-grained commit history by creating a Git repository for each student and automatically running commit/push commands every time a student compiles code. This approach makes a rich source of trace data that can track student progress in real-time, identify common problems students are having, alert faculty of students that are falling behind, and verify project authorship. However, analyzing individual repositories in a large class of students can be tedious and complex, so we have developed a system that provides quick access to all student repositories and summary information and statistics for their projects. This paper describes our approach for obtaining fine-grained source control commits in real-time, a method for tracking student and overall class progress using this data, and our experiences using this system.

CCS CONCEPTS

• **Social and professional topics → Student assessment**.

KEYWORDS

assessment, classroom management, source control

ACM Reference Format:
Gustavo Rodriguez-Rivera, Jeff Turkstra, Jordan Buckmaster, Killian LeClainche, Shawn Montgomery, William Reed, Ryan Sullivan, and Jarett Lee. 2022. Tracking Large Class Projects in Real-Time Using Fine-Grained Source Control. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022), March 3–5, 2022, Providence, RI, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3478431.3499389

1 INTRODUCTION

One of the biggest challenges of managing a large class is that small problems are exacerbated by the number of students in the class. For instance, challenging deadlines may cause students to submit incomplete work, or fail to submit anything at all. However, these same outcomes may be due to severe student procrastination. These problems are far more challenging to diagnose in large classes, and may result in massive frustration and anxiety for everyone.

In most projects, the only data instructors receive occurs after the project has been turned in and graded, when it is too late to apply any corrective measures. Instead of waiting until the project is turned in, it would be beneficial to monitor and detect problems in real-time while the project is being developed. If students find the project difficult to complete, it is important to recognize that as soon as possible so that instructors can organize extra help, extend the deadline, or simplify the project. If a student is falling behind due to procrastination, discovering this fact well before the project is due would allow instructors to intervene, perhaps by sending the student a reminder, or offering additional help.

We have called our system *EnCourse*. EnCourse keeps track of projects in real-time by using source control, specifically Git. In our system we create a Git repository for each student on a central server. The students clone this repository into their home directories. The initial sources include a Makefile, or project file, with the commands needed to build the executable of the program.

Normally a programmer uses Git to manually commit and push changes for a group of files after a task has been completed and tested. This task-based commit history is useful for development teams, but does not produce commits often enough to track the small changes that students make as part of their development/debug cycle.

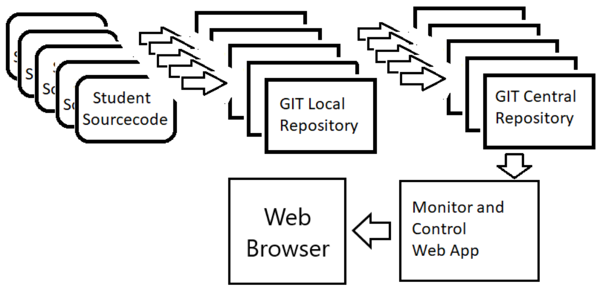

Figure 1: Student local and central repositories with EnCourse. EnCourse gets information from the student repositories in real-time, and can be accessed from a browser.

In our system, the Makefile or Project file that compiles the project contains Git commit and push commands to automatically commit changes into the student repository. Using this system, changes are tracked every time the project is compiled. When a student modifies a source file as a part of the program-build-test-debug cycle, the Makefile commits and pushes the recent changes into source control. This creates a fine-grained sequence of commits that tell the story of how the program was developed.

The students repositories are accessed through the EnCourse web application. The application periodically retrieves a summary of all student repositories to provide the instructor with an analysis of both the class's progress as a whole and that of individual students. EnCourse also provides real-time access to each individual student repository as shown in Figure 1.

2 RELATED WORK

There are currently free and commercial class management systems that simplify the submission and grading of programming assignments. Systems such as OK, Web-CAT, Autolab, and Coursemology, have been used with success in large classes [4, 5, 8]. These autograders are used to submit programs, grade projects, and provide feedback to the students. However, it is not possible to know the status of a student's project until the project is submitted. There are also commercial class management systems such as Vocareum, Codepost, Codio, and Gradescope, that are being successfully used in large classes, and are popular among small and large universities. These systems are primarily for submitting, grading, and giving feedback to students[2, 3, 7, 15]. Vocareum allows you to submit and view code, engage in automated grading, obtain basic grade analytics, and engage in in-browser development of programs. It lacks any fine grained information regarding student work progress. Codepost and Codio allow annotation and grading of student code. They also provide management mechanisms for teams of graders. There is no mechanism for gaining insight into realtime student progress. Gradescope is primarily geared toward written works. It provides detailed statistics after grading is complete. It is also submission based and lacks any mechanism for tracking progress. The MOSS (Measure of Software Similarity) system [14] is one of the top plagiarism detection systems out there, and it is widely used. However, it is also not real-time, providing information only after submission of completed work.

Source Control systems have been used previously to coordinate development of class material and projects or as part of grading [1, 9, 10, 12, 13]. The Pensieve system described in [16] is mainly designed to give instructors feedback on the student's learning process after the project has been submitted. It does provide the instructor with a history of how the project was developed. It also uses Git with fine-grained source control commits. However, the Git repositories used by the students are local to each student's directory and are unknown to the instructor until the project is submitted. It is only after submission that the instructor can see how the student developed the project.

A popular tool that provides source control in the cloud for education is GitHub Classroom [6]. This site allows repositories for an entire class and provides tools for commits, submission, and monitoring. This system makes it easy to distribute and submit homework and projects and shows statistics such as the frequency of commits per student. However, since the students have to commit their changes manually, it does not provide the fine granularity in the submissions necessary to provide feedback in real-time.

The EnCourse system, in contrast, provides real-time information of the student project by not only committing, but also pushing the commits to the student's central repository. This allows the instructor to remotely monitor progress in real-time, before the project is submitted. EnCourse is also able to send alerts to students who are falling behind, measure class progress, and give student feedback even before the project is submitted. EnCourse provides a real-time analysis of both the class as a whole and each individual student.

3 TRACKING STUDENT PROGRESS WITH FINE-GRAINED COMMITS IN REAL-TIME

When creating a project, we make a git repository for every student in a central server, using a remote file system with File Access Control Lists or FACL [11]. The permissions of each repository allow Read and Write operations for the individual student and the instructors. The repositories can be initialized with skeleton files if desired. Then, each student must clone their repository, so the student can commit/push their changes. EnCourse detects these changes and updates its statistics and history. The git commands are added directly to the Makefile that the student uses to compile the project. In this way, we can obtain a complete history that mainly contains commits focused on a single change. This can be achieved by adding a git target to the Makefile which uses shell commands for git to add relevant files to staging, commit the changes to the local repository, and then push the changes to the remote repository.

Since compilation and testing time can be large in general, EnCourse does not compile or test projects that are being monitored. Instead, instructors using EnCourse provide students with a test suite that allows them to determine a partial project grade. As part of the build process, this test suite is run and its output is captured

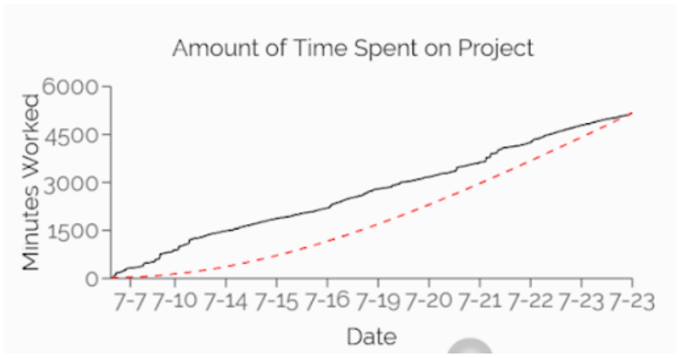

Figure 2: Estimation of time spent on a project by a Student (solid line) and the class average (dotted line) displayed by EnCourse.

and stored in the repository. EnCourse then uses this information to determine the student's progress.

Given the previously described environment, students should be instructed to follow a good workflow in order to get fine grained commits that can nicely show their progress. We also remind students that they will be subject to extra scrutiny if their commit history does not support the expected effort. Weekly grading checkpoints also encourage students to work diligently on their assignment. Students should also be encouraged to build often and test whenever they think they should be passing another test case, so that way both commit history and progress over time have good granularity.

4 ENCOURSE FEATURES

Analyzing individual repositories in a large class of students can be tedious and difficult. EnCourse provides quick access to all student repositories as well as summary information and statistics for their projects. EnCourse is able to:

- Estimate the time spent by students on a project.
- Detect possible instances of academic dishonesty.
- Send alerts to students when they are falling behind in the project.
- Provide data to learn about how students program and debug.
- Access individual student repositories and see changes.
- Sort students by time spent, number of commits, tests passed, and progress.

EnCourse estimates the amount of time that each student spends on a project using fixed-size time intervals (E.g. 20 minutes). An interval starts only after the previous interval has ended and a new commit has been made. Multiple commits that happen within a time interval are ignored. Only until the time interval expires and another commit occurs will another time interval begin as shown in Figure 2.

EnCourse also evaluates student projects to catch instances of academic dishonesty. There are three main metrics used to evaluate this. First, the rate of progress relative to the elapsed time is used to estimate how long a student spent thinking about a problem. Second, the rate of progress relative to the number of commits is used to detect cases of copying bulk amounts of code from another source. Third, the ratio of added vs. deleted lines helps estimate the frequency of mistakes that a student makes while working on their project. Fewer mistakes often indicate a higher likelihood of academic dishonesty. By combining these metrics with a measure of code similarity, and configurable weights, we generate a score which represents an estimated, relative likelihood of academic dishonesty between students.

EnCourse is capable of sending alerts to instructors and teaching assistants when students are falling behind using progress metrics retrieved from Git. Additionally, EnCourse can reveal the portions of the project on which students spend the majority of their time by determining the time spent on each task. It can also be used as a research tool to learn how students program by examining the combined Git information and calculated metrics.

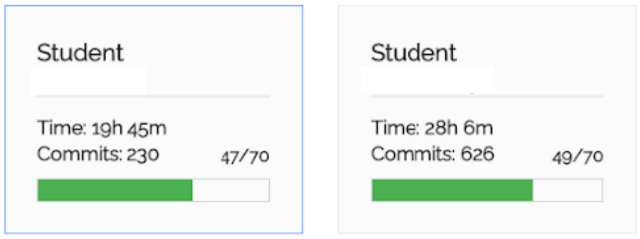

Figure 3: Summary of Student Repositories in EnCourse that can be sorted by number of hours spent in project, commits, name etc.

EnCourse has tools to provide a summary view of projects for the entire class as well as drill down to individual student statistics as shown in Figure 3. This is different from tools like GitK or Gitweb that allows browsing source code from only one repository at a time. EnCourse allows sorting the students projects by time spent, progress made, or number of commits. This helps identify both students who are falling behind on the project as well as students doing exceptionally well.

EnCourse also shows the number of commits over time (Figure 4). We would expect that a good student will have several commits spread throughout the project. No commits at the beginning of the project may be a sign of procrastination. Figure 5 shows the progress of the student as measured by the tests passed over time. We expect that the tests passed will be monotonically increasing. A large number of commits with little progress may show that the student is spending excessive time on a single test.

Another useful chart is the cumulative number of lines added and removed over time as shown in Figure 6 and Figure 7. A line change in Git is represented as removing an old line and adding a new one. We expect that a student will have both added and removed lines during the project development, giving approximately a symmetric triangular shape. In this figure, additions are represented in green, and lines removed are represented in red. A chart that has mostly lines added (green) and few or no lines removed (red) would potentially indicate a case of academic dishonesty.

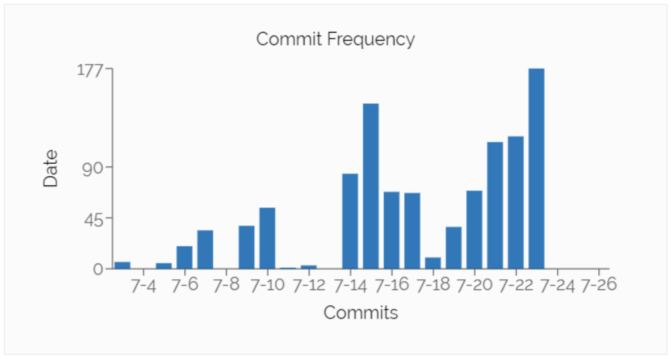

Figure 4: Commit Frequency per day of a student.

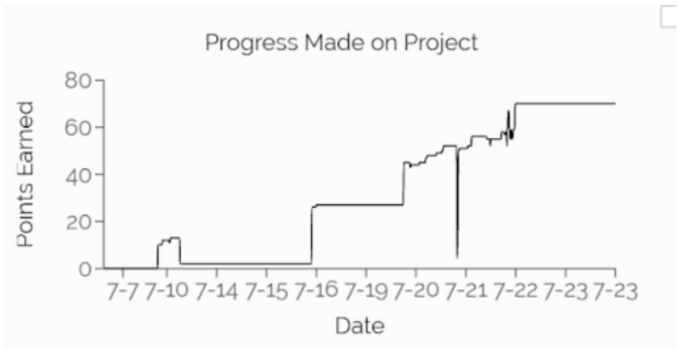

Figure 5: Progress of a student through time estimated using the test suite.

5 EXPERIENCE

We have used EnCourse in our Systems Programming course for five semesters: Fall 2018, Spring 2019, Spring 2020, Fall 2020, and Spring 2021 containing 228, 301, 311, 117, and 342 students respectively. EnCourse has proven to be a valuable resource for us, especially during the beginning of a project, by helping identify those students who are already falling behind, often due to procrastination. Knowing the amount of time that a student has spent on the project helps us reach out to students and offer assistance if needed.

When we initially used EnCourse in Fall 2018, in the first project, we detected that only one-third of the class had started after the first week, even though the students told us during the lecture that the project was going fine. So we sent an e-mail to the procrastinating students asking them to start immediately and seek help if needed.

In the middle of a project we use EnCourse to measure the class progress as a whole and to identify the parts of the project that students find the most difficult to implement. If a significant number of students are not close to finishing the project, we offer extra help hours. This has allowed us to extend deadlines for projects that turned out to be more difficult than anticipated. Such decisions can be backed up by data obtained from EnCourse.

In Spring of 2020, when lectures had to go online due to Covid-19, we had to reduce the scope of the final project. There was little time left, and the rate of progress in the final project was not the same as we expected in a regular face-to-face classroom. The statistics in EnCourse about student progress helped us make this determination.

We have also been able to identify cases of academic dishonesty by detecting when a repository shows large pieces of code added in one single commit or when a large number of tests pass after a very short period of time. This detection can be done remotely and without the student's knowledge. In these cases, we have asked the lab instructor to use other tools such as MOSS, interviews, and implementation quizzes to gather more information. Academic dishonesty is a very sensitive matter. When it occurs, the more data an instructor has, the less difficult it is to make a decision.

For example, Figure 6 shows the cumulative graph of lines of code added and deleted over time for an average student. During a regular development cycle, the student adds new lines as well as modifies existing ones to fix bugs. A line modification is registered in Git as one line deleted and one added. We expect that the portion of lines added (green) will be comparable to the number of lines deleted. (red).

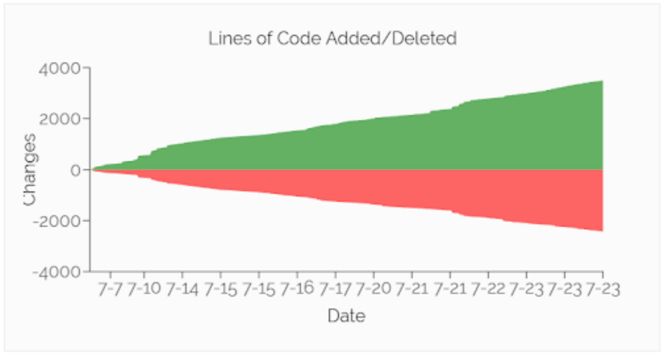

Figure 6: Cumulative graph of lines added and deleted over time of one student.

The graph in Figure 7 shows a potential case of academic dishonesty. This graph illustrates that the number of lines added is much larger than the lines deleted, implying large pieces of code were added at once without modification.

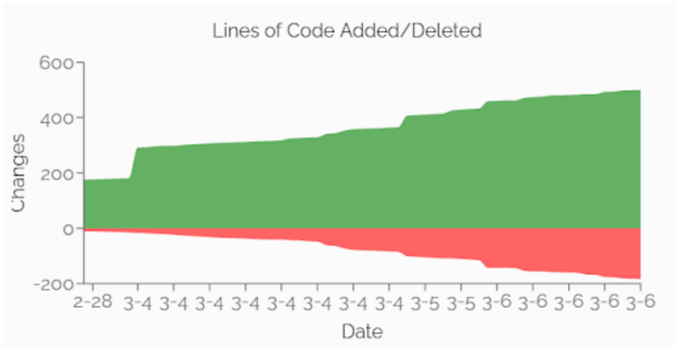

Figure 7: Cumulative graph of lines added and deleted over time of a potential academic dishonesty.

More evidence accumulates when one looks at the history of commits over time in Figure 8. This graph shows that the student worked only a few days and only built the project a few times. A commit in our system is equivalent to a write-build-test-debug cycle. For a project that has 500 lines of code, only 30 development cycles is highly suspicious.

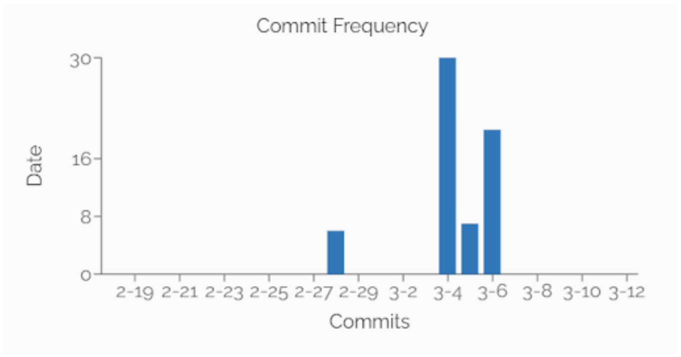

Figure 8: Commit Frequency of a possible case of academic dishonesty.

There have been false positives when some extremely good students are able to finish programming projects in short periods of time and few commits. In these cases, the lab instructor can verify through the implementation quiz that the student was the author of the project. An implementation quiz asks the student to show in the code where certain features are implemented and to explain how the code works. These are questions that the student should know if he/she is the author of the project. EnCourse gives valuable information in real-time about student progress. However, it is still up to the instructor to interpret the data and take action when necessary.

6 FUTURE WORK

We are working on making EnCourse easier to install and set up so others can use it. Since hosting a git server for source control may be a challenge for some institutions, we would like to allow GitHub classroom for hosting and have EnCourse communicate with GitHub Classroom. Since a commit/push operation into GitHub Classroom at every build cycle may take a long time, we would like to execute these operations in parallel while the student is working on the project.

In addition, we would like to improve the accuracy and utility of metrics to measure student progress. Finally, Since EnCourse provides a fine-grain record of how the student approaches programming, we see great potential in using EnCourse as a tool for Computer Science education research.

We are also exploring the possibility of integrating IDEs such as Visual Studio Code, Jupiter, PyCharm, or IntelliJ into EnCourse. EnCourse is an open source project and can be downloaded from https://www.cs.purdue.edu/homes/grr/Encourse. We invite you to use it, and we welcome your suggestions.

7 CONCLUSION

EnCourse is a new tool that proves to be very useful for instructors in computer science. It provides valuable insights both at the overall class level and for individual students. EnCourse can also help keep teaching assistants up to date on how students in their lab section are progressing. They can use this information when deciding how to organize lab presentations and which students to focus on during lab sections. EnCourse can be a valuable tool to track the progress of projects in large classes.

8 ACKNOWLEDGEMENTS

We thank the reviewers for the invaluable feedback and suggestions that improved this paper. We also thank the teaching assistants for their input while using EnCourse. Finally, we also thank the instructors of the senior project course at our institution that facilitated this project.

REFERENCES

[1] Curtis Clifton, Lisa C. Kaczmarczyk, and Michael Mrozek. 2007. Subverting the Fundamentals Sequence: Using Version Control to Enhance Course Management. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (Covington, Kentucky, USA) *(SIGCSE '07)*. Association for Computing Machinery, New York, NY, USA, 86–90. https://doi.org/10.1145/1227310.1227344
[2] Codepost. 2021. *Codepost.* Retrieved August 1, 2021 from https://codepost.io
[3] Codio. 2021. *Codio.* Retrieved August 1, 2021 from https://codio.com
[4] John DeNero, Sumukh Sridhara, Manuel Pérez-Quiñones, Aatish Nayak, and Ben Leong. 2017. Beyond Autograding: Advances in Student Feedback Platforms. *In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (2017), 651–652. https://doi.org/10.1145/3017680.3017686
[5] Stephen H. Edwards and Manuel A. Perez-Quinones. 2008. Web-CAT: Automatically Grading Programming Assignments. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education* (Madrid, Spain) *(ITiCSE '08)*. Association for Computing Machinery, New York, NY, USA, 328. https://doi.org/10.1145/1384271.1384371
[6] GitHub. 2021. *GitHub Classroom.* Retrieved August 1, 2021 from https://classroom.github.com
[7] Gradescope. 2021. *Gradescope.* Retrieved August 1, 2021 from https://www.gradescope.com
[8] Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. 2010. Review of Recent Systems for Automatic Assessment of Programming Assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '10)*. Association for Computing Machinery, New York, NY, USA, 86–93. https://doi.org/10.1145/1930464.1930480
[9] Oren Laadan, Jason Nieh, and Nicolas Viennot. 2010. Teaching Operating Systems Using Virtual Appliances and Distributed Version Control. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (Milwaukee, Wisconsin, USA) *(SIGCSE '10)*. Association for Computing Machinery, New York, NY, USA, 480–484. https://doi.org/10.1145/1734263.1734427
[10] Joseph Lawrance, Seikyung Jung, and Charles Wiseman. 2013. Git on the Cloud in the Classroom. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 639–644. https://doi.org/10.1145/2445196.2445386
[11] RedHat. 2021. *File Access control Lists in Linux.* Retrieved August 1, 2021 from https://www.redhat.com/sysadmin/linux-access-control-lists
[12] Karen L. Reid and Gregory V. Wilson. 2005. Learning by Doing: Introducing Version Control as a Way to Manage Student Assignments. *SIGCSE Bull.* 37, 1 (Feb. 2005), 272–276. https://doi.org/10.1145/1047124.1047441
[13] Karen L. Reid and Gregory V. Wilson. 2005. Learning by Doing: Introducing Version Control as a Way to Manage Student Assignments. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (St. Louis, Missouri, USA) *(SIGCSE '05)*. Association for Computing Machinery, New York, NY, USA, 272–276. https://doi.org/10.1145/1047344.1047441
[14] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. 2003. Winnowing: Local Algorithms for Document Fingerprinting. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* (San Diego, California) *(SIGMOD '03)*. Association for Computing Machinery, New York, NY, USA, 76–85. https://doi.org/10.1145/872757.872770
[15] Vocareum. 2021. *Vocareum.* Retrieved August 1, 2021 from https://www.vocareum.com/

[16] Lisa Yan, Annie Hu, and Chris Piech. 2019. Pensieve: Feedback on Coding Process for Novices. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE 2019, Minneapolis, MN, USA, February 27 - March 02, 2019*, Elizabeth K. Hawthorne, Manuel A. Pérez-Quiñones, Sarah Heckman, and Jian Zhang (Eds.). ACM, 253–259. https://doi.org/10.1145/3287324.3287483