Novel Grading Tool Based on Open Source Software

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Philip Hart

Spring, 2023

Technical Project Team Members

Ishan Mahur

Marcus Mann

Ketian Tu

Panagiotis Apostolellis, Department of Computer Science

Novel Grading Tool Based on Open Source Software

CS 4980 Capstone Report, 2023

Philip Hart
Department of Computer Science
University of Virginia
Charlottesville, VA, US
ph9aa@virginia.edu

Abstract—**Developing open source software is prominent among programmers around the world. Examining and exploring the technology stack of open source applications make for a more informed product manager and more skilled programmer. Further, practicing open source development offers exposure to new methods and diverse ideas. Throughout the development of a grading tool, several different open source software tools were engaged with, learned from, and used to modify components of the application to support the grading process for Project-Based Learning (PBL) assignments. Information regarding the processes of these open source software tools was consolidated and detailed.**

Keywords—open source, grading software, annotation technology, application development, technology stack

I. Introduction

Since the inception of the modern computer, the means of designating source code ownership has resulted in a diverse set of software. While businesses compete to deliver the best online platforms and reap the most rewards, other groups of programmers collaborate to produce code that anyone can use and distribute. Advocates of this ‘open source’ idea claim that it results in more robust software and more diverse business models [1]. The idea of open source software originates from Richard Stallman, who proposed it in 1984 with the launch of the Free Software Foundation (FSF) [2]. Raymond, an open source advocate, and a software developer wrote ‘The Cathedral and the Bazaar’, an essay that argues that when programmers work on source code, sending files to other programmers inevitably improves the code [3]. Open source software is not a new concept and has been applied in countless fields and has been proven to deliver faster and better results than proprietary technology. The open source idea is crucial to consider whether in business or as a programmer. Further, open source technology seeks to be compatible with other relevant technology, such as massive tech companies such as Apple, Microsoft, and IBM seeking to make their products compatible with powerful open source software [2].

Due to the importance of open source software, I chose to explore technology that has been built on its principles. In 2015, a non-profit named The Hypothesis Project released their annotation tool for the web named Hypothesis [4]. The tool reached 20 million annotations in 2021, proving to be a scalable software system. User interaction with the tool’s features includes highlighting text, annotating text, joining groups, and replying to other annotations in your groups.

In this paper, the Hypothesis tool’s web framework and technology stack will be detailed in section II, followed by requirements for repurposing the tool for grading in section III. Section IV details the design of the developed grading tool and Section V explores the limitations and potential features for the tool based on what has currently been developed. Lastly, section V concludes the technical report and argues for the benefits of this project.

II. Hypothesis Technology and Overview

Hypothesis is an open source software tool for annotating documents while surfing the web [5]. There are two user interfaces for the tool, a website server for account, group, and annotation management, and a Chrome extension for annotating web pages. The official Hypothesis build on the Chrome Web Store is set up in a specific way. After signing up for an account, the user joins the public group, a group where all Hypothesis users are part of. To avoid confusion, the developed grading tool does not support a public group. Users can set up their own groups, invite other users to their group through a link, and begin annotating. This is a high level overview of how a user interacts with the Hypothesis tool. However, the services and processes that allow the tool to operate are complicated and will now be explained in further detail. The technology described is only relevant to H, the source code that builds and runs the Hypothesis back-end and web portal.

A. *Application Development and Management*

The H application is built using Docker, a software that virtualizes an operating system in order to package source code, dependencies, and resources into a Docker Container. This method is very powerful, as containers package applications together, guaranteeing successful execution from one computing environment to another [6]. Further application management includes GNU make, just one utility of the vast quantity of open source software that the FSF has produced. The MakeFile contains various commands useful for code contributors, such as how to start the necessary services in Docker and how to run H. There are various other useful commands in this file that formats code, enters the server shell of H, and run developed tests.

Beyond Docker and the make utility, Hypothesis developers created custom CLI scripts extremely useful for future developers contributing and testing code. These scripts include CLI commands for reindexing Elasticsearch data, initializing the database and other services, and for populating the database with data for testing. These scripts proved to be useful when changing components of the source code.

The web framework used to build the H application is Pyramid, a Pylons Project product in Python. This web framework is highly compatible with many other open source software used in the application, such as SQLAlchemy and Jinja2. The former is compatible with PostgreSQL and the latter is an HTML friendly web templating language. Further, H benefits greatly from using other Pylons Project products such as Colander and Deform for forms, heavily used for the creation and editing of group and user information in the H portal.

B. Services

There are various services used by the H application, including Elasticsearch, PostgreSQL, and RabbitMQ.

Elasticsearch is an open source search and analytics engine capable of handling various types of data [7]. In H, Elasticsearch is where annotations are stored before the H application fetches some to display either in the Chrome extension or the H portal. Elasticsearch works by storing data, called shards, in nodes, which are a crucial component of scalability in cloud computing. Each node can replicate data from other nodes to their own if needed, however, an Elasticsearch index includes only the data in the nodes that are not replicated. An Elasticsearch cluster is defined by a group of nodes. The most relevant type of search for the H application is a text search. This is how annotations based on group are fetched for a web page. In Elasticsearch, text must be analyzed before it is stored in a node. The result of the analysis is stored in data structures that makes searching for a text attribute more efficient. Elasticsearch uses analyzers for this preprocessing step which may be customized, but the default analyzer removes punctuation and makes all letters lowercase. After the text is analyzed, it is indexed into a document, which is a JSON object, and mapped to the corresponding key. In the case of H, each document represents an annotation object, storing information such the creator, text, group id, creation date, deletion status, and more.

The application of Elasticsearch in H means that every annotation shown on a user's screen comes from an indexed annotation in the Elasticsearch service. In the back-end portal, annotations may be filtered by user, group, tags, and URLs. This filtering mechanism passes a term to the Elasticsearch engine which initiates a fetch to annotations that contain an equivalent term in a specified key of the annotation mapping. This results in a certain subset of annotations displayed on the user's screen. In the client, a user will have a group selected by default, whether it is the public Hypothesis group or any groups they have joined or created. Upon accessing a specific webpage, the client makes an API request to the H back-end to fetch annotations from the Elasticsearch index that contain a matching group id. This is an example of elasticity, and it contributes towards high performance since the service does not need to transmit all annotations to the H service, only the ones that will be displayed.

PostgreSQL, or Postgres, is an open source object-relational database system that uses the SQL language to perform complicated queries safely and securely [8]. A benefit of relational database systems is that there is much inbuilt fault tolerance. Thus, Postgres is an excellent choice for an application that must keep track of users in groups, annotations in groups, and much more. Further, the choice of database allows customization to further define how the application should run. Adding fields to the annotation table allows for the storage of more data that is incredibly useful for grading online reports. To change the database in H, Alembic, a python library for database migration, is used. First, write an Alembic script, import SQLAlchemy for Python to define columns in the table, and then use the Hypothesis CLI commands to initiate the migration.

RabbitMQ is a message broker used for handling requests to the H application. This is ideal for long running requests to H as it enables H to manage many at once, providing an extremely scalable application.

III. Grading tool Requirements

Design requirements for the grading tool were developed using the H code base by the lead researcher through his experience in grading web-based reports hosted on webpages such as Google Sites for a Human-Computer Interactions course (HCI). Further requirements were detailed from colleagues and teaching assistants in his course. The requirements were developed in the context of Project-Based Learning (PBL), where students are encouraged to tackle complex open-ended design problems. PBL is being increasingly applied in design-based engineering courses [9]. This form of assignment challenges students to prototype, test, and refine their solution to a problem. This may result in an online deliverable that includes all text and media detailing the progress on their project. Providing feedback at regular intervals over the course of a project is much more effective for a student's learning process during such a project.

A. *Within-context feedback and grading.* For feedback to be efficient for the grader and most useful to the student, its placement in student work should be where deductions are merited in the document.

B. *Personalized adjustments of score and feedback.* A fixed rubric, which has defined point deductions when student work does not meet a certain learning objective, does not adequately capture the quality of student work. Deductions should be personalized to the match student accomplishment.

C. *Collaborative grading.* Not all graders in a higher level education course have the same knowledge or skills as the professor. Thus, collaborative grading is the best way to learn the best strategies of giving feedback and to communicate with other graders for increased grading consistency.

D. *General feedback and regrade requests.* An overall summary of feedback for student work is helpful for the learning process. Should a component of the work be misunderstood, the student should be able to submit a regrade request to explain their rationale and earn points back.

Using the above requirements, the H code base, along with the client (Chrome extension) code base were repurposed for grading online deliverables in an undergraduate HCI course.

IV. Design of E^{2}Logos

The tool e^{2}logos, for Evaluating Electronic Logos (from the Greek word *λόγος* for word), was developed with the requirements detailed in section III and applied to a course to assist in the evaluation of online deliverables for PBL assignments.

The H code base implemented groups, an abstract idea that could be interpreted as project groups. Thus, users that are instructors or graders must have certain permissions to grade within these groups while students should not be able to view their grade until the assignment is released. Thus, user roles needed to be defined via an invite link for joining a group. The instructor of a course would create group, which defines them as the group creator in Hypothesis, but an instructor in e^{2}logos. The grader and student role have little permissions in the back-end portal since they should not be able to manage course materials. Changes to the H back end included the addition of assignments and courses. Thus, only relevant annotations from the selected assignment or course would appear in the dashboard. The implementation of these two components contributed to rapid progress in using Hypothesis for a grading and feedback tool.

First, custom rubrics could now be requested by the client from the Postgres database and displayed dynamically. The rubric is located in the Hypothesis sidebar which overlays the web page that is being graded. This is optimal for grading and reduces the overhead of switching tabs during grading. This fulfills requirement A, and allows graders to read, interact, and write in the connected system of the web page and the e^{2}logos Chrome extension.

Second, the database schema of annotations was altered to support grade annotations that contained either point deductions or bonus points. In the client, Graders are able to highlight text on a page, select a rubric item, and create an annotation with personalized feedback and point adjustment. The development of this feature fulfills requirement B, one of the most crucial requirements not completely supported by Hypothesis.

Third, while Hypothesis already supports live updates of new annotations, a feature was added to increase the communication between graders. Graders may make a regular annotation describing their thought process on how to grade a student's online deliverable. Currently, it is still possible to see this annotation in a group if it is not attached to a course assignment. Random annotations like these may be made at different points in a semester, so a solution was derived to implement a grader's only option for posting comments. This restricts the annotation to users in the group that have the role of a grader. This same concept applies to an annotation reply, which may make it easier to debate certain deduction and increase grading consistency. Thus, this feature fulfills requirement C.

Lastly, Hypothesis fully supports general feedback through the use of a regular annotation. However, requirement D is not fully fleshed out on the e^{2}logos system. Of course, enabling student replies to annotations is possible, but the best way of displaying those replies to an instructor or grader is difficult given the context of Hypothesis. The back end portal, or course dashboard, is likely the best place to aggregate student replies to grade annotations and allow instructors or graders to reject the appeal or adjust the feedback accordingly.

Regarding other back end updates, the bucketing system was altered to bucket by assignment then by group. So each assignment section has multiple group buckets with annotations relevant to the group and assignment. A feature was implemented with these buckets to display the score of an assignment on a group bucket header. This shows a quick calculation of the group's grade from the back end portal without having to navigate to the page.

V. Limitations and Future Work

While e^{2}logos offers a novel system for evaluating online student work, there are still many useful course management features that the tool lacks.

The definition of a user's role in e^{2}logos is dependent on a specific group. This causes a few issues. First, the back-end portal is not aware of a user's role when any non-group filter is applied. Upon login, the default filter is no filter, meaning all of the relevant annotations in a user's groups are displayed. Since all groups are accounted for, the system is unable to assign a user a specific role, since it may vary from semester to semester. Thus, elevated permissions, such as creating assignments, groups, and courses are restricted to admins and staff of e^{2}logos. Future iterations of this tool may benefit greatly from role definition upon account signup. However, the flexibility of roles from group to group does serve a good purpose in the context of higher education courses. An alternative and perhaps more powerful solution would be to implement a user management page, where the admin of the site (the instructor in this case) is able to set the role of each user that has joined a group in their course.

Compared to other grading tools, which display statistics such as the mean, median, and standard deviation of an assignment's grade, e^{2}logos does not offer in-built features for tracking and analyzing individual and overall student learning. A future goal is to implement these statistics for the assignment filter page, which displays annotations from multiple groups for one assignment. One aspect of e^{2}logos that may help track student learning objective is analyzing the rubric sections for deductions shared among multiple groups. This is a form of tagging, shown to help assist the learning process by informing the instructor of what concepts to focus on [10].

Another limitation is the overall design of the dashboard located at the H portal. For one, Hypothesis was designed to bucket annotations by URL, which is rationale because that is how the web is traversed. However, it is more logical for annotations to be grouped together by the group and assignment that they belong to. This enables the instructors to aggregate only annotations that are relevant to their grading review in their dashboard. Thus, the bucketing of annotations on the dashboard is currently limited since the same annotations from the same assignment and group may appear on two different pages. The ideal implementation would be to paginate per assignment if the number of annotations per assignment is large but would only populate one page if the number of annotations per assignment is small. Thus, I believe future logic must be designed to accommodate these two cases, as instructors who are users may have different course contexts. Another issue is grouping annotations by their document in a bucket. Although, this could be easily implemented by sorting the annotations based on document id and then designing the UI which would show separation between annotations within a single expanded bucket. An extremely helpful component in testing changes to the bucketing system is a mirror of the HCI course database. This mirror allows for testing of code with a massive number of copied annotations.

As mentioned in section IV, there is no regrade request management system in the back end portal, which make processing regrade requests for the instructor difficult. Future iterations of the tool would benefit the instructor and graders greatly by including a regrade request management system.

Within the back end, relevant annotations must be easily accessible, or else the usefulness of aggregated annotations is lost. Therefore, much UI improvement and data organization must go into customizing the back end portal. Hypothesis was built for a different purpose, and user engagement with the portal will suffer if it contains no features beneficially to the grading process and managing the course.

VI. Conclusion

Throughout the development of this project, I have been exposed to various open source software tools and learned more about how they interact with other applications on the internet. I was also able to contribute to software that is actively being used to grade online deliverables in a higher education course. Through this, I gained more insight on the open source programming mindset and collaboratively with other developers. Fully understanding the technology that enables various features of an application is essential to experimenting with it effectively. Further, updating and defining queries in the database helped me understand how complex object relationships can get within an application and is a good lesson to learn from for future design problems.

I believe this project has much value for benefitting student learning. The consolidation of information on one page is timesaving for graders and more influential for a student's learning. Collaborative grading is easy and annotation syncing can be initialized with the press of a download button. Communication between instructors, graders, and students is optimized by overlaying the e^{2}logos sidebar on the submission site and enabling inter-grader annotations. Finally, the dashboard serves as an aggregation of feedback for students and grades, supporting teaching practices by providing data on how the projects are going.

Open source software is powerful and easily integrated with other software with its likeness. Examining this technology through the implementation of grading features provided me with a foothold for understanding the programming design, as well as computing and network resources, required to build open source software

References

[1] M. W. Wu and Y. D. Lin, "Open source software development: An overview," *Computer (Long Beach Calif)*, vol. 34, no. 6, pp. 33–38, Jun. 2001, doi: 10.1109/2.928619.

[2] A. Bonaccorsi and C. Rossi, "Why open source software can succeed," *Res Policy*, vol. 32, no. 7, pp. 1243–1258, Jul. 2003, doi: 10.1016/S0048-7333(03)00051-9.

[3] E. Raymond, "The cathedral and the bazaar," Oct. 2005. doi: 10.1007/S12130-999-1026-0.

[4] R. Shaikh-Lesko, "Web annotation tool Hypothesis hits a milestone," *Nature*, vol. 569, no. 7755, p. 295, May 2019, doi: 10.1038/D41586-019-01427-9.

[5] "Home : Hypothesis." https://web.hypothes.is/ (accessed May 08, 2023).

[6] "What is a Container? | Docker." https://www.docker.com/resources/what-container/ (accessed May 08, 2023).

[7] N. Kathare, O. V. Reddy, and V. Prabhu, "A Comprehensive Study of Elasticsearch," *International Journal of Science and Research*, doi: 10.21275/SR21529233126.

[8] "PostgreSQL: About." https://www.postgresql.org/about/ (accessed May 08, 2023).

[9] M. C. English and A. Kitsantas, "Supporting Student Self-Regulated Learning in Problem- and Project-Based Learning," *Interdisciplinary Journal of Problem-Based Learning*, vol. 7, no. 2, p. 6, Sep. 2013, doi: 10.7771/1541-5015.1339.

[10] T. Im, T. Im, and V. Dennen, "Building a Collaborative Knowledge Base in Diigo: %How Links, Tags, and...," in *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare,...*, Chesapeake, VA: AACE, Oct. 2013, pp. 794–797.