Empirical Analysis of Pull Requests for Google Summer of Code

Saheed Popoola
School of Information Technology
University of Cincinnati
USA
saheed.popoola@uc.edu

Abstract

Internship and industry-affiliated capstone projects are popular ways to expose students to real world experiences and bridge the gap between academic training and industry requirements. However, these two approaches often require active industry collaboration, and many students struggle to find industry placements. Open-source contributions are a crucial alternative to gain real world experience, earn publicly verifiable contribution with real-world impact, and learn from experienced open-source contributors. The Google Summer of Code (GSoC) is a global initiative that matches students or new contributors with experienced mentors to work on open-source projects. The program aims to introduce the students to open-source development, help them gain valuable skills under the guidance of mentors, and hopefully encourage them to continue contributing to open-source projects. This study presents an empirical analysis of pull requests created by interns during the GSoC program. We extracted and analyzed 17,232 pull requests from 2,456 interns across 1,937 open-source projects. The results show most tasks involve both code-intensive activities like adding new features and fixing bugs, as well as non-code tasks like updating documentation and restructuring the codebase. The results also show that feedback from reviewers covers code functionality and programming logic, testing coverage, error handling, code readability, and adherence to best practices.

CCS Concepts

• **Social and professional topics** → **Computing education**.

Keywords

open source, pull requests, summer programs, review comments, empirical analysis

ACM Reference Format:
Saheed Popoola. 2025. Empirical Analysis of Pull Requests for Google Summer of Code. In *26th Annual ACM Conference on Cybersecurity and Information Technology Education (ACM SIGCITE 2025), November 6–8, 2025, Sacramento, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3769694.3771151

1 Introduction

Most software engineering education research targets students in formal academic settings. Unfortunately, traditional approaches to software engineering education in these academic settings often limit students' exposure to and engagement with real-world projects, failing to fully harness their potential and creativity [5, 6]. The industry frequently notes that fresh graduates have limited exposure to real-world projects and are often ill-equipped handle industrial challenges [3, 14–16]. Initiatives such as internships and university-industry collaborations on capstone projects have been designed to provide students with valuable industrial experience. However, these initiatives often require active industrial partnerships, which may not be always feasible.

Open-source software communities continuously need new contributors to maintain and sustain their codebase. Contributions to open source can be publicly verified and may demonstrate competence during job searches. Open-source code repositories are abundant, easily accessible, and provide real-world examples that are essential to prepare students for the industry. Therefore, recruiting students to work on open-source code repositories equips the students with industrial experience, provides publicly-verifiable evidence of competence, and helps sustain the pool of contributors needed for open-source projects.

The Google Summer of Code (GSoC)[1] [9] is an annual, global, three-month internship program that connects students to open-source software communities. Participants work on real-world open-source projects under the guidance of experienced mentors. Google provides stipends to student participants after completing certain milestones. The program is the first global initiative to introduce students to open source software development and bring in new contributors. Since 2005, GSoC has connected more than 19,000 new contributors from 112 countries with 18,000 mentors from 133 countries. The program has generated over 43 million lines of code across numerous projects for more than 800 open-source organizations[2]. The code contributions of the participants are publicly available, thereby providing valuable data to study novice activities in open-source development.

Contributions to open-source are often made via pull requests where contributors clone a codebase, make changes (e.g, add a new feature) locally, and then notify the repository team that they have completed a feature or modified the codebase [7]. Collaborators then review and provide comments on the proposed changes. Code review comments provide valuable feedback that guide the contributors and enhance code quality. Maintainers must approve the requested changes before they can be integrated into the main codebase. Although several studies have examined pull requests,

[1]https://summerofcode.withgoogle.com/
[2]https://summerofcode.withgoogle.com/about

there is limited research on the analysis of pull requests created by interns or students such as GSoC participants.

This study analyzes pull requests created by interns during the GSoC program from 2020 to 2023 (a 4-year period). We examine pull request data, review comments, project-related features, and contributors (intern) data to gain informative insights into the contributions interns make, the feedback they receive, and possible implications for software engineering education. The study analyzed 17,232 pull requests by 2,456 interns across 1,937 projects. We used topic modeling techniques to gain insights into common themes in the pull request titles, descriptions and review comments. In summary, the paper provides the following contributions:

(1) A publicly available dataset[3] of 17,232 pull requests created by GSOC interns from 2020 to 2023. This dataset was extracted directly from the official GSoC program page (via web scrapping) and Github (via the GiHub API). The dataset also include the scripts used for web scrapping and extracting the pull requests from GitHub.
(2) Insights into the success or failure of the GSOC program in promoting open-source contributions. Specifically, we provide information on interns' continued contributions to the same project after the program.
(3) An analysis of the tasks interns worked on and the feedback they received from the project collaborators.
(4) A discussion on the implications of these results for software engineering education.

2 Background on Google Summer of Code Program

Google Summer of Code (GSoC) is an annual, global program run by Google that encourages university students or new contributors to participate in open-source software development. The program pairs new contributors with mentors from open-source organizations. The program has two main goals. Firstly, to expose students or novices to open-source development and help them gain real-world experience. Secondly, to provide open-source organizations with new contributors that will continue to contribute to open-source communities even after the program is completed. Google also provides stipends to contributors who successfully meet some specified milestones.

The GSoC program provides an excellent opportunity for students to develop their skills and make meaningful contributions to projects used by people around the world. There are numerous benefits associated with the program and they include the following [21],[24].

(1) **Learning opportunity.** Students gain hands-on experience in software development in a real-world context.
(2) **Networking.** Participants build relationships with mentors and other open-source contributors.
(3) **Career development.** The contributions implemented during the GSoC program are public and easy to verify. Many students saw the GSoC program as an opportunity to build their resume, kickstart their career, and increase their visibility when seeking a job [21]. Furthermore, the program is popular, highly regarded by employers, and many students go on to work for the organizations they contributed to. This makes participation in the program an excellent component of participants' resume when searching for jobs.
(4) **Open-source contribution.** Participants contribute code to open-source projects, which is publicly available and can be used by the global software community.
(5) **Open-source sustainability.** The program provides new contributors to open-source communities which is necessary for sustaining the open-source projects. Tan et al. [24] noted that sustainability of the project was a major motivating factor for mentors to participate in the program.

The GSoC program offers an excellent avenue to introduce students or newcomers to open source, and support the sustainability of open-source communities. Majority of the organizations participating in the program have also reported that the program helps them to find new contributors who are active and committed to the open source project [10]. The contributions of students to the open source community are also public and accessible to anyone. This public data from students' contributions provides a valuable dataset for analyzing the program's impact, understanding challenges or barriers to open-source software development, and probe factors that ensure successful engagement with the open source communities.

3 Literature Review

Several studies have examined the impact of summer coding programs and internships on software development education. However, to the best of our knowledge, none of the studies have provided empirical analysis of pull requests created during summer programs. Silva et al. [19, 21] conducted a qualitative study to understand students' motivations for participating in GSoC program, and they then developed a theoretical framework to model students' engagement. Their findings reveal that students often joined the program to acquire new software engineering skills, enhance their resumes, and earn stipends. Tan et al. [24] explored mentors' motivations in the GSoC program and noted that intrinsic motivation to sustain the open-source codebase, learn new skills, and increased recognition in the open-source community are the major reasons why mentors participate in the program. Trainer et al. [25] analyzed 22 GSoC projects and 22 hackatons to understand student engagements with the open-source scientific community. Their findings reveal that private mentor-student communications often create strong ties, while public sharing of progress reports create weak ties with other community members. Interestingly, 18% of students later became mentors.

Majority of the approach so far have used qualitative study (without considering the codebase) to analyze students and mentors participation in the GSoC program. The closest to our work was conducted by Silva et al. [20, 22] who analyzed the code commits of 866 students in the GSoC program from 2013 to 2015. The authors discovered a strong correlation between the number of commits and students' retention within projects during and after the GSoC program. Additionally, 82% of the projects merged at least one commit from students into their codebases. However, while commits provide granular insights into code changes, they may not fully

[3]https://github.com/compedutech/gsoc

capture task types or objectives. Pull requests on the other hand, contains aggregate changes from multiple commits and provide a high-level data on the kind of tasks the students work on. Pull requests also provide opportunities for other members of the community to review changes, provide comments or feedback, and suggest modifications to the proposed changes [7]. Furthermore, pull requests have been shown to be more precise than commits in evaluating the contributions of community members [1]. Our study extends the work by Silva et al. [20, 22] to use pull requests for understanding students' contributions to the GSoC program.

Internships and industry-based capstone projects play crucial roles in providing students with real-world exposure. Kapoor et al. [11] and Lehman et al. [12] found that internships significantly improves professional development, while Groeneveld et al. [8] noted that both internships and capstone projects promotes the development of soft skills among students. Dean et al. [4] emphasized that the impact of internships on students varies by student involvement levels. Sudol et al. [23] reported that internship experience reduces misconceptions about software engineering concepts. The related work on internships discussed so far have adopted qualitative studies to analyze the impact of internships on students without considering the actual code contributions. The closest work to our study was by Menezes et al. [13] where they described their experience on an internship model that pairs students with volunteer mentors in open-source projects. The authors then conducted a qualitative study using surveys and reports to understand the impact of the internship model on students' skills and job placement. Their results show significant increase in students' technical skills, soft skills, and job placement after graduation. They also noted that students who are paid stipends are more willing to work full time on the project.

This literature review section have discussed studies related to GSoC and internships. Although, these studies provided valuable insights, none of them conducted large-scale analyses of pull requests within the context of internships or summer coding programs. This study fills this gap by examining GSoC participants' pull requests, to offer a comprehensive view of their contributions and feedback.

4 Design Methodology

This section outlines the data collection process, research methodology, and the research questions we aim to answer.

4.1 Data Collection

We extracted pull request data using web scrapping and the GitHub API. The following paragraphs highlights how we extracted the needed data.

(1) The Selenium Python Package was used to scrape the URLs of each intern's personal reflection website on the GSoC web page for the years 2020 to 2023.
(2) The personal reflection websites of the GSoC participants often contained links to their contributions during the program. So, we extracted all URLs from each intern's webpage by locating HTML anchor tags.
(3) URLs containing "GitHub" and either "issues" or "pull" were filtered to identify pull request links. For example the url link "https://github.com/repos/asyncapi/community/pulls/720" contains the words "github" and "pull". Most pull requests URLs on GitHub often follow this pattern and we believe it is the easiest way to extract most of the pull request links.
(4) The GitHub API was used to extract the pull request data associated with the filtered URLs.

The dataset includes three main categories: pull request details, contributor (intern) information, and the project metadata. The final dataset comprises of 17,232 pull requests submitted by 2,456 interns across 1,937 open-source projects. The projects in the dataset have an average of 1,795 stars and 34 branches. The top ten projects based on the number of stars [17] are tensorflow/tensorflow, huggingface/transformers, nodejs/node, rust-lang/rust, godotengine/godot, django/django, opencv/opencv ,tensorflow/models, swiftlang/swift, and webpack/webpack. Furthermore, the 17,232 pull requests includes 60 million lines of code added, 18 million lines deleted, and modifications to over 300,000 files. On average, a pull request remained opened for 4.7 hours, garnered 4.3 comments and received 9.5 review comments. Table 1 provides a statistical description of key pull request features.

4.2 Research Questions

The goal of this study is to provide insights into the pull request activities created by interns during the GSoC program. A detailed analysis of the pull requests can reveal patterns in the contributions interns make to open-source repositories. It may also shed light on how collective thinking emerges among a group of individuals, the core features interns often work on, the type of feedback they frequently receive, and factors that contribute to a successful or unsuccessful GSoC experience. These insights can inform necessary changes to the software engineering curriculum, encourage students to contribute to open source, and better prepare students for the workforce. In this context, this study is guided by the following research questions.

- **RQ1. How successful has the GSoC program been in encouraging contribution to open source communities?** A continuous stream of new contributors is crucial to the sustainability of the open-source projects. One of the major goals of the GSoC program is to encourage contributions to open-source. This research question aims to evaluate how effectively the GSoC program attracts and retains new contributors.
- **RQ2. What are the major tasks that interns often work on?** This question seeks to categorize the tasks or contributions interns commonly engage in. This can provide insights into interns' interest and lay a foundation for attracting new comers to open-source communities.
- **RQ3. What are the major feedback interns often receive on their contributions.** Pull request contributions are typically reviewed by experienced contributors or moderators, who provide comments on how to improve the contributions. These feedback can offer valuable insights into the challenges interns face, identify patterns of mistakes or errors, and inform improvements to the software engineering curriculum to better prepare students for the workforce and open-source contributions.

Statistics	Is_merged	State	Additions	Deletions	Changed Files	Commits	PR comments	Review Comments	Time taken (Hours)
Total	14775	16325	60,049,227	18,668,966	353,946	198,880	74, 343	165,330	14,898,582
Mean	0.857	0.947	3484.95	1083.45	20.5	11.5	4.3	9.5	4.7
Std Dev	0.35	0.22	67352.30	52568.90	263.91	84.04	8.48	26.08	2812

Table 1: Descriptive Statistics of Pull Request Data

These three research questions aims to deepen our understanding of interns' contributions to open-source projects and discuss the implications of the findings for software engineering education.

4.3 Study Design

For the study, we used topic modeling techniques to address the research questions. Topic modeling, or topic clustering, is an unsupervised machine learning method for discovering the underlying thematic structures in textual content. This process generally consists of three key steps: pre-processing, topic modeling, and interpretation of results.

Firstly, we preprocess the text documents to improve the accuracy of topic modeling. Texts were converted to lowercase and cleaned to remove irrelevant characters (such as '/'), punctuations, numbers, and short words with only one or two letters. Then, we applied tokenization to break the text into individual words, followed by removal of stop words (e.g., 'a', 'an' 'the'), and stemming to reduce words to their root forms (e.g., converting 'understanding' to 'understand').

Secondly, the Latent Dirichlet Allocation (LDA) [2], implemented using the Gensim Python library [18], was employed to identify topics in the document. Coherence scores were used to compare a range of topic numbers in order to select the optimal number of topics for each tasks.

Finally, we interpreted the results by associating the generated topics with their respective keywords. We selected posts that contains the identified keywords, manually reviewed the posts to understand the context, and then applied open-coding techniques to label the topics generated from the Gensim package. We adopted a 2-step open-coding process where we first extract the top 20 keywords for each topic identified by the LDA algorithm and then manually assigned labels to the topics based on these keywords. The goal of this final interpretation process is to ensure that we generate meaningful descriptions for the main identified topics.

To answer the first research question, we extracted all the pull requests created by a user in a project. We then separated the pull requests that were created by that user before or during the GSoC program. This allows us to examine the contributions of the user to the project after the GSoC program. It should be noted that we were able to access only the pull requests created by the user in the project(s) they worked on during the GSoC program and not other projects. We were unable to track if the user contributed to other open-source projects outside of the project they worked on during the GSoC program.

To answer the second research question, we applied topic modeling techniques to the title and body of the pull request. The titles convey the purpose and scope of the changes while the body contains the descriptions of the pull requests. The body of a pull request is crucial for providing context and explaining the changes made in the code, allowing reviewers to understand the scope and rationale behind the work. A well-crafted pull request body typically includes a clear explanation of what the pull request does, the reason for the change, and any special instructions for testing or deployment. Therefore, a detailed analysis of pull request titles and bodies (or descriptions) allows us to understand the major tasks or topics the interns worked on.

To answer the third research question, we applied topic modeling to the review comments. The review comments in the dataset contains valuable feedback provided to the interns. These comments are essential for improving the code quality, ensuring adherence to best practices, and enhancing clarity and maintainability. A well-constructed review comment provides constructive criticism, suggests improvements, and sometimes includes code snippets to guide developers. An analysis of these comments can identify key areas where interns frequently encounter challenges and need guidance.

This methodology ensures a structured analysis of interns' contributions, tasks, and feedback, thereby offering valuable insights for improving software engineering education and supporting open-source community engagement.

5 Results

This section presents the results of our analysis with respect to the three research questions discussed in Section 4.2.

5.1 RQ1: GSoC Success

The first research question examines how successful the GSoC program is in promoting open-source contributions. The results of the analysis from Table 1 show that 85.74% (or 14,775 out of 17,232) of the pull requests were accepted and successfully merged to the codebase. Furthermore, 1,113 out of 2,456 users had all their pull requests accepted while 319 had no accepted pull request.

Next, we examined the pull requests created by each user before and after the GSoC program. The results show that 63.5% (1,561 users) contributed at least one pull request to the same project after the program, while 70.7% (1,738 users) contributed at least one pull request to the project before starting the GSoC program. This shows that the majority of users started contributing to the project before the GSoC program and many continued to do so after the program. In fact, only 15% contributed solely during the GSoC program with no contributions before or after the program. This may indicate that the GSoC program motivates students to contribute to open source. Students may also have contributed to an open source project before the program to increase their chances of being selected.

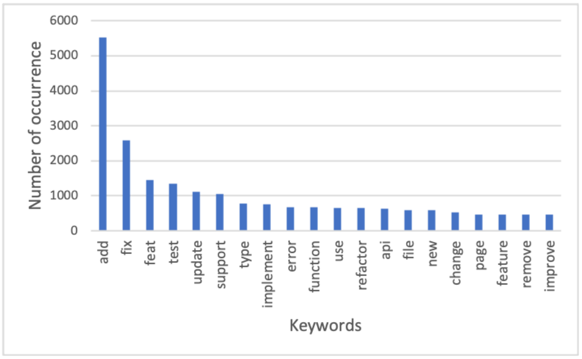

Figure 1: Frequencies of Words in the Title

5.2 RQ2: Tasks and Contributions

We used word frequencies and topic modeling techniques to analyze the titles and descriptions (body) of the pull request with the goal of extracting the types of tasks students worked on. First, we extracted the most frequent words that appear in both the titles and descriptions. Figure 1 illustrates the most recurring word in the titles. Next, we used LDA topic modeling to extract the major topics in the descriptions and the key words that capture these topics. Thirdly, we extracted sample text from the documents containing the keywords from the word frequency and topic modeling in order to understand the context of the words. Finally, we categorized the topics based on the keywords and the context in which they appear. The extracted topics for the tasks are.

(1) **Adding new features and APIs.** This topic involves developing and implementing new features like knowledge panels or API endpoints. Sample pull requests with this topic include: "Added support for opening a file", "Added custom IDE and improved other IDE launch commands", and "Add an explore mode to run the app without marking messages as read".
(2) **Corrective maintenance tasks.** This topic covers pull requests that attempt to correct an error or ensure a feature is working as expected. Examples of pull requests in this topic include "Fix double-checked locking in ConnectionFactory", "Fix axeDev tools + lighthouse error for learner dashboard and make the lighthouse score to 1", and "Add and update tests".
(3) **Project setup.** This topic covers pull requests that deal with initializing the project structure and adding necessary dependencies. Examples of pull requests in this topic include: "Setting up the foundational infrastructure for FastAPI", "Code relocated under maryam/", "added setup.py, src: setup: Add root install support", "Initial PlanetGenerator Setup", and "Created the spring boot project with the necessary dependencies".
(4) **Documentation updates** This topic covers pull requests that relate to making changes to the README file or other documentation files to ensure clarity and up-to-date information. Sample pull requests include "Updating or adding information to ReadMe files", "Privacy Statement Updates September 2022", "Added basic chondro documentation", "restructured install documentation".
(5) **Refactoring.** This topic covers pull requests that aim to change parts of the codebase without changing behavior. Sample topics include "wxGUI refactoring: New WorkspaceManager class", "refactor: color blending function", "refactoring of webapp to consume typescript client", and "[GSoC] API change for the newly added system of ODEs solvers".

The analysis of the second research question shows that while majority of the contributions are code-intensive, a significant portion of the pull requests involves changes to the documentation files and restructuring the organization of the codebase. This may also indicate the need to emphasize the variety of (code and non-code) contributions students can make to open source.

5.3 RQ3: Feedback and Review Comments

To answer the third research question, we also applied word frequencies and topic modeling techniques to the review comments to understand the type of feedback reviewers provide for the pull requests. The topics identified in the review comments include the following.

(1) **Code organization and structure improvements.** This topic emphasizes project structure and organization, particularly where code files, tests, and other assets should be placed. This helps to maintain a clean project hierarchy, improve readability, and simplify navigation for future developers. Sample comments include "You should put test in a tests/ directory" and "If you move this to the test directory, it becomes clearer".
(2) **Best practices and coding style.** This topic includes comments that suggest changes that align with best practices, such as ensuring minimal exposure of unnecessary components and adherence to consistent coding styles. This helps ensure code reliability, consistency, and security. Sample comments include "I always prefer if we keep safe and only publish necessary things" and "Why not just use a str for facet_value?".
(3) **Functionality and logic corrections.** This topic relates to the clarity of logic or ensuring the code behaves as intended. Reviewers help identify sections where functionality may not align with the expected outcome or where the logic could be improved. Sample comments include "This is more the why of knowledge panels" and "The logic here could be improved by changing this function"
(4) **Testing coverage and validation.** This relates to adding more tests or moving test cases to appropriate locations to ensure the functionality is well-covered. They emphasize the importance of testing edge cases and ensuring that new features are adequately validated before being merged. Sample comments include "You should write a test for this specific scenario" and "This becomes easier to test if you move it into a dedicated test case"
(5) **Error handling and resilience.** This topic focuses on ensuring that the code is resilient to errors and can handle potential failure cases gracefully. Reviewers point out where

additional checks or error handling mechanisms can be implemented to improve robustness. Sample comments are “Make sure we handle this case properly when the data is missing" and “Consider adding a check for this specific error".

(6) **Code readability and clarity.** This topic relates to requests for improvements in code readability, suggesting changes that make the code more understandable. This involves better naming conventions, simpler logic flows, and clearer documentation or comments. Sample review comments include “This part of the code could be written more clearly" and “The variable names should better reflect their purpose".

The result presented above shows some key areas that reviewers have focused on. It can be observed that reviewers are not only concerned about the correctness of the code, but are also interested in ensuring that the newly added or modified code is easy to maintain.

6 Discussion and Implications for IT Education

The results of the analysis on GSoC pull requests and review comments have significant implications for software engineering education. The identified tasks, and feedback from review comments can help educators better prepare students for real-world software development practices. The findings highlight key areas that should be emphasized in IT curricula as explained below.. Firstly, The study underscores the critical role of peer reviews in ensuring the quality of code contributions. For software engineering education, this suggests integrating code reviews into course projects and emphasizing communication skills to facilitate effective collaboration. Next, the findings show that reviewers frequently comment on the structure and organization of code, such as file placement and module organization. This highlights the need to emphasize code organization in software engineering education by 1) teaching modular design and file organization, and 2) following real-world practices in project layout within software engineering courses. Furthermore, many of the review comments relates to issues with error handling, inadequate testing, and highlights the importance of providing and receiving feedback. This shows the need for students to have testing and collaboration skills. Finally, many findings reflect practices commonly observed in open-source development environments, such as how to use Git. We recommend that educators should use real-world GitHub projects as teaching tools, and incorporate version control systems to stimulate collaboration in software engineering classes.

The findings of this study offer several key insights for software engineering education. When educators focus on real-world practices such as code organization, testing, error handling, performance optimization, and collaboration; they can better prepare students for the challenges of the software industry. The integration of code reviews, refactoring exercises, and open-source contributions into the curriculum will likely foster a culture of continuous improvement and collaboration. This approach equips future software engineers with the skills and mindset they need for success. These educational adjustments will help students internalize best practices in software development, become better collaborators, and emerge as thoughtful, effective developers in both open-source and proprietary software environments.

7 Threats to Validity

There are several potential threats to the validity of our findings. These threats can affect the reliability and generalization of the results. Firstly, pull request descriptions vary widely in quality and completeness. Some pull request may lack detailed explanations of the changes, leading to incomplete or inaccurate summaries of tasks. Secondly, the study may not generalize well to all software projects. Different types of software projects (e.g., open-source vs. proprietary, small vs. large projects, front-end vs. back-end development) have distinct workflows, contributor behaviors, and review cultures. Therefore, findings from a single type of project may not apply broadly. Finally, the dataset used for the study was sourced solely from GitHub repositories of projects that participated in the GSoC program. Since this program is being run by only Google, there is a possibility that the projects follow specific coding practices, use particular technologies, or operate in niche domains. Hence, the findings may not be representative of broader software development practices.

The study provides valuable insights into pull requests and code review comments. However, the validity of the findings could be affected by the aforementioned threats. We have used various approaches (such as manual validation of some of the comments) to minimize these threats and ensure careful interpretation of the results. In the future, we will expand the dataset, refine the analytical techniques, and account for variability in development practices to enhance the validity and generalizability of the findings.

8 Conclusion

This study provides an in-depth analysis of GitHub pull requests and review comments created during the Google Summer of Code program from 2020 to 2023. The study offers insights into common tasks and feedback themes. The results shows that pull requests frequently focus on feature additions, bug fixes, and documentation updates. The review comments often emphasize the need for improved code organization, additional tests, performance optimizations, and better error handling.

The results highlights the need to enhance the software engineering curriculum by incorporating a stronger emphasis on code reviews, code organization, testing, and best practices from open-source repositories. The findings can also help development teams that work with interns to enhance their review practices by focusing on key areas such as test coverage, performance improvements, and how to maintain a well-structured codebase.

In the future, we plan to study the evolution of tasks and comments across multiple years. We also plan to conduct surveys and interviews with the GSoC participants to augment the findings reported in this study. Finally, we intend to explore other summer programs and hackathons that target students to validate the generalizability of this study’s findings across different repositories and software domains.

References

[1] Marcus Vinicius Bertoncello, Gustavo Pinto, Igor Scaliante Wiese, and Igor Steinmacher. 2020. Pull requests or commits? which method should we use to study contributors’ behavior?. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER).* IEEE, 592–601.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[3] Eric Brechner. 2003. Things they would not teach me of in college: what Microsoft developers learn later. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. 134–136.

[4] Christopher Dean, Thomas D Lynch, and Rajiv Ramnath. 2011. Student perspectives on learning through developing software for the real world. In *2011 Frontiers in Education Conference (FIE)*. IEEE, T3F–1.

[5] Vahid Garousi, Görkem Giray, Eray Tüzün, Cagatay Catal, and Michael Felderer. 2019. Aligning software engineering education with industrial needs: A meta-analysis. *Journal of Systems and Software* 156 (2019), 65–83.

[6] Vahid Garousi, Gorkem Giray, Eray Tuzun, Cagatay Catal, and Michael Felderer. 2019. Closing the gap between software engineering education and industrial needs. *IEEE software* 37, 2 (2019), 68–77.

[7] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An exploratory study of the pull-based software development model. In *Proceedings of the 36th international conference on software engineering*. 345–355.

[8] Wouter Groeneveld, Joost Vennekens, and Kris Aerts. 2019. Software engineering education beyond the technical: A systematic literature review. *arXiv preprint arXiv:1910.09865* (2019).

[9] Leslie Hawthorn. 2008. GOOGLE SUMMER OF CODE. *The Open Source Business Resource* (2008), 5.

[10] Mr Andreas Hornig, Dieter Fritsch, Mr Kai Wilke, Mr Manfred Ehresmann, Ms Blagica Jovanova, Mario Merino, and Mr Ulrich Beyermann. 2019. SUMMER OF CODE: BRINGING TOGETHER STUDENTS WITH OPEN-SOURCE SPACE ORGANIZATIONS. In *70th International Astronautical Congress*. 1–10.

[11] Amanpreet Kapoor and Christina Gardner-McCune. 2019. Understanding CS undergraduate students' professional development through the lens of internship experiences. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 852–858.

[12] Kathleen J Lehman, Kaitlyn N Stormes, Katie N Smith, and Julia C Lapan. 2024. Sealing the Deal: Factors that Promote Computing Interns' Interest in Computing Careers. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. 715–721.

[13] Tyler Menezes, Alexander Parra, and Mingjie Jiang. 2022. Open-Source Internships With Industry Mentors. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*. 365–371.

[14] Pan-Wei Ng and Shihong Huang. 2013. Essence: A framework to help bridge the gap between software engineering education and industry needs. In *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 304–308.

[15] Alex Radermacher and Gursimran Walia. 2013. Gaps between industry expectations and the abilities of graduates. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 525–530.

[16] Alex Radermacher, Gursimran Walia, and Dean Knudson. 2014. Investigating the skill gap between graduating students and industry expectations. In *Companion Proceedings of the 36th international conference on software engineering*. 291–300.

[17] Baishakhi Ray, Daryl Posnett, Vladimir Filkov, and Premkumar Devanbu. 2014. A large scale study of programming languages and code quality in github. In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*. 155–165.

[18] Radim Rehurek and Petr Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3, 2 (2011), 2.

[19] Jefferson Silva, Igor Wiese, Daniel M German, Christoph Treude, Marco Aurélio Gerosa, and Igor Steinmacher. 2020. A theory of the engagement in open source projects via summer of code programs. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 421–431.

[20] Jefferson De Oliveira Silva, Igor Scaliante Wiese, Daniel M German, Igor Fabio Steinmacher, and Marco Aurélio Gerosa. 2017. How long and how much: What to expect from summer of code participants?. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 69–79.

[21] Jefferson O Silva, Igor Wiese, Daniel M German, Christoph Treude, Marco A Gerosa, and Igor Steinmacher. 2020. Google summer of code: Student motivations and contributions. *Journal of Systems and Software* 162 (2020), 110487.

[22] Jefferson O Silva, Igor S Wiese, Igor Steinmacher, and Marco A Gerosa. 2017. Students' engagement in open source projects: an analysis of Google Summer of Code. In *Proceedings of the XXXI Brazilian Symposium on Software Engineering*. 224–233.

[23] Leigh Ann Sudol and Ciera Jaspan. 2010. Analyzing the strength of undergraduate misconceptions about software engineering. In *Proceedings of the Sixth international workshop on Computing education research*. 31–40.

[24] Xin Tan, Minghui Zhou, and Li Zhang. 2023. Understanding mentors' engagement in OSS communities via google summer of code. *IEEE Transactions on Software Engineering* 49, 5 (2023), 3106–3130.

[25] Erik H Trainer, Chalalai Chaihirunkarn, Arun Kalyanasundaram, and James D Herbsleb. 2014. Community code engagements: summer of code & hackathons for community building in scientific software. In *Proceedings of the 2014 ACM International Conference on Supporting Group Work*. 111–121.