The International Review of Multidisciplinary Research
Volume 1, Issue 4, April 2026, pp. 254-267
https://doi.org/10.5281/zenodo.19603143

GitHub Repository Workflows in Software Engineering Education: A Structured Review of Uses, Outcomes, and Challenges

Reymond C. Dalupang
University of La Salette, Inc.
rdalupang@uls.edu.ph

Article Details:

Received: 22 January 2026
Revised: 28 March 2026
Accepted: 10 April 2026
Published: 16 April 2026
Corresponding Email:
rdalupang@uls.edu.ph

Recommended Citation:

Dalupang, R. C. (2026). GitHub Repository Workflows in Software Engineering Education: A Structured Review of Uses, Outcomes, and Challenges. The International Review of Multidisciplinary Research. 1 (4), 254-267. https://doi.org/10.5281/zenodo.19603143

Index Terms:

version control pedagogy, collaborative software development, repository-supported learning, pull-request feedback, classroom automation, programming education, educational technology

Abstract. This structured review examines how GitHub is used in software engineering and programming education and what comparisons can be made with related classroom workflows and platforms. The review used source-by-source searching, staged screening, common extraction fields, source-role weighting, light appraisal, and narrative thematic synthesis. Searches were updated in March 2026 across major academic databases, ERIC, CEUR Workshop Proceedings, GALA, and openly accessible institutional or author-posted sources. Preliminary screening identified 127 candidate records. After staged screening, 58 accessible full texts were assessed, and 44 sources were retained: 38 substantive review sources and 6 method-guidance sources. The literature shows that GitHub is commonly used for assignment delivery, team projects, code review, feedback, autograding, and portfolio development, often supporting both individual learning and teamwork in realistic development environments. Reported benefits include better process visibility, clearer feedback placement, enhanced support for collaboration, and stronger workflow learning that integrates social coding, traceable contributions, and structured peer interaction. Common difficulties include the Git learning curve, setup burden, uneven student readiness, and the weak use of activity counts as direct evidence of learning. Comparative conclusions are clearest against basic file-submission workflows, emphasizing GitHub's role in organizing course materials and feedback, and more cautious for GitLab and Bitbucket because direct classroom comparisons remain limited and context-dependent. Overall, GitHub appears most useful when it is introduced with clear guidance, aligned with course objectives, and supported with guidance for onboarding, workflow management, and assessment interpretation. This review contributes to educational technology research by synthesizing multi-source evidence, clarifying both the pedagogical potential and adoption constraints of repository-supported learning, and providing instructors with a detailed understanding of how platform features, classroom practices, and learning outcomes intersect in software engineering education.

Introduction

GitHub is one of the most visible tools in modern software work. It supports version control, issue tracking, code review, documentation, workflow automation, and public portfolio building. Because of this, many teachers have brought GitHub into programming and software engineering courses. The basic rationale is straightforward: if students learn inside a tool that resembles real development practice, they may develop not only coding skill, but also better habits for teamwork, feedback, and project management. Early work by Dabbish et al. (2012) and Zagalsky et al. (2015) already suggested that GitHub is more than a repository host. It is also a social and collaborative space where work is visible, discussable, and easier to coordinate.

In education, GitHub became even more attractive as GitHub Classroom matured and more classroom studies began to report how it was used. Recent studies describe GitHub Classroom as a workflow that can support assignment templates, individual or group repositories, progress checking, pull-request feedback, and autograding through GitHub Actions. This

lets instructors keep materials, submissions, review, and revision closer to one workspace instead of splitting them across email, learning management systems, zip files, and manual checking (Bennedsen et al., 2022; Hecht et al., 2023; Blair, 2026).

Even so, the educational literature on GitHub is scattered across many kinds of studies. Some papers focus on student experience, some on teacher practice, some on acceptance and motivation, some on version-control pedagogy, and some on activity traces such as commits and pull requests. A smaller body of work also compares or relates GitHub to GitLab, Bitbucket, or general version-control teaching. Because this literature is spread across different study types, teachers who want evidence-based guidance often have to piece the picture together.

Recent review discussions have begun to focus more narrowly on GitHub Classroom itself (Blair, 2026). That shift is valuable, but it still leaves a wider question open: how should GitHub Classroom evidence be read alongside older GitHub studies, nearby repository platforms, and reports from settings where access, readiness, and support vary? The present review therefore widens the lens and includes adjacent classroom workflows as part of the comparison.

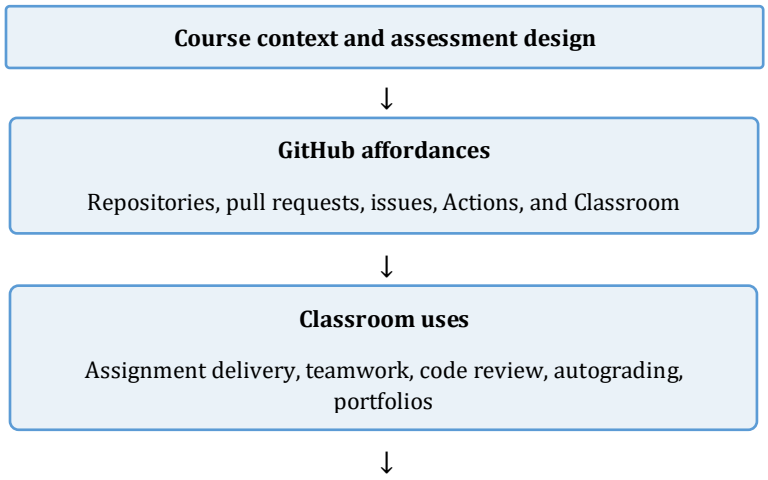
Accordingly, this review addresses four questions: how GitHub is being used in software engineering and programming education, what outcomes are being reported, what challenges recur across studies, and what can reasonably be concluded when GitHub is read alongside traditional submission methods and nearby platforms such as GitLab and Bitbucket. The review avoids strong platform ranking because the evidence is uneven. Instead, it separates stronger classroom evidence from supporting material and shows where comparison is stronger or still limited.

The main contribution of this review is its layered synthesis. Instead of looking only at GitHub Classroom or treating all sources as if they carry the same weight, it brings together direct classroom studies, foundational work on GitHub in education, feature-level workflow studies, and contextual evidence on adoption conditions. It also separates classroom evidence from method guidance and background fact-checking so that weaker support material is not mistaken for proof of educational effect. This helps clarify what GitHub is clearly helping with, what remains uncertain, and why onboarding and teaching conditions still matter.

Conceptual Framework

Because this review synthesizes studies with different designs, samples, and outcome measures, it uses a conceptual rather than a single-theory framework. The framework draws on prior work that treats GitHub as a visible social-coding environment and on classroom studies that connect repository features to learning practice (Dabbish et al., 2012; Zagalsky et al., 2015; Tu et al., 2022). It links five recurring elements in the literature: course context, GitHub affordances, classroom uses, learning processes, and reported outcomes.

Figure 1 shows this logic in a direct way. GitHub does not improve learning by itself. Its effects in the literature usually appear when teachers connect specific platform features, such as repositories, pull requests, issues, and GitHub Actions, to specific classroom uses, such as assignment delivery, teamwork, code review, or autograding. These uses then shape learning processes like visibility of work, traceability of contribution, workflow practice, and feedback exchange. The reported outcomes in the literature, such as collaboration, workflow competence, engagement, and readiness for project work, grow out of that chain (Dabbish et al., 2012; Zagalsky et al., 2015; Hsing & Gennarelli, 2019; Tu et al., 2022; Hecht et al., 2023; *Blair, 2026).*



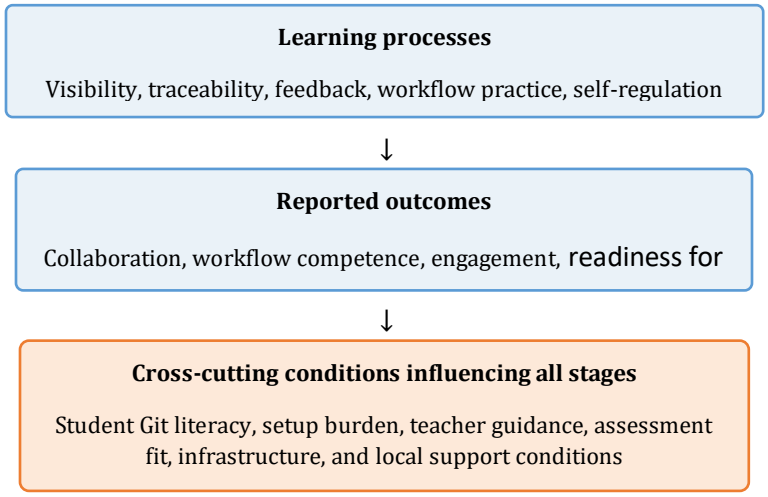


Figure 1. Conceptual framework used to organize the review.

The framework also clarifies why similar tools can produce different classroom results. Across the reviewed studies, outcomes are shaped by student Git literacy, setup burden, uneven access to devices or internet, teacher support, task design, and the limits of using simple activity counts as direct evidence of learning.

Methodology

Review Design

This study used a structured review with narrative thematic synthesis, light appraisal for interpretive weighting, and explicit source-role weighting. The design suited a mixed body of sources, including classroom studies, experience reports, feature-level studies, contextual papers, and review-guidance material. A common extraction frame was applied across the included sources, and the final discussion was written so that stronger claims rested mainly on direct classroom evidence while more tentative observations were reserved for adjacent or thinner evidence. The review did not treat all sources equally. It grouped them by evidentiary role and used that in drawing conclusions, especially in the cross-workflow discussion.

Review Questions

The review was guided by four questions: (1) what classroom uses of GitHub are reported in the literature; (2) what learning, teaching, or workflow outcomes are associated with those uses; (3) what difficulties recur during adoption; and (4) what can reasonably be concluded when GitHub is discussed alongside traditional file submission and nearby repository platforms such as GitLab and Bitbucket.

Search Strategy and Sources

Searching was conducted source by source across each academic database or open-access archive. Searches were updated in March 2026, and eligible publicly accessible studies available at screening were considered. When a relevant March 2026 study was available and retrievable, it was considered for inclusion; when none was available for a topic, the most recent eligible study available at screening was retained. Each source was searched separately rather than through one exported master query because the review drew from multiple databases and open scholarly archives with different search and export behavior. The main sources were Google Scholar, ACM Digital Library, IEEE Xplore, ScienceDirect, SpringerLink, Wiley Online Library, Taylor & Francis Online, ERIC, CEUR Workshop Proceedings, and the University of Greenwich Academic Literature Archive (GALA), together with openly accessible institutional or author-posted versions of eligible studies. Platform documentation was checked only during drafting for terminology and feature verification and was not retained in the final included-source set.

To keep the process auditable, the author maintained a structured screening log that recorded the source searched, the query block used, the date checked, the screening decision, and notes on merge, exclusion, or retention. Appendix A

summarizes the searched sources, representative query families, staged counts, grouped non-retention categories, and the final evidence layers so that readers can see what was searched, how decisions were staged, and why records were not carried forward.

Eligibility Criteria

Sources were included if they examined GitHub or a related repository workflow in a teaching or learning setting and provided empirical findings, implementation experience, foundational insight, feature-level evidence, or classroom-relevant context. Eligible sources included journal articles, conference papers, experience reports, technical reports, and clearly attributable open versions of papers. Sources were excluded if they discussed Git or GitHub without an educational context, duplicated a more complete version of the same study, provided too little methodological or descriptive detail for comparison, relied only on vendor feature descriptions, or lacked an openly accessible full text or substantive public page. Platform documentation was used only to check terminology or feature names and was not retained as formal included evidence. Only sources whose substantive content could be read directly in a public version were kept in the final set of sources.

Preliminary source-page and title screening identified 127 candidate records. Duplicate or less complete versions were merged during title and source screening, and clearly off-topic or non-substantive records were removed during relevance screening. This left 58 openly accessible full texts or substantive public pages for detailed assessment. Of these, 38 sources were retained as substantive review evidence and 6 method-guidance sources were retained to frame the review design, screening logic, and synthesis boundaries. Fourteen items were read at the full-text stage but were not carried into the final included set because they fell into one or more broad categories: duplicate or less complete version, weak educational fit, insufficient methodological or descriptive substance, documentation-only support material, or inadequate relevance to the review questions after full-text reading. The final included set therefore comprised 44 sources, as reported in Appendix B.

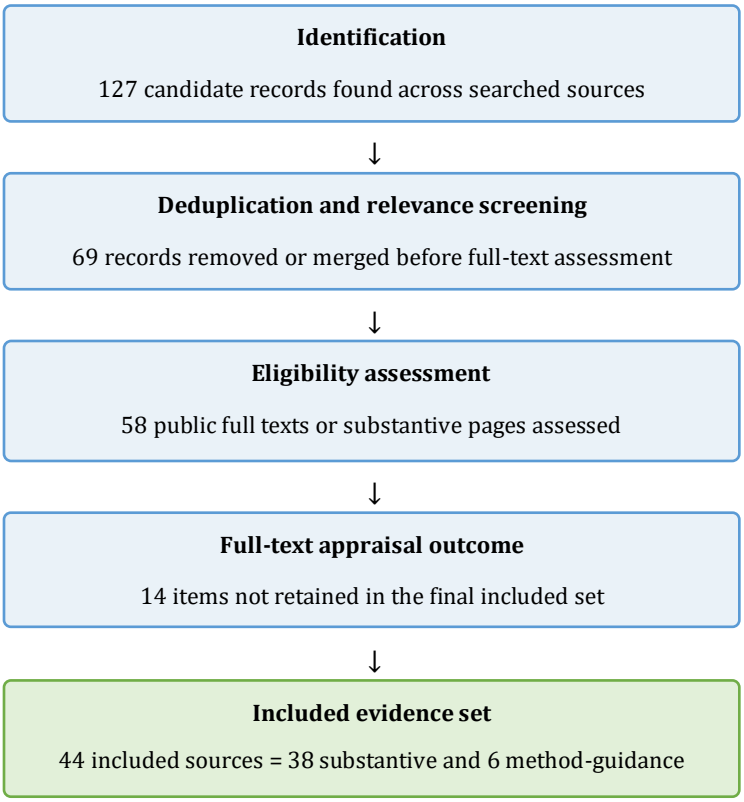


Figure 2. Review workflow and screening summary

Source Selection, Coding, and Synthesis

Each included source was coded using common extraction fields: publication type, educational setting, platform or feature focus, method, sample or cohort details where stated, reported benefits, reported difficulties, and stated limitations. This common frame helped keep comparison consistent across different study types. To keep the synthesis proportionate to the evidence, the review distinguished between direct classroom studies, adjacent educational or analytics studies, contextual or local sources, foundational sources, and method-guidance references. The Results and Discussion were based mainly on the 38 substantive sources. The 6 method-guidance sources were used only to support the review method and reporting, not to show classroom effects. No meta-analysis or effect-size synthesis was attempted because the included studies varied substantially in design, context, outcome measures, and reporting.

Scope and Reporting Limits

The reported counts provide the audit trail of this structured multi-source review. Because the searched interfaces differed in export behavior, the retained screening log rather than a single database dump served as the common audit record across sources. The review therefore reports stabilized counts, decision stages, representative query families, grouped non-retention categories, appraisal dimensions, and included-source roles in a transparent and checkable way, while avoiding claims that would require a fully registered systematic-review workflow.

Results and Discussion

Uses of GitHub in Software Engineering Education

The interpretive claims in this section are based mainly on the 38 substantive sources summarized in Table 1. The 6 method-guidance sources support the review method and reporting but are not used as evidence of classroom effects. Across these 38 substantive sources, GitHub is not treated as a single-purpose classroom tool. It appears as a submission channel, collaboration space, review environment, automation layer, and sometimes a learning target in its own right because students are expected to learn version-control practice (Zagalsky et al., 2015; Feliciano et al., 2016; Hsing & Gennarelli, 2019; Tu et al., 2022; Hecht et al., 2023; Blair, 2026).

One recurring classroom pattern is assignment distribution and collection through template repositories and student or team repositories. Bennedsen et al. (2022), Snowberger and Lee (2023), Adam (2024), Blair (2026), and Hecht et al. (2023) describe how GitHub Classroom keeps starter code, instructions, submission history, and feedback close to the same workspace, reducing the split between assignment materials and submitted work.

Collaborative project work is another major use. GitHub makes teamwork easier to observe because students can branch, merge, open pull requests, discuss changes, and review one another's work. Tushev et al. (2020), Tu et al. (2022), Feliciano et al. (2016), and Patani et al. (2024) show why this matters in project-based courses: instructors can inspect process rather than only final output, while teams gain a clearer structure for distributing and integrating work.

Another common use is feedback and code review. Pull requests provide a shared space for comments, clarification, and revision on specific files and lines. In classroom settings, this is useful because feedback stays attached to the code instead of being separated into email or a grading sheet. Hsing and Gennarelli (2019), Tu et al. (2022), Adam (2024), and Blair (2026) all point to the practical value of that arrangement.

Automation has become more visible as GitHub Classroom evolved. Hecht et al. (2023) show how GitHub Classroom can support distribution, collection, and autograding inside one course workflow, while Bennedsen et al. (2022) and Blair (2026) describe the teaching value of having submission, checking, and revision happen in the same environment.
Another use is employability and portfolio development. Adam (2024), Beckman et al. (2021), Patani et al. (2024), and Fernandez et al. (2025) show that repository work is not just for submission. It also helps students build workflow habits, visible project histories, and outputs they can use beyond one course.

Reported Outcomes

Collaboration is the clearest outcome theme in the reviewed literature. GitHub does not merely allow several students to touch the same project; it records the sequence of that work through commit history, pull requests, comments, branches, and issue links. Those traces support awareness, accountability, and negotiation in team settings (Dabbish et al., 2012; Zagalsky et al., 2015; Feliciano et al., 2016; Hsing & Gennarelli, 2019; Tushev et al., 2020).

Another consistent outcome is workflow competence. Many studies treat GitHub not only as a container for course files but also as a practical way to learn branching, committing, merging, review routines, and repository hygiene. Beckman et al. (2021), Haaranen and Lehtinen (2015), Clifton et al. (2007), and Lawrance et al. (2013) all support the argument that repository work can be taught as an explicit part of course learning rather than as invisible infrastructure.

The literature also suggests that GitHub improves feedback placement, and often feedback quality, by keeping comments close to the code itself. Rather than separating evaluation from the artifact, GitHub allows feedback on lines, files, pull requests, and workflow results. Hsing and Gennarelli (2019), Adam (2024), Blair (2026), and Tu et al. (2022) all describe practical advantages in this arrangement.

Several studies further point to stronger engagement and a greater sense of ownership. A repository history gives students a visible record of progress, and public or semi-public work can encourage more careful revision when the course is designed well. Patani et al. (2024), Adam (2024), and Hsing and Gennarelli (2019) all report this theme in different ways, although mostly in single-course or perception-based settings.

Readiness for project-based and professional work is another repeated outcome. The reviewed studies do not claim that a single course instantly produces industry-ready graduates. They do show, however, that GitHub introduces students to the language and routine of collaborative development - repository setup, branching, review, issue tracking, and automation - in ways that reduce the gap between classroom expectations and workplace practice (Beckman et al., 2021; Bennedsen et al., 2022; Tu et al., 2022; Snowberger & Lee, 2023).

However, the outcome evidence should be interpreted carefully. Many studies rely on perceptions, small groups, or a single course. Although positive findings are common, the studies do not all provide the same level of evidence. Hsing and Gennarelli (2019), Tushev et al. (2020), Adam (2024), and Patani et al. (2024) are useful, but they differ in design and level of comparison. The evidence supports careful claims about collaboration, workflow learning, and feedback, but not universal achievement gains across all settings.

Recurring Challenges

The most common challenge is the difficulty of learning the system at first. GitHub depends on Git concepts that are unfamiliar to many beginners: repositories, local versus remote state, staging, commits, branches, merges, conflicts, and sometimes command-line work. Cochez et al. (2013), Haaranen and Lehtinen (2015), Eraslan et al. (2020), and Wagner and Thurner (2025) all show that repository tools can feel like added complexity unless the course provides deliberate onboarding.

Setup is another common challenge. Before students can benefit from the platform, they may need to create accounts, install Git, configure editors, connect credentials, and learn how local work appears online. Instructors also need to prepare starter repositories, tests, grading rules, and support materials. Bennedsen et al. (2022), Tu et al. (2022), Hecht et al. (2023), and Blair (2026) all suggest that adoption works best when setup is simplified, practiced, and checked before the first graded task.

Another challenge is how repository activity is interpreted. Commit counts, pull requests, and issue activity are useful traces, but they do not automatically show understanding, code quality, or fair contribution. Guerrero-Higueras et al. (2020), Wessel et al. (2023), and Canale et al. (2024) show that these traces can support evaluation or prediction only when they are interpreted carefully and used with stronger assessment evidence.

Prior knowledge also affects adoption. Some students enter a course already comfortable with GitHub, while others have never resolved a merge conflict or used the command line. Cochez et al. (2013), Tu et al. (2022), Olipas et al. (2021), and Olipas (2022) show that learner readiness matters and that GitHub can widen differences when a course assumes skills that some students do not yet have.

Another common issue is instructor workload, especially at the start of adoption. GitHub can save time later through automation, clearer submission workflows, and better traceability, but the first round of course redesign is often heavier. Tu et al. (2022), Bennedsen et al. (2022), Blair (2026), and Hecht et al. (2023) all show that teachers need templates, naming conventions, feedback plans, and sometimes different grading routines before the workflow becomes efficient.

Cross-Workflow Comparison

This section gives a comparison across workflows rather than a direct platform ranking. In the reviewed literature, the clearest comparison is between GitHub-based workflows and traditional file submission. GitHub is often seen as stronger

because it keeps both the process and the final output in one place. Instead of submitting only a final zip file, students work in a repository that shows ongoing progress. Teachers can see the development process, not just the last version, and feedback can be attached directly to code, commits, pull requests, and issues. This helps explain why GitHub is often linked to better traceability, clearer feedback, and more visible teamwork.

Direct comparison with GitLab and Bitbucket is still limited. All three platforms support repository-based collaboration, but the main issue is not missing features. It is the uneven evidence in the literature. GitHub has more classroom-based studies, while GitLab and Bitbucket appear more often in related educational, analytics, or practitioner discussions than in direct classroom comparisons. Because of this, the review does not make a firm platform ranking. Instead, it shows where the available evidence is stronger and where it is still too limited for stronger conclusions.

In the reviewed sources, GitLab appears mostly in related course studies, not in direct classroom comparisons with GitHub. Cortés Ríos et al. (2019), Eraslan et al. (2020), and Schauer et al. (2024) show that GitLab can support consultation, learning analytics, and workflow monitoring. This makes GitLab useful for cautious comparison, but the classroom evidence is still much smaller than for GitHub. Bitbucket appears even less often in education-specific studies, so it is best treated as a possible alternative rather than a platform with equally strong classroom evidence.

Some studies compare GitHub features. Some examine repository use in general, while others focus on pull requests, discussions, automation, or workflow analytics. Yu et al. (2016), Hata et al. (2022), and Wessel et al. (2023) show how these features affect review, communication, and workflow behavior. Although these are not classroom studies in the strict sense, they still matter because teachers increasingly use the same features for assessment and coordination.

No.	Study	Evidentiary role	Synthesis use in the review
1	Adam, E. (2024)	Direct classroom evidence	Used in the main thematic synthesis for classroom uses, reported outcomes, and recurring adoption challenges because it provided direct teaching or learner evidence.
2	Alimboyong, C. R., & Bucjan, M. E. (2021)	Context	Used to interpret institutional readiness, infrastructure, and not used alone to claim platform effects.
3	Beckman, M. D., Çetinkaya-Rundel, M., Horton, N. J., Rundel, C. W., Sullivan, A. J., & Tackett, M. (2021)	Direct classroom evidence	Used in the main thematic synthesis for classroom uses, reported outcomes, and recurring adoption challenges because it provided direct teaching or learner evidence.
4	Bennedsen, J., Böjttjer, T., & Tola, D. (2022)	Direct classroom evidence	Used in the main thematic synthesis for classroom uses, reported outcomes, and recurring adoption challenges because it provided direct teaching or learner evidence.
5	Blair, S. (2026)	Direct classroom evidence	Used in the main thematic synthesis for classroom uses, reported outcomes, and recurring adoption challenges because it provided direct teaching or learner evidence.
6	Bringula, R. P., Geronimo, S., & Aviles, A. (2020)	Context	Used to interpret institutional readiness, infrastructure, and not used alone to claim platform effects.
7	Calatrava Arroyo, A., Ramos Montes, M., & Segrelles Quilis, J. D. (2021)	Adjacent support evidence	Used to refine careful comparisons with non-GitHub workflows or adjacent repository practices when direct classroom comparison evidence was limited.
8	Canale, L., Cagliero, L., Farinetti, L., & Torchiano, M. (2024)	Feature / analytics evidence	Used to explain how specific features such as pull requests, dashboards, analytics, automation, or trace data shaped workflow and feedback claims.

No.	Study	Evidentiary role	Synthesis use in the review
9	Clifton, C., Kaczmarczyk, L. C., & Mrozek, M. (2007)	Foundational basis	Used to define social coding, visibility, collaboration, and version-control pedagogy concepts that anchored interpretation across later classroom studies.
10	Cochez, M., Isomöttönen, V., Tirronen, V., & Itkonen, J. (2013)	Direct classroom evidence	Used in the main thematic synthesis for classroom uses, reported outcomes, and recurring adoption challenges because it provided direct teaching or learner evidence.
11	Cortés Ríos, J. C., Kopec-Harding, K., Eraslan, S., Page, C., Haines, R., Jay, C., & Embury, S. M. (2019)	Adjacent support evidence	Used to refine careful comparisons with non-GitHub workflows or adjacent repository practices when direct classroom comparison evidence was limited.
12	Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012)	Foundational basis	Used to define social coding, visibility, collaboration, and version-control pedagogy concepts that anchored interpretation across later classroom studies.
13	Eraslan, S., Cortés Ríos, J. C., Kopec-Harding, K., Embury, S. M., Jay, C., Page, C., & Haines, R. (2020)	Adjacent support evidence	Used to refine careful comparisons with non-GitHub workflows or adjacent repository practices when direct classroom comparison evidence was limited.
14	Eraslan, S., Kopec-Harding, K., Jay, C., Embury, S. M., Haines, R., Cortés Ríos, J. C., & Crowther, P. (2020)	Adjacent support evidence	Used to refine careful comparisons with non-GitHub workflows or adjacent repository practices when direct classroom comparison evidence was limited.
15	Feliciano, J., Storey, M.-A. D., & Zagalsky, A. (2016)	Direct classroom evidence	Used in the main thematic synthesis for classroom uses, reported outcomes, and recurring adoption challenges because it provided direct teaching or learner evidence.
16	Fernandez, J. R. E., Santillan, C. G., & Puntos, L. L. (2025)	Context	Used to interpret institutional readiness, infrastructure, and not used alone to claim platform effects.
17	Glassy, L. (2006)	Foundational basis	Used to define social coding, visibility, collaboration, and version-control pedagogy concepts that anchored interpretation across later classroom studies.
18	Guerrero-Higueras, Á. M., Fernández Llamas, C., Sánchez González, L., Gutiérrez Fernández, A., Esteban Costales, G., & Conde González, M. Á. (2020)	Feature / analytics evidence	Used to explain how specific features such as pull requests, dashboards, analytics, automation, or trace data shaped workflow and feedback claims.
19	Haaranen, L., & Lehtinen, T. (2015)	Foundational basis	Used to define social coding, visibility, collaboration, and version-control pedagogy concepts that anchored interpretation across later classroom studies.
20	Hata, H., Novielli, N., Baltes, S., Kula, R. G., & Treude, C. (2022)	Feature / analytics evidence	Used to explain how specific features such as pull requests, dashboards, analytics, automation, or trace data shaped workflow and feedback claims.
21	Hecht, R., Liu, R., Zenke, C., & Malan, D. J. (2023)	Direct classroom evidence	Used in the main thematic synthesis for classroom uses, reported outcomes, and recurring adoption challenges because it provided direct teaching or learner evidence.

No.	Study	Evidentiary role	Synthesis use in the review
22	Heckman, S., & King, J. (2018)	Adjacent support evidence	Used to refine careful comparisons with non-GitHub workflows or adjacent repository practices when direct classroom comparison evidence was limited.
23	Hsing, C., & Gennarelli, V. (2019)	Direct classroom evidence	Used in the main thematic synthesis for classroom uses, reported outcomes, and recurring adoption challenges because it provided direct teaching or learner evidence.
24	Lawrance, J., Jung, S., & Wiseman, C. (2013)	Foundational basis	Used to define social coding, visibility, collaboration, and version-control pedagogy concepts that anchored interpretation across later classroom studies.
25	Milentijevic, I., Ciric, V., & Vojinovic, O. (2008)	Foundational basis	Used to define social coding, visibility, collaboration, and version-control pedagogy concepts that anchored interpretation across later classroom studies.
26	Olipas, C. N. P. (2022)	Context	Used to interpret institutional readiness, infrastructure, and not used alone to claim platform effects.
27	Olipas, C. N. P., Leona, R. F., Villegas, A. C. A., Cunanan, A. I., Jr., & Javate, C. L. P. (2021)	Context	Used to interpret institutional readiness, infrastructure, and not used alone to claim platform effects.
28	Patani, P., Tiwari, S., & Rathore, S. S. (2024)	Direct classroom evidence	Used in the main thematic synthesis for classroom uses, reported outcomes, and recurring adoption challenges because it provided direct teaching or learner evidence.
29	Santos, I., Felizardo, K. R., Sarma, A., Steinmacher, I., & Gerosa, M. A. (2025)	Adjacent support evidence	Used to refine careful comparisons with non-GitHub workflows or adjacent repository practices when direct classroom comparison evidence was limited.
30	Schauer, L., Stewart, R. J., & Maarek, M. (2024)	Adjacent support evidence	Used to refine careful comparisons with non-GitHub workflows or adjacent repository practices when direct classroom comparison evidence was limited.
31	Snowberger, A. D., & Lee, C. H. (2023)	Direct classroom evidence	Used in the main thematic synthesis for classroom uses, reported outcomes, and recurring adoption challenges because it provided direct teaching or learner evidence.
32	Tu, Y.-C., Terragni, V., Tempero, E., Shakil, A., Meads, A., Giacaman, N., Fowler, A., & Blincoe, K. (2022)	Direct classroom evidence	Used in the main thematic synthesis for classroom uses, reported outcomes, and recurring adoption challenges because it provided direct teaching or learner evidence.
33	Tushev, M., Williams, G., & Mahmoud, A. (2020)	Adjacent support evidence	Used to refine careful comparisons with non-GitHub workflows or adjacent repository practices when direct classroom comparison evidence was limited.
34	Wagner, G., & Thurner, L. (2025)	Adjacent support evidence	Used to refine careful comparisons with non-GitHub workflows or adjacent repository practices when direct classroom comparison evidence was limited.

No.	Study	Evidentiary role	Synthesis use in the review
35	Wessel, M., Vargovich, J., Gerosa, M. A., & Treude, C. (2023)	Feature / analytics evidence	Used to explain how specific features such as pull requests, dashboards, analytics, automation, or trace data shaped workflow and feedback claims.
36	World Bank. (2022)	Context	Used to interpret institutional readiness, infrastructure, and not used alone to claim platform effects.
37	Yu, Y., Wang, H., Yin, G., & Wang, T. (2016)	Feature / analytics evidence	Used to explain how specific features such as pull requests, dashboards, analytics, automation, or trace data shaped workflow and feedback claims.
38	Zagalsky, A., Feliciano, J., Storey, M.-A. D., Zhao, Y., & Wang, W. (2015)	Foundational basis	Used to define social coding, visibility, collaboration, and version-control pedagogy concepts that anchored interpretation across later classroom studies.

Table 1. Summary matrix of the 38 sources used in the analysis (source-role audit in Appendix B)

Workflow	Strengths	Weak points	What this review can support	Best fit
Traditional file submission	Simple for novices; low setup burden	Limited process visibility; feedback is often detached from the code artifact	Usually treated as the baseline that repository-based workflows improve on; better for early or low-support contexts	Small or early courses with limited technical support
GitHub / GitHub Classroom	Strong traceability, review support, templates, autograding, and authentic workflow exposure	Learning curve, setup friction, and uneven prior knowledge	Has the strongest and most direct classroom evidence in the reviewed literature	Project-based and team-based courses
GitLab-based workflow	Strong merge-request workflow, metrics, and integration options	Less classroom-specific literature than GitHub	Supported mainly by adjacent or smaller-scale education evidence; direct classroom comparisons remain limited	Courses emphasizing dashboards, analytics, or DevOps-style practice
Bitbucket and Pipelines	Integrated CI/CD and repository workflow	Very limited education-specific evidence in the reviewed set	Appears more often as a plausible workflow alternative than as a deeply studied classroom platform	Teams already using Atlassian tools

Table 2. Cross-workflow view of classroom options discussed in the literature

Research Gaps and Future Directions

The literature shows four main research gaps. The first is methodological strength. Many studies are useful case studies, but the field still needs more mixed-method and long-term research. We know a lot about how students and teachers respond to GitHub after one course, but less about whether these benefits last, transfer to later courses, or affect employability. Stronger comparative studies would help make the evidence more conclusive.

The second gap is feature specificity. Many studies treat GitHub as a single tool, but its features may affect learning in different ways. These include GitHub Classroom, pull requests, dashboards, and Actions. Future studies should examine these features separately, especially now that automation, autograding, and guided contribution workflows are being studied more directly (Hecht et al., 2023; Bennedsen et al., 2022; Santos et al., 2025).

The third gap is geographic and institutional diversity. A large part of the accessible literature comes from North America, Europe, and a smaller number of well-resourced institutions elsewhere. That leaves important questions about adoption in contexts with weaker infrastructure, larger skill gaps, or different curriculum conditions. This matters for institutions that are still at an early stage of tool adoption. In such settings, the main issue may not be optimization of an existing GitHub workflow, but whether and how GitHub should be introduced at all. Some contextual studies also suggest that onboarding, instructional support, coding confidence, and wider infrastructure conditions may shape adoption and student experience, but these sources should be read as background context rather than as direct evidence of GitHub's classroom effects (Fernandez et al., 2025; Bringula et al., 2020; Alimboyong & Bucjan, 2021; World Bank, 2022; Olipas et al., 2021; Olipas, 2022).

The fourth gap is direct comparison. Few studies directly compare GitHub with other platforms. Few also compare GitHub-based teaching with well-designed teaching that does not use GitHub. More studies are needed to compare platforms, workflow designs, and guided versus less guided use. For now, the safest conclusion is simple: GitHub can support good teaching, but its success depends on course design, context, and student readiness.

Implications for Teaching and Curriculum

For teachers, the practical lesson is that GitHub works best when it is introduced as part of instructional design rather than as an extra requirement. Courses should begin with a low-risk onboarding task, clear repository conventions, and short explanations of why each workflow step matters. Feedback should be tied to pull requests or concrete code locations. If autograding is used, students should be told what the tests can and cannot measure. These choices reduce confusion and help students see GitHub as support rather than punishment (Tu et al., 2022; Blair, 2026; Hecht et al., 2023).

For curriculum planners, the literature suggests that GitHub is most valuable in project-based, team-based, or iterative courses. It may be less useful in very early courses unless strong scaffolding is available. A staged curriculum may therefore be more effective than a one-course experiment. Clifton et al. (2007), Lawrance et al. (2013), Haaranen and Lehtinen (2015), Beckman et al. (2021), and Wagner and Thurner (2025) all support the idea that version-control practice should be sequenced rather than dropped on novices all at once.

For researchers, the clearest next step is to connect process traces with richer learning evidence. Repository logs, pull-request comments, and workflow results are useful, but they need to be paired with rubrics, interviews, or performance tasks. Guerrero-Higueras et al. (2020), Wessel et al. (2023), and Canale et al. (2024) all reinforce that process metrics can assist interpretation, but they should not be treated as self-sufficient evidence of learning.

Limitations

This review has three main limitations. First, it used a structured multi-source review chosen for a heterogeneous evidence base. That design provides a transparent and checkable account of searching, screening, coding, light appraisal, and synthesis, but it does not claim the fuller procedural features of a registered systematic review. The review used a light appraisal based on clarity of educational setting, transparency of method or implementation, traceability of claims, and acknowledgement of limitations to guide interpretive weighting rather than a formal risk-of-bias scoring system, so conclusions should be read as cautious synthesis rather than pooled causal evidence. Second, the comparative evidence is uneven across platforms. GitHub has a richer body of classroom work than GitLab or Bitbucket, so cross-platform conclusions remain necessarily cautious. Third, the search favored accessible English-language and openly reachable sources, which may under-represent classroom work reported only in local venues, paywalled proceedings, or other languages.

These limitations affect the strength of some claims, not the usefulness of the review. Broad patterns such as the importance of collaboration, workflow visibility, feedback placement, and onboarding support appear consistently enough to inform teaching and curriculum reflection. More specific claims about platform superiority, transferability across contexts, or feature-level advantage, however, should still be treated with caution until more direct comparative classroom studies and stronger longitudinal evidence are available.

Conclusion and Recommendations

The review contributes in three ways: topical, methodological, and evidentiary. It brings together evidence strands that are usually discussed separately, weights them by evidentiary role, and shows where stronger classroom support ends and more tentative comparison begins. It also makes the audit trail, source roles, and evidentiary boundaries more visible so that readers can judge the strength of the synthesis more directly.

The literature reviewed in this paper suggests that GitHub can be a useful environment in software engineering education because it brings student work, feedback, teamwork, and workflow history into one visible space. Its clearest educational value lies in making process inspectable: who changed what, when work moved forward, how feedback was exchanged, and how teams coordinated their tasks.

At the same time, the review does not support a simple claim that GitHub is automatically better than every other workflow. Its value depends on course design, student readiness, and the way teachers connect platform features to learning tasks. Where teachers provide onboarding, staged practice, and feedback-rich use of pull requests or automation, GitHub appears especially useful. Where those supports are absent, the platform can become an added burden.

Comparative observations are clearest when GitHub is discussed against basic file submission and much weaker when the discussion shifts to other collaborative platforms, where direct classroom studies remain scarce. For that reason, the main value of this review is not a platform ranking. It is a clearer map of what is currently known, where the stronger evidence lies, what remains adjacent rather than directly comparative, and what questions still need direct comparative study before stronger claims can be made.

Acknowledgement

The author acknowledges the researchers and educators whose published work informed this review.

Funding

No external funding was received for this study.

Competing Interests Statement

The author declares no competing financial or non-financial interests related to this review.

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study; all data used were obtained from previously published sources as cited in the reference list.

References

Adam, E. (2024). An Investigation Into the Perceived Effectiveness of GitHub Repositories to Teach Programming. International Conference on Education Research, 1(1), 1–9. https://doi.org/10.34190/icer.1.1.2774

Alimboyong, C. R., & Bucjan, M. E. (2021). Cloud computing adoption among state universities and colleges in the Philippines: Issues and challenges. International Journal of Evaluation and Research in Education, 10(4), 1455–1461. https://doi.org/10.11591/ijere.v10i4.21526

Beckman, M. D., Çetinkaya-Rundel, M., Horton, N. J., Rundel, C. W., Sullivan, A. J., & Tackett, M. (2021). Implementing version control with Git and GitHub as a learning objective in statistics and data science courses. Journal of Statistics and Data Science Education, 29(S1), S132–S144. https://doi.org/10.1080/10691898.2020.1848485

Bennedsen, J., Böjttjer, T., & Tola, D. (2022). Using GitHub Classroom in Teaching Programming. In Proceedings of the 18th International CDIO Conference (pp. 690–702). https://www.cdio.org/knowledge-library/documents/using-github-classroom-teaching-programming

Blair, S. (2026). GitHub Classroom: Student Attainment, Feedback, Feedforward and Industrial Practice. Compass: Journal of Learning and Teaching in Higher Education, 18(2). https://doi.org/10.21100/compass.v18i2.1648

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. Journal of Systems and Software, 80(4), 571–583. https://doi.org/10.1016/j.jss.2006.07.009

Bringula, R. P., Geronimo, S., & Aviles, A. (2020). Programming project in an undergraduate software engineering in the new normal: Challenges and proposed solutions. In SEED/NLPaSE@APSEC 2020 (pp. 29–37). https://ceur-ws.org/Vol-2799/Paper4_SEED.pdf

Calatrava Arroyo, A., Ramos Montes, M., & Segrelles Quilis, J. D. (2021). A Pilot Experience with Software Programming Environments as a Service for Teaching Activities. Applied Sciences, 11(1), 341. https://doi.org/10.3390/app11010341

Canale, L., Cagliero, L., Farinetti, L., & Torchiano, M. (2024). On Predicting Exam Performance Using Version Control Systems' Features. Computers, 13(6), 150. https://doi.org/10.3390/computers13060150

Clifton, C., Kaczmarczyk, L. C., & Mrozek, M. (2007). Subverting the fundamentals sequence: Using version control to enhance course management. SIGCSE Bulletin, 39(1), 86–90. https://doi.org/10.1145/1227310.1227344

Cochez, M., Isomöttönen, V., Tirronen, V., & Itkonen, J. (2013). How do computer science students use distributed version control systems? In ICTERI 2013 revised selected papers (pp. 210–228). https://doi.org/10.1007/978-3-319-03998-5_11

Cortés Ríos, J. C., Kopec-Harding, K., Eraslan, S., Page, C., Haines, R., Jay, C., & Embury, S. M. (2019). A Methodology for Using GitLab for Software Engineering Learning Analytics. In Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering (pp. 3–6). https://doi.org/10.1109/CHASE.2019.00009

Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012). Social coding in GitHub: Transparency and collaboration in an open software repository. Proceedings of CSCW 2012. https://doi.org/10.1145/2145204.2145396

Eraslan, S., Cortés Ríos, J. C., Kopec-Harding, K., Embury, S. M., Jay, C., Page, C., & Haines, R. (2020). Errors and Poor Practices of Software Engineering Students in Using Git. In Proceedings of the 4th Conference on Computing Education Practice (pp. 1–4). https://doi.org/10.1145/3372356.3372364

Eraslan, S., Kopec-Harding, K., Jay, C., Embury, S. M., Haines, R., Cortés Ríos, J. C., & Crowther, P. (2020). Integrating GitLab Metrics into Coursework Consultation Sessions in a Software Engineering Course. Journal of Systems and Software, 167, 110613. https://doi.org/10.1016/j.jss.2020.110613

Feliciano, J., Storey, M.-A. D., & Zagalsky, A. (2016). Student experiences using GitHub in software engineering courses: A case study. Proceedings of ICSE Companion 2016. https://doi.org/10.1145/2889160.2889195

Fernandez, J. R. E., Santillan, C. G., & Puntos, L. L. (2025). GitHub feature use and programming skill development among BSICT students. International Journal of Advanced Research in Science, Communication and Technology, 5(9), 422–432. https://doi.org/10.48175/IJARSCT-28254

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

Guerrero-Higueras, Á. M., Fernández Llamas, C., Sánchez González, L., Gutiérrez Fernández, A., Esteban Costales, G., & Conde González, M. Á. (2020). Academic success assessment through version control systems. Applied Sciences, 10(4), 1492. https://doi.org/10.3390/app10041492

Haaranen, L., & Lehtinen, T. (2015). Teaching Git on the side: Version control system as a course platform. Proceedings of ITiCSE 2015. https://doi.org/10.1145/2729094.2742608

Hata, H., Novielli, N., Baltes, S., Kula, R. G., & Treude, C. (2022). GitHub Discussions: An exploratory study of early adoption. Empirical Software Engineering, 27, 3. https://doi.org/10.1007/s10664-021-10058-6

Hecht, R., Liu, R., Zenke, C., & Malan, D. J. (2023). Distributing, Collecting, and Autograding Assignments with GitHub Classroom. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2 (p. 1179). https://doi.org/10.1145/3545947.3569627

Heckman, S., & King, J. (2018). Developing Software Engineering Skills Using Real Tools for Automated Grading. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (pp. 794–799). https://doi.org/10.1145/3159450.3159595

Hsing, C., & Gennarelli, V. (2019). Using GitHub in the classroom predicts student learning outcomes and classroom experiences: Findings from a survey of students and teachers. Proceedings of SIGCSE 2019. https://doi.org/10.1145/3287324.3287460

Kitchenham, B. (2004). Procedures for performing systematic reviews. Keele University Technical Report TR/SE-0401. https://www.inf.ufsc.br/~aldo.vw/kitchenham.pdf

Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering (EBSE Technical Report EBSE-2007-01). https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf

Lawrance, J., Jung, S., & Wiseman, C. (2013). Git on the cloud in the classroom. Proceedings of SIGCSE 2013. https://doi.org/10.1145/2445196.2445386

Milentijevic, I., Ciric, V., & Vojinovic, O. (2008). Version control in project-based learning. Computers & Education, 50(4), 1331–1338. https://doi.org/10.1016/j.compedu.2006.12.010

Olipas, C. N. P. (2022). A phenomenological study on the feelings, challenges and difficulties experienced by information technology students in learning computer programming. Path of Science, 8(7), 2001–2007. https://doi.org/10.22178/pos.83-3

Olipas, C. N. P., Leona, R. F., Villegas, A. C. A., Cunanan, A. I., Jr., & Javate, C. L. P. (2021). The academic performance and the computer programming anxiety of BSIT students: A basis for instructional strategy improvement. International Journal of Advanced Engineering, Management and Science, 7(6), 125–129. https://doi.org/10.22161/ijaems.76.15

Patani, P., Tiwari, S., & Rathore, S. S. (2024). The impact of GitHub on students' learning and engagement in a software engineering course. Computer Applications in Engineering Education. https://doi.org/10.1002/cae.22775

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology, 64, 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

Santos, I., Felizardo, K. R., Sarma, A., Steinmacher, I., & Gerosa, M. A. (2025). OSSDoorway: A Gamified Environment to Scaffold Student Contributions to Open Source Software. In Proceedings of CSEE&T 2025 (pp. 325–335). https://doi.org/10.1109/CSEET66350.2025.00037

Schauer, L., Stewart, R. J., & Maarek, M. (2024). Integrating Canvas and GitLab to Enrich Learning Processes. In Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training (pp. 180–190). https://doi.org/10.1145/3639474.3640056

Snowberger, A. D., & Lee, C. H. (2023). A workflow for practical programming class management using GitHub Pages and GitHub Classroom. Journal of Practical Engineering Education, 15(2), 331–339. https://doi.org/10.14702/JPEE.2023.331

Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. Journal of Business Research, 104, 333–339. https://doi.org/10.1016/j.jbusres.2019.07.039

Tu, Y.-C., Terragni, V., Tempero, E., Shakil, A., Meads, A., Giacaman, N., Fowler, A., & Blincoe, K. (2022). GitHub in the classroom: Lessons learnt. Australasian Computing Education Conference. https://doi.org/10.1145/3511861.3511879

Tushev, M., Williams, G., & Mahmoud, A. (2020). Using GitHub in large software engineering classes: An exploratory case study. Computer Science Education, 30(2), 155–186. https://doi.org/10.1080/08993408.2019.1696168

Wagner, G., & Thurner, L. (2025). Teaching tip: Rethinking how we teach Git: Pedagogical recommendations and practical strategies for the information systems curriculum. Journal of Information Systems Education, 36(1), 1–12. https://doi.org/10.62273/BTKM5634

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. MIS Quarterly, 26(2), xiii–xxiii. https://aisel.aisnet.org/misq/vol26/iss2/3/

Wessel, M., Vargovich, J., Gerosa, M. A., & Treude, C. (2023). GitHub Actions: The impact on the pull request process. Empirical Software Engineering, 28, 131. https://doi.org/10.1007/s10664-023-10369-w

World Bank. (2022). Digital transformation of Philippine higher education. World Bank. https://documents1.worldbank.org/curated/en/099925001062333685/pdf/P17757402843a10c90b3e30308406a38304.pdf

Yu, Y., Wang, H., Yin, G., & Wang, T. (2016). Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment? Information and Software Technology, 74, 204–218. https://doi.org/10.1016/j.infsof.2016.01.004

Zagalsky, A., Feliciano, J., Storey, M.-A. D., Zhao, Y., & Wang, W. (2015). The emergence of GitHub as a collaborative platform for education. Proceedings of CSCW 2015. https://doi.org/10.1145/2675133.2675284

Appendices

No appendices are attached to this study.