ISSN: 2473-4901
v11 n6321

Teaching Case:

Code Together, Apart: Introducing Students to Asynchronous Team Development Workflows Using GitHub

Kareem Dana
kdana@wtamu.edu

Abraham Abby Sen
aabbysen@wtamu.edu

West Texas A&M University
Canyon, TX 79016, USA

Jeen Mariam Joy
joyj2@vcu.edu
Virginia Commonwealth University
Richmond, VA 23222, USA

Hook

When technology and industry expectations evolve faster than your syllabus, what do you do? Are you tired of student drama in group projects? With this teaching case, we present a novel group assignment that minimizes frustration and better prepares students for their IS careers.

Abstract

Software development workflows in the information systems industry change quickly and are often different than what students experience in the classroom. This gap leaves many students underprepared for the engineering, analysis, or management jobs they seek. Asynchronous collaboration and tools like GitHub are increasingly prevalent in development teams. Traditional group projects, while beneficial, present persistent challenges, especially in online courses. This teaching case offers a practical assignment that bridges the classroom-industry gap and improves group work experience. We drew on established pedagogical foundations to design an assignment that emphasizes cooperation—where students work independently toward a shared outcome—over collaboration, introduces students to real-world developer workflows with GitHub, and uses a narrative story to improve engagement. Students complete individual yet interdependent tasks within an existing codebase through GitHub. We share evidence showing that our assignment increased student confidence, self-efficacy, and satisfaction, particularly compared to traditional group projects. We conclude with several practical recommendations instructors can use to reduce group work frustration and build job-ready skills.

Keywords: Asynchronous learning, Git, Group Projects, Pedagogy, Collaboration, Cooperative learning

ISSN: 2473-4901
v11 n6321

Code Together, Apart: Introducing Students to Asynchronous Team Development Workflows Using GitHub

Kareem Dana, Abraham Abby Sen, Jeen Mariam Joy

1. INTRODUCTION

More and more software development occurs remotely through geographically distributed teams collaborating asynchronously. This increase is driven in part by the rise of remote and hybrid work and the desire by companies to hire the best talent regardless of location (Kumar et al., 2024; Sharma, 2023). These teams are increasingly using collaborative tools to manage their work and improve productivity. Among these tools, Git and GitHub have become industry standards (Dabbish et al., 2012; Hu et al., 2016).

Git is a widely used version control system, while GitHub is a web-based platform built on top of Git that hosts code repositories. GitHub has features such as issue tracking, pull requests, and code reviews that naturally support asynchronous, collaborative workflows needed by distributed software teams. As of 2023, GitHub hosts over 420 million repositories and is used by over 100 million developers worldwide, including many large corporations such as Microsoft, Google, and IBM (Dohmke, 2023).

This widespread use of GitHub and the uptick in asynchronous software development suggests that mastery of this tool and workflow is now a core expectation for new graduates entering the workforce. This expectation is not just for computer scientists or software developers. Experience with GitHub and asynchronous collaboration are growing in importance across a variety of Information Systems (IS) jobs. Product owners, managers, and systems analysts all use GitHub to track issues, requirements, and communicate with their teams (Wagner & Thurner, 2025).

Additional research indicates that IS students are frequently unfamiliar with the workflows and tools that they will encounter in the workplace, leaving them underprepared to start their careers (Craig et al., 2018; Liebenberg et al., 2015; Moreno et al., 2012). Consequently, as IS educators, we are tasked with this challenge. Traditional classroom settings, or online courses relying solely on discussion forums, may not sufficiently prepare students for their careers.

On the other hand, group projects can prepare students well for collaboration and are frequently utilized in IS classes, including online. However, implementing effective group projects remains difficult, especially asynchronously (Hafner & Ellis, 2004). They are often met by student frustration. Students highlight issues such as unequal participation, communication challenges, and a lack of accountability. These frustrations are often exacerbated in asynchronous environments (Awuor et al., 2022; Bakir et al., 2020, Roberts & McInnerney, 2007).

In this paper, we present a teaching case that addresses both challenges. This learning activity simulates real-world development workflows while introducing students to asynchronous collaboration using GitHub. It further follows best practices to avoid many group work pitfalls.

Our main contributions are:

1. The description and development of the learning activity, backed by a strong pedagogical foundation outlined in sections two and three.
2. Evidence shown in section four, showcasing the assignment's efficacy through statistical analysis of pre- and post-assignment survey results along with qualitative feedback from students.
3. Practical recommendations, explained in section five, for IS educators to implement and adapt to different objectives and instructional contexts.

2. MOTIVATION AND BACKGROUND

We taught this assignment in our IS department's second programming course for undergraduates. We developed it after discussions with our capstone instructor that centered around the question, "How can we introduce advanced IS topics used in the capstone course earlier in the curriculum?" This led to further discussions about student preparedness for real-world projects and challenges with group work.

Preparing Students for IS Workflows

Past research highlights a well-acknowledged gap between industry expectations and graduating student skill levels (Aasheim et al., 2012; Craig et al., 2018; Moreno et al., 2012; Oguz & Oguz, 2019; Tuzun et al., 2018). Craig (2018) states that soon after graduation, students notice real life projects are of a different breed from the ones they have handled during their education. School assignments usually involve small stand-alone programs, while companies rely on large, existing codebases with a variety of dependencies and libraries. Companies often use tools such as Git and GitHub to manage those codebases. To address this skills gap, we drew on prior literature and pedagogical theories.

First, we provide students with an existing codebase, written by the instructor, to simulate the large codebases seen in IS jobs. Second, to help students avoid feeling overwhelmed by unfamiliar code, we require them to follow a structured GitHub workflow that includes creating issues, branching, merging, and resolving merge conflicts. This workflow guides their initial interactions with the codebase through clearly defined, step-by-step tasks allowing them to build familiarity and gain confidence as they progress. This design builds on the work Kertész (2015), Luce (2021), and Glassey (2019), who all emphasize the pedagogical value of incorporating GitHub in the classroom. In addition, Agrawal (2024) and Xia (2018) highlight the importance of introducing students to unfamiliar codebases.

Lastly, our assignment provides a platform to introduce advanced topics and workflows earlier in the curriculum at lower levels (remember and understand) of Bloom's taxonomy (Anderson, et al., 2001), with reinforcement occurring in later courses. This design also supports the introduce-reinforce-emphasize curriculum model advocated by the accreditation board, ABET, where topics are introduced in foundational classes, reinforced in later classes, and emphasized or mastered in a capstone class (Almuhaideb & Saeed, 2020; Calderón, et al., 2016). In our curriculum, these topics include C# class libraries and unit testing, both of which are part of that skills gap (Craig et al., 2018).

To help introduce students to these topics, we designed the assignment to follow a task-based pedagogical model, wherein students complete well-defined tasks with step-by-step instructions. This is a beneficial way to successfully introduce new, challenging, or advanced topics without cognitively overloading students (Kulesza et al., 2011; Warrick, 2021).

Another key innovation of our approach was adding a narrative-based story. Narrative pedagogy uses storytelling for learning complex topics and has been shown to increase learning and improve engagement (Humpherys & Babb, 2020; Lee et al., 2006).

Group Work Challenges

Group projects are another excellent pedagogical tool to prepare students for real-world projects and are used often in IS classes. However, prior research suggests that many students dislike group work (Lowes, 2014). Roberts & McInnerney (2007) and Ekblaw (2016) highlight seven major challenges students face with group work, while Bakir, et al. (2020) discusses group work challenges within a Management Information Systems course.

Students often express apathy towards group work. They may not be motivated or do not understand the benefits (Ekblaw, 2016; Roberts & McInnerney, 2007). Students are most motivated by their grade. Fairly assessing individual performance can significantly reduce this challenge (Favor & Harvey, 2016; Roberts & McInnerney, 2007). We accomplish that by tracking student contributions through GitHub commit logs and ultimately grading each student individually based solely on their contributions.

Lack of communication is another major challenge in group work. To solve that challenge, we structured our assignment so that students work on their parts asynchronously and independently. This technique is recommended by Lowes (2014) and Ekblaw (2016). We further make a distinction between collaboration and cooperation. Cooperation is where individuals are working towards a shared goal, but each works independently. Collaboration involves interactions among all group members to achieve the goal (Bruffee, 1995; Panitz, 1999; Paulus, 2005). In our assignment, students cooperate but do not need to collaborate.

We are not suggesting that collaboration is unimportant. On the contrary, collaboration and communication skills are important. This teaching case, simply, offers an alternative group work technique that can be useful.

Another group challenge is a lack of accountability or "free riding". That is when one group member slacks off or does not perform. We approach this problem with the same techniques as described above. Cooperation, as defined above, and fairly assessing individual performance through GitHub commit logs mitigates free riding.

Finally, students complain about poor distribution of responsibilities. Our assignment clearly defines which group member will do which task ahead of time to eliminate this common problem.

3. THE ASSIGNMENT

In this section, we describe how we developed the assignment, provide the steps needed for instructors to deploy it in their classroom, outline how a typical student would complete the assignment, and share grading guidelines.

Instructor Guidelines

Our programming course is taught online and asynchronously at a regional business school in the southwestern United States. All students are computer information systems (CIS) majors. Most are sophomores or juniors and have already taken an introductory programming course. They have basic C# skills and experience with Visual Studio (VS) Code. While students also had one semester of experience with git and GitHub, no prior experience is required.

First, to build a narrative story, we created ThoughtTronix, a morally ambiguous technology company that develops artificial intelligence devices. The story of ThoughtTronix starts early in the semester and is woven throughout the semester to keep students interested, engaged, and motivated. As described earlier, this is supported by narrative pedagogy theory (Humpherys & Babb, 2020; Lee et al., 2006).

Through successive assignments, students take on the role of junior developers contributing to ThoughtTronix's ethically questionable products, such as MindSync, a brain implant, and SoulSear, an AI weapon. In this assignment, students are required to add features to an existing web application that processes orders for those products. The features are to calculate (1) sales tax and (2) shipping charges. Most students are familiar with ordering products online, so these features will feel familiar to them. This also mimics real-world development where many software developers fix bugs, add features, or maintain existing applications instead of creating new ones (Craig et al., 2018).

Second, to support this existing codebase we created a template repository in GitHub that contains the starter code – a C# ASP.NET Core web application for order processing with the sale tax and shipping cost features not implemented. A template repository allows the instructor to create new repositories for each student team with the same structure and starter code.
We wrote the starter code in a deliberate manner such that students can complete their parts independently and then merge them to form the complete, working application. We chose to separate each feature into a class library and have students implement their class library. This gave us the benefit of introducing students to class libraries, which are used in our capstone course.

Third, we placed students in teams of two chosen randomly and created a GitHub repository for each team. The instructor was the repository administrator and added each student as a collaborator.

Finally, the instructor emailed both students. In the email, we assigned each their feature – either to calculate sales tax or calculate shipping charges. Both tasks achieve the same learning objectives and can be completed in any order. The email also included links to the GitHub repository, assignment instructions, and the grading rubric - all the details needed for students to start working on the assignment.

Learning Objectives

The assignment has five learning objectives, designed to align with industry skills, particularly those associated with the skills gap discussed earlier:

1. Gain experience in asynchronous software development teams. This is the main objective of the assignment. Students gain hands-on experience with GitHub in a team setting, reflecting the distributed collaboration workflows commonly used in the IS industry.
2. Create Git branches. Students create branches to facilitate development of their software features, reflecting common industry practices for managing parallel work and isolating code changes.
3. Implement class libraries. Students are introduced to class libraries, create their own class library, and connect it to the existing code. This models the modular, reusable code design expected in professional software development. This objective can be replaced by another as needed by the instructor.
4. Run unit tests. Students were given existing unit tests and were required to test their code. This introduces students to test-driven development and automated testing practices that are widely used in real-world software development teams to ensure code reliability and maintainability. This learning objective introduces unit testing at the lower levels of Bloom's taxonomy

(remembering and applying). Reinforcement and emphasis occur in the capstone course. This is the other learning objective that can be replaced to match other instructors' goals.

5. Merge Git branches and resolve merge conflicts. Students learn how to merge their branches back into the main branch and cooperate to resolve merge conflicts, closely simulating version control challenges and code integration issues developers face in industry but rarely see in the classroom.

Student Experience
There is no prior lecture content. We designed the assignment to introduce students to each topic through a learn-by-doing approach (Reese, 2011). Students did not have any prior knowledge of GitHub issues, branching, class libraries, unit testing, or git merging. Instructors can supplement this assignment with additional content to fill in any gaps in theory if desired. However, our evidence suggests that students were successfully introduced to all these topics through this learning activity.

The students complete the assignment through the following steps:

Step 1: Create an issue in GitHub. An Issue is a way to track and manage tasks. It is a key collaborative feature in GitHub. Through this step, the student learns to clearly articulate what feature they are implementing and how to implement it. They are instructed to list clear steps on how they plan to complete the assignment. This helps students prepare and improves the chance of success.

Step 2: Create a branch. Students then create a Git branch tied to their issue. This introduces them to branching, a foundational part of most team-based workflows.

Step 3: Checkout their branch. After creating a branch, the student will clone the repository and checkout their branch. Students complete this step through VS Code or the command line with a "git checkout" command.

Step 4: Write the code. Students implement their assigned feature following the step-by-step instructions. While this may seem like the most important step, it primarily serves as a means for engaging the students in the broader learning objectives.

Step 5: Test their code. Students run pre-written unit tests to validate what they wrote in step 4. This introduces them to test-driven development and provides immediate feedback, allowing iterative improvement until all tests pass.

Step 6: Commit and push changes. Students commit their changes and push them to GitHub using VS Code or git at the terminal. We instruct students to visit GitHub to verify that their changes have been successfully pushed.

Step 7: Merge branches and close the issue. Students merge their branch back into the main branch and then close their issue from step one. This closes the loop, demonstrating the full development cycle from planning to delivery – albeit of a small feature.

Each student completes these steps independently. When both team members succeed, the application is functionally complete. This design fosters cooperative engagement without requiring direct collaboration, minimizing a common pitfall of traditional group projects.

Grading Guidelines
We developed a grading rubric for the assignment (see Appendix A), which was also shared with students in advance. After the due date, we reviewed each team's GitHub repository and assessed each student individually based on their contributions. GitHub's commit history provided an objective record, allowing us to verify what each student completed and when.

We, then provided detailed feedback through our university's learning management system with references to specific rubric items (e.g. "Missed Task 10") and, as needed, further explanation of why and how to correct mistakes.

4. RESULTS AND STUDENT FEEDBACK

We administered pre- and post-assignment surveys to students across two semesters: Fall 2024 (n = 24) and Spring 2025 (n = 19). The surveys asked students to rank their comfort or familiarity with the assignment's five learning objectives: GitHub issues, creating branches, merging branches, class libraries, and unit tests using a five-point Likert-scale (1 = not comfortable, 5 = very comfortable) along with one question about how prepared students feel for a job in a software development team. The post-survey added questions about students' satisfaction with the assignment and open-ended questions for qualitative feedback. See Appendix B for the complete survey questions. We identified three key themes in the survey results.

	Pre-Assignment Survey (N = 43)		Post-Assignment Survey (N = 43)				
How comfortable are you with the following:	Mean	SD	Mean	SD	*p*-Value	t(df)	Cohen's d
GitHub Issues	2.91	1.31	4.56	0.63	< .001	-8.93 (42)	1.21
Creating Branches	3.33	1.17	4.56	0.63	< .001	-7.93 (42)	1.02
Merging Branches	2.91	1.27	4.40	0.69	< .001	-7.73 (42)	1.26
Class Libraries	3.02	1.23	4.09	0.75	< .001	-6.65 (42)	1.06
Unit Tests	2.86	1.21	3.88	0.79	< .001	-5.02 (42)	1.34
Job Preparedness	2.72	1.05	3.70	0.99	< .001	-6.48 (42)	0.99

Table 1: Paired Sample T-Test Results

Improved Learning
We combined both semesters of survey data together and used a paired samples t-test measuring improvements between the pre- and post-assignment surveys for each of the five learning objectives. We observed statistically significant self-reported improvement in all learning objectives and present them in Table 1 below.

Students self-reported these results, but through grading the assignment, we confirm students demonstrated sufficient knowledge of the learning objectives.

Improved Job Preparedness
There was a statistically significant increase in self-reported job preparedness when asked "How prepared do you feel for a job in a software development team?" as shown on the final row of Table 1. Job preparedness is a multi-faceted challenge, and we recognize that a single assignment has limited impact. Nonetheless, the results suggest that students felt at least somewhat more prepared to work in a professional software development team. A valuable extension of this analysis would be a longitudinal study to track whether these perceptions persist over time.

Positive Assignment Satisfaction
On the post-assignment survey, students expressed increased satisfaction with this assignment compared to prior group work experiences. Of the forty-three respondents, 36 (84%) indicated a preference to asynchronous group work like this assignment, while 7 (16%) stated no preference. Zero students preferred their prior group work experiences. We recognize that we had no control over nor any familiarity with their prior group experiences.

Student Feedback
In addition, students left open-ended feedback expressing the same preference:

- "One of the benefits of this type of asynchronous group assignment was that there were no scheduling conflicts as we could finish our part of the assignment when we had the time. The instructions were clear and easy to follow as well."
- "Yes, not having to depend on others and getting graded individually."
- "Not being responsible for my teammates work."

Several students also noted the storytelling aspect of the assignment as a benefit:

- "I really enjoyed this class. The themed homework assignments with the ominous defence company and their products was a really nice touch and I enjoyed the narrative that it sort of wove through the homework. I learned a lot from this class!"
- "Teaching methods were clear, well-organized, and engaging, using interesting and funny examples to make complex concepts relatable and applications more enjoyable."

5. DISCUSSION AND TEACHING SUGGESTIONS

In this section we discuss challenges encountered, modifications instructors can make to the assignment, and practical teaching suggestions based on our experiences.

Challenges
We experienced some challenges with this assignment. It can be onerous to manually create GitHub repositories for all teams and invite students individually, especially in a large class. Moreover, GitHub invitations expire after seven days and several students forgot to accept the invitation, requiring the instructor to re-send it.
While we had developed the ThoughtTronix

narrative throughout the semester, it may be time-consuming or challenging for another instructor to use the same narrative or develop their own.

Students expressed their own challenges as well. Some stated learning the inner workings of GitHub was challenging. Several other students expressed confusion with merging and merge conflicts, as well as challenges related to software incompatibilities with git, .NET, VS Code, and the students' own personal computers.

Finally, during grading, we identified that several teams had one student who did not participate. However, the free rider did not negatively impact their teammate, and no student expressed frustration with free riding.

Modifications
A benefit of this assignment is that it can be modified in a myriad of ways. Below we outline a few of those ways:

Change Learning Objectives. Our course specific learning objectives were class libraries and unit tests. Those can be swapped out and still maintain the core benefits of the assignment. For example, advanced students could be asked to implement a full authentication-based library, while beginner students might implement small logic changes and both cohorts of students would benefit from utilizing GitHub and gain valuable experience following a professional workflow in an asynchronous setting.

Change Programming Language. This assignment is written in C# to fit the needs of our curriculum. It could be modified for another language while still utilizing GitHub, issues, branches, and asynchronous teamwork.

Change Group Size. This assignment is designed for groups of two. There are two features to implement, and each student implements one. We understand common groups tend to be 3 or 4 students. Supporting larger groups requires additional features to be brought into the assignment. Perhaps a third student could implement a rewards discount. Ultimately, changing the group size will require an additional feature for each extra student to implement.

Change Course. We believe this assignment can be adapted for an introductory programming course with only a few modifications. First, we recommend assigning it near the end of the term once students have learned basic programming skills. Second, instructors can add preparatory materials and simplify or change learning objectives 3 and 4 to better match their course goals.

The assignment can also be extended to non-programming courses or to students from non-technical majors. For example, students could create UML diagrams instead of class libraries. Instructors teaching non-technical students might emphasize issue tracking, GitHub collaboration, and workflow planning rather than programming. These adjustments maintain the assignment's structure and benefits while meeting students at their skill level.

Teaching Suggestions
Through developing this assignment, student feedback, and ongoing conversations with colleagues we present several teaching suggestions below. Whether you implement this assignment or not, these are practical strategies for any IS educator to utilize.

Teaching Suggestion 1: Offer an asynchronous assignment like this one as an alternative to traditional group work. While this assignment does not replace group projects with full collaboration, it can be a valuable tool and provide students with a positive experience. Our students reported, overwhelmingly, preferring this assignment over traditional groupwork.

We showed that this learning activity mitigates some of the major challenges students face with group work. Additionally, providing multiple means of engagement to cater to a diverse set of learners can help all students succeed (Gadsden & Goegan, 2023; Kolb, 2014; Saunders & Wong, 2023). This assignment offers another means of engagement through a hands-on alternative to lecture and a different group project dynamic. This suggestion highlights the importance of finding a balance between the benefits of group work and the frustrations.

Teaching Suggestion 2: In group projects, we recommend grading students individually based on their own contributions. Instructors should actively track these contributions to promote accountability and clearly communicate to students that individual efforts will be recognized.

This approach helps address common concerns about free riding. Even when a groupmate contributes little or no work, individual grading based on tracked contributions ensures that a diligent student can still earn full credit. Our students expressed high satisfaction with this approach, as one student remarked on the survey, "*I was able to independently get my portion of the work done in my own time and at my own pace which greatly reduced stress and frustration on my end. I also didn't feel like there was a power dynamic where one group member tries to take everything or the opposite, does nothing, I really liked this way of doing things.*"

Teaching Suggestion 3: Introduce advanced topics earlier in the curriculum. We introduced class libraries and unit testing earlier than usual, giving students exposure to key concepts before they encounter them in more advanced courses. Instructors can choose any topic, and we believe students will benefit from this early introduction. This strategy aligns with the pedagogical foundation of Bloom's taxonomy, which provides a framework for introducing concepts at lower cognitive levels and reinforcing them in later classes (Anderson et al., 2001). It is also consistent with ABET's introduce-reinforce-emphasize model (Almuhaideb & Saeed, 2020; Calderón, et al., 2016).

Teaching Suggestion 4: Use a narrative story in your assignments, especially when introducing complex topics. Research supports this approach, showing that narrative-based learning can be an effective instructional tool (Humpherys & Babb, 2020; Lee et al., 2006; Zazkis, 2009). Our findings support this, as our students demonstrated positive outcomes and shared feedback showing satisfaction with the narrative story. When possible, consider using a narrative approach throughout the course instead of for one assignment.

Teaching Suggestion 5: Utilize real-world tools to simulate professional development environments. Integrating tools like GitHub into assignments does more than just support asynchronous work; it exposes students to industry-standard workflows they are likely to encounter after graduation. We familiarized our students with core professional practices, such as using GitHub Issues to define tasks, creating feature branches, resolving merge conflicts, and modifying an existing codebase. Students reported feeling more prepared in their ability to contribute to a development team after completing this assignment.

This suggestion is consistent with two well-established pedagogical theories. The first, Experiential Learning Theory (Kolb, 2014), emphasizes learning as a process where knowledge is created through direct experience and reflection. According to Kolb, students learn most effectively when they engage in a full cycle of doing, reflecting, conceptualizing, and applying. The second, Learning-by-Doing (Reese, 2011), asserts that students gain deeper understanding when they are actively engaged in meaningful tasks, rather than passively receiving information. Reese highlights that active participation and direct engagement leads to better motivation and skill development when compared to methods like lectures. By immersing students in real-world development workflows, our assignment puts these principles into practice.

6. CONCLUSIONS

The purpose of this teaching case is to provide IS educators with a practical model for integrating real-world programming workflows into a unique, asynchronous group assignment, ensuring that students are better equipped for an IS career.

Our strategies of mitigating group work frustrations through cooperative (rather than collaborative) work and individual accountability, integrating a narrative story, and using real-world workflows are backed by strong pedagogical theories. By combining these elements into a cohesive, practical assignment, we believe we provide a model that not only engages students but also prepares them for the expectations of industry.

Evidence from student feedback shows increased confidence and self-efficacy with GitHub and associated developer workflows, as well as a strong preference for this type of group assignment. This case can also be adapted by instructors to fit a range of learning objectives and instructional contexts.

Finally, all the teaching materials including the assignment instructions, starter code, and rubric are available through a GitHub repository. Given their size and code-heavy nature, they are not included in the appendix. The solution is also available from the authors or the editor upon request. The GitHub repository link is: https://github.com/kareemy/CodeTogether_TeachingMaterials

7. REFERENCES

Aasheim, C., Shropshire, J., Li, L., & Kadlec, C. (2012). Knowledge and skill requirements for entry-level IT workers: A longitudinal study. *Journal of Information Systems Education*, *23*(2), 193-204.

Agrawal, E., Alam, O., Goenka, C., Iyer, M., Moise, I., Pandian, A., & Paul, B. (2024). Code compass: A study on the challenges of navigating unfamiliar codebases. *arXiv*. https://doi.org/10.48550/arXiv.2405.06271

Almuhaideb, A. M., & Saeed, S. (2020). Fostering sustainable quality assurance practices in outcome-based education: Lessons learned from ABET accreditation process of computing programs. *Sustainability*, 12(20), 8380. https://doi.org/10.3390/su12208380

Anderson, L. W., Krathwohl, D. R., & Bloom, B. S. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives: complete edition.* Addison Wesley Longman.

Awuor, N. O., Weng, C., & Militar, R. (2022). Teamwork competency and satisfaction in online group project-based engineering course: The cross-level moderating effect of collective efficacy and flipped instruction. *Computers & Education*, 176, 104357. https://doi.org/10.1016/j.compedu.2021.104357

Bakir, N., Humpherys, S., & Dana, K. (2020). Students' Perceptions of Challenges and Solutions to Face-to-Face and Online Group Work. *Information Systems Education Journal*, 18(5), 75-88.

Bruffee, K. A. (1995). Sharing our toys: Cooperative learning versus collaborative learning. *Change: The Magazine of Higher Learning*, *27*(1), 12-18. https://doi.org/10.1080/00091383.1995.9937722

Calderón, H. E., Vásquez, R. E., Aponte, D. A., & Del Valle, M. (2016). Successful Assessment Strategies for ABET Accreditation of Engineering Programs Offered at Different Campuses. In *Proceedings of the 14th LACCEI International Multi-Conference for Engineering, Education, and Technology: "Engineering Innovations for Global Sustainability"*, San José, Costa Rica (pp. 20-22). https://doi.org/10.18687/LACCEI2016.1.1.226

Craig, M., Conrad, P., Lynch, D., Lee, N., & Anthony, L. (2018). Listening to early career software developers. *Journal of Computing Sciences in Colleges*, 33(4), 138-149.

Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012). Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on computer supported cooperative work* (pp. 1277-1286). https://doi.org/10.1145/2145204.2145396

Dohmke, T. (2023). 100 Million Developers and Counting. https://github.blog/2023-01-25-100-million-developers-and-counting/

Ekblaw, R. (2016). Effective Use Of Group Projects In Online Learning. *Contemporary Issues in Education Research (CIER)*, 9(3), 121–128. https://doi.org/10.19030/cier.v9i3.9707

Favor, J. K., & Harvey, M. (2016). We Shall Not be Moved: Adult Learners' Intransigent Attitudes About Group Projects. *Adult Education Research Conference*. Retrieved June 6, 2025 from https://newprairiepress.org/aerc/2016/papers/18/

Gadsden, A. D., & Goegan, L. D. (2023). Informing Inclusive Practice in Post-Secondary Environments: Perspectives of Post-Secondary Instructors with Learning Disabilities. *The Canadian Journal for the Scholarship of Teaching and Learning*, 14(2). https://doi.org/10.5206/cjsotlrcacea.2023.2.8020

Glassey, R. (2019, May). Adopting git/github within teaching: A survey of tool support. In *Proceedings of the ACM Conference on Global Computing Education* (pp. 143-149). https://doi.org/10.1145/3300115.3309518

Hafner, W., & Ellis, T. J. (2004). Asynchronous collaborative learning using project-based assignments. In *34th Annual Frontiers in Education, 2004. FIE 2004.* (pp. F2F-6). IEEE. https://doi.org/10.1109/fie.2004.1408607

Hu, Y., Zhang, J., Bai, X., Yu, S., & Yang, Z. (2016). Influence analysis of GitHub repositories. *SpringerPlus*, 5, 1-19. https://doi.org/10.1186/s40064-016-2897-7

Humpherys, S. L., & Babb, J. (2020). Using Folklore, Fables, and Storytelling as a Pedagogical Tool in Assessment Exams. *Information Systems Education Journal*, 18(5), 34-53.

Kertész, C. Z. (2015). Using GitHub in the classroom-a collaborative learning experience. In *2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME)* (pp. 381-386). IEEE. https://doi.org/10.1109/siitme.2015.7342358

Kolb, D. A. (2014). Experiential learning: Experience as the source of learning and development. FT press.

Kulesza, J., DeHondt, G., & Nezlek, G. (2011). More Technology, Less Learning?. *Information Systems Education Journal*, *9*(7), 4-13.

Kumar, A., Priyadarshi, P., & Garg, N. (2024). Bibliometric Analysis of Remote Working: 20-year Literature Review. *Human Resource Development Review*. https://doi.org/10.1177/15344843241305920

Lee, J. H. M., Lee, F. L., & Lau, T. S. (2006). Folklore-based learning on the web—pedagogy, case study, and evaluation. *Journal of Educational Computing Research*, 34(1), 1-27. https://doi.org/10.2190/3hfm-d9nq-g7y7-qc1g

Liebenberg, J., Huisman, M., & Mentz, E. (2015). The relevance of software development education for students. *IEEE Transactions on Education*, 58(4), 242-248. https://doi.org/10.1109/TE.2014.2381599

Lowes, S. (2014). How Much "Group" Is There in Online Group Work? *Journal of Asynchronous Learning Networks*, 18(1), n1. https://doi.org/10.24059/olj.v18i1.373

Luce, T. (2021). Distributed Project Teams and Software Development: An Introduction to the Use of Git and GitHub for ASP. NET MVC Development. *Information Systems Education Journal*, *19*(5), 45-57.

Moreno, A. M., Sanchez-Segura, M. I., Medina-Dominguez, F., & Carvajal, L. (2012). Balancing software engineering education and industrial needs. *Journal of systems and software*, 85(7), 1607-1620. https://doi.org/10.1016/j.jss.2012.01.060

Oguz, D., & Oguz, K. (2019). Perspectives on the gap between the software industry and the software engineering education. *IEEE Access*, 7, 117527-117543. https://doi.org/10.1109/ACCESS.2019.2936660

Panitz, T. (1999). Collaborative versus cooperative learning: A comparison of the two concepts which will help us understand the underlying nature of interactive learning. Retrieved June 6, 2025 from https://files.eric.ed.gov/fulltext/ED448443.pdf

Paulus, T. M. (2005). Collaborative and cooperative approaches to online group work: The impact of task type. *Distance education*, 26(1), 111-125. https://doi.org/10.1080/01587910500081343

Reese, H. W. (2011). The Learning-by-Doing Principle. *Behavioral Development Bulletin*, 17(1), 1-19. https://doi.org/10.1037/H0100597

Roberts, T. S., & McInnerney, J. M. (2007). Seven problems of online group learning (and their solutions). *Educational Technology & Society*, 10(4), 257–268.

Saunders, L., & Wong, M. (2023). Multiple Means of Engagement: Connecting with Students Across Modalities through Choice, Flexibility and Authentic Assessment. In *Proceedings of the ALISE Annual Conference*. http://dx.doi.org/10.21900/j.alise.2023.1264

Sharma, T. K. (2023). Hybrid Working: The Future of Organizations. In Reshaping the Business World Post-COVID-19. Apple Academic Press.

Tuzun, E., Erdogmus, H., & Ozbilgin, I. G. (2018, May). Are computer science and engineering graduates ready for the software industry? Experiences from an industrial student training program. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training* (pp. 68-77). https://doi.org/10.1145/3183377.3185754

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

Warrick, A. (2021). Strategies for reducing cognitive overload in the online language learning classroom. *International Journal of Second and Foreign Language Education*, *1*(2), 25-37. https://doi.org/10.33422/ijsfle.v1i2.124

Xia, X., Bao, L., Lo, D., Xing, Z., Hassan, A. E., & Li, S. (2017). Measuring program comprehension: A large-scale field study with professionals. *IEEE Transactions on Software Engineering*, *44*(10), 951-976. https://doi.org/10.1109/tse.2017.2734091

Zazkis, R., & Liljedahl, P. (2019). *Teaching mathematics as storytelling*. Brill. https://doi.org/10.1163/9789087907358

ISSN: 2473-4901
v11 n6321

APPENDIX A
Grading Rubric

Below is the grading rubric used for this teaching case assignment. All the teaching materials including the assignment instructions, starter code, rubric, and solution are available from the authors or the editor upon request and at this GitHub link:
https://github.com/kareemy/CodeTogether_TeachingMaterials

Task	**Points**
Create an issue on GitHub	5
Assign the issue to yourself	5
Create a new branch on GitHub	10
Checkout your branch	5
Write all your code in your branch (Do not code directly in the main branch)	5
Create a new class library	5
Add your class library to the main solution file	5
Add a reference to your class library in OrderApp	5
Write your code in the class library	10
Uncomment the correct using directive in ReviewOrder.cshtml.cs	5
Uncomment the correct line of code in ReviewOrder.cshtml.cs	5
Test your code with dotnet test	5
Push your changes back to GitHub in the correct branch	5
Create a pull request on GitHub	10
Merge pull request and recognize "Can't automatically merge" message	10
Resolve merge conflicts if applicable	5
Total:	**100**

ISSN: 2473-4901
v11 n6321

APPENDIX B
Survey Questions

The purpose of this survey is to better understand your learning experiences with group projects in general and your understanding of software development project workflows.
Please take the time to answer these questions.
Thank You.

Pre-Assignment Survey Questions:

1. How comfortable are you with using the Issues feature on GitHub?

	1	2	3	4	5	
Not comfortable	○	○	○	○	○	Very comfortable

2. How comfortable are you with creating branches on GitHub?

	1	2	3	4	5	
Not comfortable	○	○	○	○	○	Very comfortable

3. How comfortable are you with merging code changes on GitHub?

	1	2	3	4	5	
Not comfortable	○	○	○	○	○	Very comfortable

4. How familiar are you with the computer programming concept of class libraries?

	1	2	3	4	5	
Not familiar	○	○	○	○	○	Very familiar

5. How familiar are you with the concept of unit tests?

	1	2	3	4	5	
Not familiar	○	○	○	○	○	Very familiar

6. How prepared do you feel for a job in a software development team?

	1	2	3	4	5	
Not prepared	○	○	○	○	○	Very prepared

ISSN: 2473-4901
v11 n6321

Post-Assignment Survey Questions:

1. As you respond, consider only prior online group experiences. How would you rate your overall experience with PAST online group projects?

	1	2	3	4	5	
Poor	○	○	○	○	○	Excellent

2. Describe the main challenges you faced with PAST online group projects. [Open-Ended]
3. Consider your CURRENT experience with this asynchronous group assignment. How would you rate your overall experience?

	1	2	3	4	5	
Poor	○	○	○	○	○	Excellent

4. Describe the main challenges you faced with THIS assignment. [Open-Ended]
5. Did this type of assignment have any benefits over your prior group experience? If so, please describe them. [Open-Ended]
6. After experiencing this asynchronous group assignment and considering your prior experience with online group projects, which style of assignment do you prefer?

   ○ I prefer group assignments that are asynchronous like this assignment.

   ○ I prefer my prior group projects.

   ○ I have no preference.

7. How comfortable are you with using the Issues feature on GitHub?

	1	2	3	4	5	
Not comfortable	○	○	○	○	○	Very comfortable

8. How comfortable are you with creating branches on GitHub?

	1	2	3	4	5	
Not comfortable	○	○	○	○	○	Very comfortable

9. How comfortable are you with merging code changes on GitHub?

	1	2	3	4	5	
Not comfortable	○	○	○	○	○	Very comfortable

10. How familiar are you with the computer programming concept of class libraries?

	1	2	3	4	5	
Not familiar	○	○	○	○	○	Very familiar

11. How familiar are you with the concept of unit tests?

	1	2	3	4	5	
Not familiar	○	○	○	○	○	Very familiar

12. How prepared do you feel for a job in a software development team?

	1	2	3	4	5	
Not prepared	○	○	○	○	○	Very prepared