Writing Infrastructures:
GitHub in the Technical and Professional Communications Classroom

Stephen J. Quigley[1], Esther Lui[2], Samantha Whelpley[3], and Joseph Flot[4]

1 University of Pittsburgh

2 National Taiwan Normal University
3 VISIMO
4 University of Pittsburg

Abstract
GitHub provides a project hosting platform and Git-based version control system for individuals and teams looking to develop and manage software and documentation online. Technical writers have long played an important role in this process, contributing the documentation infrastructure that organizes and sustains project development. As GitHub continues to grow in popularity, the field of technical and professional communication (TPC) educators will need to devote more effort to researching GitHub while developing both critical pedagogies and industry best practices committed to design justice. This paper provides a primer for this discussion as well as tools and scaffolding designed to assist GitHub implementation in the TPC classroom.

GitHub provides a project hosting platform and Git-based version control system for individuals and teams looking to develop and manage software and documentation online. Technical writers have long played an important role in this process, contributing the documentation infrastructure that organizes and sustains project development. As GitHub continues to grow in popularity, from 73 million users in 2021 to an expected 100 million by 2025 (GitHub 2021), the field of technical and professional communication (TPC) educators will need to devote more effort to researching GitHub while developing both critical pedagogies and industry best practices committed to design justice. This paper provides a primer for this discussion as well as tools and scaffolding designed to assist GitHub implementation in the TPC classroom. We begin by situating GitHub in relation to TPC discussions of technical skills and coding literacy, then offer an introduction to GitHub as a platform, site of analysis, and teaching tool. Next, we provide descriptions of sample assignments that can be used to scaffold code literacy in the TPC classroom and share student GitHub project narratives. Finally, for readers who wish to learn more about GitHub and/or adapt sample assignments and projects, and in the hope that the field will continue to build on this work, we offer resources from an open-source GitHub workshop we facilitated at the 2021 ATTW Conference.

GitHub in the Technical Communications Classroom

Technological innovation presents ongoing challenges to TPC educators who must monitor industry tools and trends while judging whether and how to address these innovations in the classroom. On the one hand are those educators who believe it our responsibility to prepare students for technical skills they will surely need for the workplace. In "Quo Vadis Technical Communications?", Sides (1994) argued long ago that our field work more closely with corporations to ensure our curricula addresses industry needs, including technical skills like computer programming. Indeed, today, in that employers increasingly favor micro-credentialing over traditional degrees as a key indicator determining future employee outcomes (Fong et al 2016, 3; Milligan and Kennedy 2017, 3-4), universities and departments may need to rethink how to incorporate micro-credentialing into their curricula, or at the least, better ensure student competencies in these hard and soft skills. Others in our field, Opel and Rhodes (2018) for instance, are more skeptical of industry practices. Working first from Katz (1992), Opel and Rhodes argue that assimilation of industry practices like usability studies exposes our students to the "expediency" and "efficiency" inherent in product development (76). In their article, they argue instead for the kinds of humanistic practices educators can bring to the classroom space to disrupt industry ones like usability studies. Their critical pedagogical framework, a "theory + play" approach derived from Learner Centered Design (LCD) pedagogies, invites students to test the creative fecundity and learning capacity in human + design relations, modeling how we might view critical/technical tension as a productive one (78-9). While their "theory + play" approach offers students a space for making and learning, it won't necessarily prepare students working with UX (User Experience) methods using the industry's toolkit. Others in our fields have modeled a third approach that employs critical pedagogy to improve industry practices. Walton's (2016) call to adopt the first principles of design in technical communications provides an example of how we might tread this middle ground. Echoing Buchanan's (2001) similar call in the field of Human Centered Design (HCD), Walton exhorts technical communicators to first consider the dignity of the other—those with whom we communicate directly or indirectly through our designs. Both Walton and Buchanan before her draw heavily on Kant's second formulation of his categorical imperative, which states that individuals should treat the other as the end rather than a means to achieving an end, for otherwise, "everything has either a price or a dignity" (Kant qtd. in Walton, 409). Recent scholarship in other fields will surely inform our discussion and our field moving forward. Costanza-Chock's (2020) critique of HCD argues that intentionality matters little if our processes do not labor to right long-standing inequalities and inequities (77-8). While HCD approaches leverage user insights and add to the prestige of the professional design researchers and practitioners, Costanza-Chock wonders how they benefit the user or redistribute power and control to the communities they engage. Ultimately, Costanza-Chock reminds us that we must do the work of establishing just outcomes through just practices (40-1).

If we desire our teaching and research in TPC to inform industry tools and best practices, then we should similarly investigate how industry tools and best practices might inform our classroom pedagogy. In the case of GitHub, we must explore critically, exposing students to a variety of cases, tools, and methods, working to navigate each situation justly. This will require a discussion of digital redlining, a term Gilliard coined and defined as "Tech policies, practices, pedagogies, investment decisions that reinforce class and race boundaries" (Stachowiak 2016). Scholars in other fields have incorporated GitHub into their classroom practices for a variety of purposes, often in place of traditional Learning Management Systems (LMS), to provide students with course repositories and project management tools (Zagalsky et al 2015, 1914-5). Several scholars in our field have also employed GitHub in their technical communications classroom towards a variety of ends. Watson's (Brewer, Grady, and Watson 2017) upper-level technical communications course provides a model for integrating technical skills into the technical communications classroom. With the intention of providing exposure to technical skills used in industry, Watson introduced his students to agile project management, scrum, Markdown, Jekyll, Git, GitHub, and a primer on API's. Likewise, Duin and Tham (2019) offer useful considerations and examples for refitting an upper-level technical communications course that introduces students to Darwin Information Typing Architecture (DITA), Extensible Markup

Language (XML), Hypertext Markup Language (HTML), and Cascading Style Sheet (CSS). Over the course of this implementation, Duin and Tham encouraged students to compose technology narratives reflecting on personal interactions with technology. Rather than focusing wholly on "code literacy," which they define as "code," "tool," and "structure" (45), Duin and Tham enlarge their praxis to address Vee's concept of coding literacy (55), the "socially situated, symbolic system that enables new kinds of expressions as well as the scaling up of preexisting forms of communication" (Vee 2017, 3). While these scholars make a strong case for technical skills/code literacy in upper-level courses, we must also consider how TPC educators might establish these literacies in introductory TPC courses. Like Duin and Tham (2019), we believe Vee's (2017) concept of coding literacy may offer opportunities for such an ingress.

For many of those in our field, the technological component in what Cargile Cook (2002) refers to as our "layered literacies" consists largely of our ability to interact with computers through user interface (UI) design. Robinson et al's (2019) study of the digital practices of some 328 educators in the fields of Writing and Communication supports this claim in that the words "code" and "coding" are not mentioned in their report. Instead, their study points to a reliance on drag-and-drop website tools like SquareSpace, Wix, Weebly, and WordPress (2). Luckily, Vee's (2017) coding literacy framework provides us with a simple starting place for addressing these literacies and reminds us that we are already using code all the time—typically through a user interface, but programming and writing code all the same. Rather than viewing writing code or programming as learning a new skill, perhaps we should look to strategies that are more akin to unconcealing, as Vee suggests, what we are already doing when we interact with digital tools. Other factors that we might associate with coding literacy include a sense of confidence in knowing how code works, participation in local and remote discourse communities, and lastly, the development of rhetorical awareness and rhetorical agency. Beyond these categories, Byrd's (2019) research suggests other factors that may encourage coding literacy, even those not directly related to technology, including specific incidents, objects, and relationships that might favorably position an individual in relation to technology. In the following section, we offer an introduction to GitHub as a platform, site of analysis, and potential TPC teaching tool for scaffolding coding literacy.

Introduction to GitHub

GitHub provides server space for storing different versions of software and integrated Git-based tools that developers use to track changes in documentation, thus ensuring that one individual's contributions do not overwrite another's. In some ways, collaborating in GitHub using Git version control is akin to working remotely with a partner on a Google Doc. Like Google Docs, collaborators in GitHub can suggest changes and even overwrite one another depending on the preconfigured settings. But unlike Google Docs, individuals do not collaborate in realtime. Instead, individuals either work on different branches of the same project or on the same branch while negotiating a tightly controlled system of protocols designed to prevent one individual from overwriting another. Linus Torvald, a leader in the free and open-source software movement, invented Git version control in 2005 to enable teams of individuals working remotely to collaborate in the development of the Linux kernel, the underlying program that runs the Linux Operating System. In addition to GitHub, a number of other companies, such as GitLab, Bitbucket, and SourceForge, provide similar software development platforms offering comparable repository space and Git version control systems.

GitHub users access GitHub in a variety of ways. The majority write Git commands using the command line on their local computer, sending instructions to both their local computer and the remote GitHub server. Due to the high bar of writing Git in the command line, GitHub provides simple Git functionality integrated into their website's user interface, desktop app, mobile phone app, and even Atom, GitHub's code editor. Along with Git version control, GitHub furnishes other developer tools to manage a wide range of permissions, govern how individuals contribute, facilitate communication, organize scrums and sprints, and automate continuous integration. GitHub provides a variety of affordances for repository owners to control the levels and kinds of interactions they maintain with other contributors working within their repository. While some GitHub repositories are kept private, the majority of repository owners adopt a free and open-source philosophy to writing and maintaining code online. Each repository is owned by either a single individual or by a group of users, and at any point, depending on the permissiveness of a software license, any other individual within the GitHub community can copy that software to a new repository and develop it further, either individually or with a new set of collaborators.

We should note that while open-source development ensures that individuals can view the code that runs a given software, and even run and modify the code on their own computer, open-source does not denote that a software is "free." To the contrary, the licensor of an open-source software determines the level of software licensing, which varies by permissiveness, conditions, and limitations. Some versions of free open-source software licenses, such as Public Domain, MIT (Massachusetts Institute of Technology), and BSD (Berkeley Source Distribution), are highly permissive, but others like GPL (General Public Licenses) require more of the user, including sharing and stating local changes with the larger opensource community under a practice referred to as copyleft (Stallman 2009, 91-2). While sites like GitHub aid in the distribution of open-source software, they are especially useful for GPL software development communities because they allow any individual the freedom to examine the full history of a given software, both changes to that software at its source and instances of branching development performed by other users downstream.

For example, at the time of this publication, the WordPress development repository on GitHub (https://github.com/WordPress/wordpress-develop) had some 42 development branches within the repository, each with its own changes made by 74 repository contributors, but also another 1,000 forked repositories downstream, each with its own owner(s), development branch(es), and development practices, the sum total of which may be inspected by any user at any time, in compliance with the original software GPL licensing.

Large corporations, like Adobe, Amazon, Facebook, Google, and of course Microsoft, also leverage GitHub open-source repositories to develop software. By doing so, these companies stand to benefit from non-employees who make contributions to improve or customize a given software following the theory of lead-user theory (Von Hippel 1986, 792-8). According to this theory, customers, rather than product developers, provide the most significant product innovations. Thus, the companies that nurture user innovation stand to gain from both existing product iteration and new product development. Companies that implement what has become known as lead-user innovation methodology will also profit from resources that would have been traditionally reinvested in research and development. Brackets, Adobe's code editor development project, provides a good example of this kind of product development model. Adobe built the basic functionality for Brackets (including a code preview toggle that made the editor extremely popular) and then permitted users to create development plug-ins and other add-ons to further customize the editor. While Adobe eventually abandoned the product, they gained insights into design and customization that sped up their product development timeline and could possibly have led to innovation in other product areas.

To facilitate network interaction, GitHub leverages elements of social media to help repository owners better showcase their projects, circulate content, and attract user contributions. GitHub incentivizes these user contributions by tracking them, graphically displaying total contributions over time on the user profile page, and by keeping running tabs of all contributors on the repository page. Contributors participate on other GitHub users' projects for a variety of reasons: to learn about and develop software, to work in direct or adjacent collaboration, and sometimes just to show off their skills to the larger GitHub community. Dabbish et al (2012) note the transparent design of GitHub's track change and commenting process contributes a great deal to whether and how individuals interact with one another within repository spaces. As regards to the social nature of GitHub, their study cites "attention signaling," praise from fellow contributors in the feed, and "action signaling," problems that arise in the feed that need action, as major drivers for user contributions (1281-2). The public nature of writing code in an open-source community combined with the comment feeds in GitHub can result in a highly competitive, if not toxic, environment. As a result, GitHub encourages repository owners to establish clear guidelines directing how work gets accomplished within their repository, including statements on inclusion and rules governing interpersonal communication. Still, as Prana et al (2021) note, only 10% of the most popular open-source repositories employ a code of conduct, a statistic that should broadcast a greater call to justice (12).

GitHub presents itself as a meritocracy, a networked marketplace connecting individuals similarly interested in developing software. But we should actually see GitHub as something more complex: an environment where corporations and individuals stand to profit in various ways through their own labor and the labor of others with whom they collaborate directly or adjacently within the GitHub network. Words like "profit," "marketplace," and "labor" should recall Bourdieu's (1986, 16) four forms of capital. While GitHub provides a space where individuals can develop their habitus and accrue different kinds of capital (economic, cultural, social, and at times symbolic), we should also acknowledge the inequalities and inequities that contribute to and are made manifest by these systems and structures. Costanza-Chock (2020) notes the inherent injustice of corporations employing lead-user innovation to benefit from user contributions, especially in that these corporations do not directly engage with "race, gender, class, or axes of structural inequity" (79). We must also recognize the large number of individuals who either lack access to GitHub or may be limited by other factors. Prana et al (2021, 11-12), for example, report a consistently large gender gap in GitHub contributors across all regions of the globe. Their study cites toxic work environments and lack of project infrastructure as contributing factors. These researchers offer recommendations to address the inequities in how GitHub values contributions and the inequality often engineered into project infrastructures, including increasing the number of repositories with codes of conduct, providing inrepository mentoring for women, and offering more recognition for female contributors. They also suggest GitHub automate information distribution and tasks like document and developer assignments to increase access and ensure a more equitable distribution of opportunities (12-13).

Since its launch in 2008, GitHub has grown into a robust collaborative writing infrastructure attracting some 73 million users merging over 170 million pull requests per year (GitHub 2021). Though GitHub is free to most users, the site has managed to net some US $300 million in 2018, the same year Microsoft bought it for U.S. $7.5 billion. As GitHub continues to grow in popularity, the platform has created shifts in not only how work gets done, but how employers hire. Computer science researchers suggest that a candidate's GitHub repositories and code contributions could provide employers a better assessment of skills than resumes or recommendations (Saxena & Pedanekar 2017, 299-300). Because GitHub also provides a primary site where technical writing gets done, our field should closely consider these findings and determine if this data warrants adjustments to our own classroom practices. In the next section, we make the case for GitHub as a platform for collaborative writing assignments in the introductory TPC classroom.

Collaborative Writing Projects in GitHub

Because introductory technical communication courses attract a variety of students coming from different majors, fields, and frameworks, each with different skills and interests, collaborative writing projects in these classes can provide dynamic learning opportunities for students. In addition to developing coding literacy, such assignments can encourage students to dream big as they work together to solve problems and make an impact. However, one frustration associated with collaborative projects is their tendency to be one-off projects. For example, a group of students will put in a great deal of time into a project, the course ends, the students move on, the project dies. Another frustration with such assignments is that students tend to create the same kinds of projects semester after semester, each group starting from square-one, reinventing the wheel.

What if instead, we began to think about the benefits of free and open-source philosophy and tools like GitHub? What if student projects didn't just die off at the end of the semester? Maybe instead of only focusing on developing project deliverables, we focused more on project infrastructure and growing long tails. Though the concept of long tails goes back to a 1946 paper identifying a logical fallacy or heuristic in statistics (Brown and Tukey), Wired Magazine's editor-in-chief Chris Anderson (2006) popularized the term while explaining how shifts in online business strategy were extending the life of products and revolutionizing the digital economy. Anderson recognized the key factors in long tail products and provided examples of companies like Netflix and Amazon who relied more heavily on sustaining customer relationships and extending the life of their product than selling individual products one-off at a point-of-sale. Anderson notes that while the initial sales profit for a given product may be high, a video game for instance, the aftermarket purchases, mods, and updated versions often result in graphs in which the total size of tail profit may dwarf that of the head (20-26).

The concept of long tails also has implications and applications for the work we do in the TPC classroom. When we encourage students to construct projects that are by nature free and opensource, when we encourage modularity and strong documentation detailing project goals, expectations, and development, then student work starts to grow long tails. As a result, students in subsequent semesters can then fork the work of others, perhaps choosing to continue working on the same problem or choosing to adapt the design to solve new problems. Either way, the work goes on, one way or another, sometimes in form, sometimes in kind and content, from one semester to the next. In the case of a project named Virtual Advisor, which will be referenced in more detail later in this paper, subsequent groups augmented and refined what a prior group had accomplished, while others still, refitted the project, or parts of it, for other purposes.

Before students embark on project development, they should consider the ethics and implications of their project designs for a wide range of individuals within our classroom and without. Who is this project for? Who gets to design it? What are the power dynamics by which this design might come into fruition? To what degree does this project implementation constitute cultural appropriation? If we wish to promote design justice through our design pedagogies, each of us should consider the principles of design justice (Costanza-Chock 2020, 190-204). Rather than focusing wholly on product, we should stress the importance of a product architecture that seeks external contributions more aligned with design justice principles emphasizing process over product. Doing so requires students to draft policy, including ReadMe's, inclusive documents, codes of conduct, licensing, and other documentation that will encourage open, inclusive, fair, and ethical practices.

In addition to learning about project development, collaborative projects provide an opportunity for students to develop project management skills aligned with the principles of design justice. Duin et al's (2017) work extends the concept of radical collaboration (Hamm 2008; Simons, Gupta, and Buchanan 2011) by presenting a model for managing the stages of collaborative ideation based on mutual respect. Counter to other hierarchical project management philosophies, Duin et al's radical collaboration resists roles initiated by power dynamics and instead prioritizes "visibility, curiosity, empathy, and open mindsets" among members (67). Thinking along with Pope-Ruark (2015), I wondered how students working together in non-hierarchical settings towards similar ends may benefit from exposure to scrum strategies that emphasize communication and accountability. To this end, we encourage educators to incorporate field texts written by technical communications practitioners in that they provide fodder for discussion. Gentle's Docs Like Code (2017) and Gales' Product is Docs (2017) offer narratives describing collaborative teams working synchronously and asynchronously in industry. While industry methods may contrast with our classroom practices, they do provide juxtaposition and thus a point of discussion to assist students in forming their own ideas about how work should get done beyond the classroom.

Scaffolding Code Literacy

The following assignment cycle offers a model for scaffolding code literacy in the classroom. Rather than focusing on technical skills per se, this assignment cycle attempts to infuse code into the kinds of assignments we were already doing in the TPC classroom, including introductory icebreakers, writing product directions, building professional portfolios, and making project pitches.

Assignment #1 Electrate Fuego - In prior years, students in my "Intro to Technical Writing" classes would have begun the semester by producing an Adobe Spark webtext to communicate a little about themselves, their interests, their coursework, and their field. Adobe Spark provides a web-based drag-and-drop user interface for building webpages. Students can quickly build content by writing text, adding pictures, and a range of other media content. Adobe Spark employs a range of parallax scrolling effects to add animation to their webtext. While attractive, the program's simplicity negates its educational value. As an alternative, we used a webtext generator called Electrate Fuego. This program, based on Open Fuego coding pedagogy, uses code templates with hidden code comments that can only be viewed once the student opens the file using a code editor. While the code comments provide students with instructions on how to add content to the webtext, they also offer informative information explaining what different parts of the code are doing. Such comments convey essential computer science knowledge like HTML, CSS, and File Management. For example, they might explain what each line of code in the document <Head> is doing and why students need to pay attention to file types and naming conventions. By working through the content, students also develop aspects of computational thinking (Wing 2006). For example, when students read through the code, they will begin noticing patterns in how the code is written using open <> and closed </> symbols (pattern recognition). They will also learn how CSS is used to assign attributes pertaining to a <div class> in HTML (abstraction). They will similarly learn the necessary parts and steps needed to create and circulate a webtext (algorithmic thinking). Finally, the more students work with code, the more they will begin to interact with other webtexts and consider how code is used to facilitate their design (decomposition). This assignment also requires students to think about creating content that is inclusive and accessible so as to avoid digital redlining. Students must ensure that images contain alt descriptions and are sized to accommodate areas with slow internet speed, and test content to ensure it scales across media in that the majority of users access web content solely through the use of cell phones (Pew 2021).

Assignment #2 Directions in Markdown - Students tasked with writing directions in intro to technical writing classes have little trouble imagining their intended audience. This fact, coupled with the constraints of writing short sentences, provides students an excellent opportunity to hone their usage and mechanics skills while keeping the user in mind. These assignments can also be easily adapted into code learning opportunities by requiring students to write their directions in Markdown and publishing their files in GitHub. Markdown is a lightweight markup language that formats text for viewing in browsers. John Gruber (2012) developed Markdown in response to more difficult markup languages, in what he hoped would be a simpler way to create both human and machine-readable texts. The language employs a limited range of symbols and simple syntax to arrange and style user content. Markdown provides affordances for creating standard notation, bulleted points, block text, links, text alignment, and use of bold, italic, and underlined fonts. Creating a level H1 heading is as simple as adding an asterisk (*) and space prior to a line of text. An H2 heading uses double asterisks (**) and a space. Clearly, Markdown achieves its simplicity through design constraints. Students use code editors to write and edit their documents, and then host them as a README.md in a dedicated GitHub repository. In this case, students do not need to deploy GitHub pages because GitHub repositories are designed to automatically display a README.md when a user opens the repository link. Along with giving students an opportunity to hone their technical writing skills, this assignment provides students with an additional opportunity to see the connections between writing code and viewing it on a browser. Each of these activities requires the student to think deeply about their user's needs and experience.

Assignment #3 Bootstrap Website - Once students learn to work with code using Open Fuego, they can easily learn to use other common CSS systems for writing HTML. Working with common CSS libraries like Bootstrap CSS and W3 CSS ensures that students are designing their documents while following design best practice. Black Rock Media, a web design group, designed Bootstrap to standardize their development methods. Rather than writing new CSS pages for each of their clients, the design group could defer to pre-designed attributes. Again, the importance here lies in standardizing best practices for usability and accessibility. Students in this "Intro to Technical Writing" course use a basic Bootstrap template to assemble a professional portfolio. To get started, they select and download a template from Startstrap.com, a free source for basic Bootstrap websites. From here, students must learn how to read through the various files to activate modal windows, email contacts, and other features available in Bootstrap. If students wish to add content, they can find Bootstrap code templates from a variety of sources and help from sites like W3 and Stack Overflow. Rather than hosting their site on GitHub, students in this "Intro to Technical Writing" course are encouraged to host their site on the university's LINUX server. This requires students to use a file transfer program either native to their Windows OS, or in the case of Mac OS, using a file transfer protocol (FTP) client such as FileZilla. Through this activity, students learn how to create directories and subdirectories to host multiple websites and texts on their university server-space with the added benefit of a university URL.

Assignment #4 Project Proposal - This group assignment requires students to work together in GitHub to design content for a group presentation. Students use Open Fuego's Elevator Pitch Generator to create a multimodal communication document to pitch their projects. Students work through a template to answer the four questions of stasis theory: conjecture (what is it?), definition (what will it do?), quality (why is it important?), and policy (what action will we take moving forward?). Because the parts can be easily separated, students often divide the content between the different group members. Students learn quickly that they must either use Git to control the content they add to their repository or risk overwriting changes made by other group members.

Project Implementation

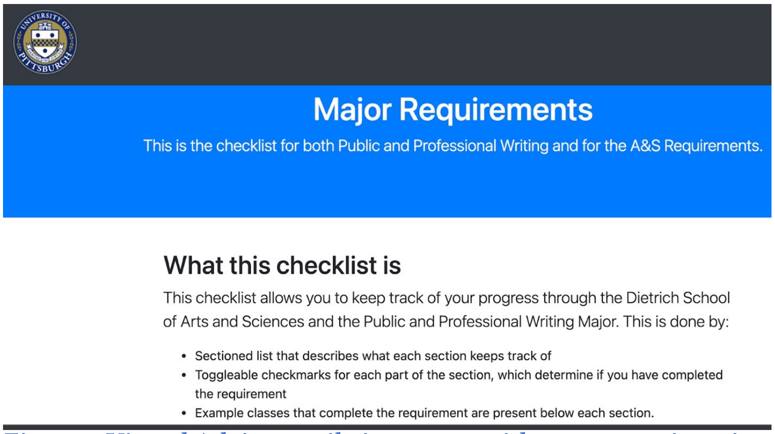
Once students have learned the basics of web development and GitHub, they are ready to learn more about working on teams using project management strategy to develop open-source product deliverables. Each student pitches a web-based product idea and tries to recruit a team of 3-5 students. Projects can be original or variations of projects initiated in other semesters. Each product concept must be vetted in terms of its usability, feasibility, and just process/intentions. These projects can range in kind from training manuals to online guides to tools for queuing or finding a mentor. Some ideas take, others fail to garner support. Once students have enlisted help, they embark on a two week sprint (2-3 scrums per week) to develop a web-based product and product infrastructure. Students open a GitHub repository and divide and designate work for the next scrum using their project Kanban board. Often students will divide into roles (back-end developers, front-end developers, and technical writers producing web content and supporting documentation) but are encouraged to change roles from scrum to scrum depending on project need. Throughout the project, students are encouraged to follow our classroom best practices for project management and must complete mid-project and final project self-evaluations. Students are also responsible for writing product documentation in Markdown to support product development. These documents include a README statement—a document that provides collaborators with software description, instructions, version history, etc. Students must also use Markdown to establish a license, a Q&A or wiki to troubleshoot software problems, an inclusivity statement, and any other documents intended to facilitate a fair and equitable writing infrastructure.

To illustrate project implementation and highlight student experiences and perspectives, we now share narratives from three student projects.

Project Narratives

Pitt Virtual Advisor - Samantha Whelpley

During the Fall 2019 semester, my Technical Writing group worked on a project called Virtual Advisor. The proposed application would aid University of Pittsburgh students in planning and registering for classes. Ideally, it would integrate with the existing PeopleSoft Course Registration system and create a communication platform for students to ask others about specific courses and schedules for each major. After future development by each school within the University of Pittsburgh and their respective advising departments, the web application could easily connect students with resources about course selection across all parts of the university. These resources would include University guidelines and recommendations as well as advice from students, which would be provided through forums.



Major Requirements

This is the checklist for both Public and Professional Writing and for the A&S Requirements.

What this checklist is

This checklist allows you to keep track of your progress through the Dietrich School of Arts and Sciences and the Public and Professional Writing Major. This is done by:

- Sectioned list that describes what each section keeps track of
- Toggleable checkmarks for each part of the section, which determine if you have completed the requirement
- Example classes that complete the requirement are present below each section.

Figure 1. Virtual Advisor easily integrates with current registration software.

We created this project using the features of GitHub and its version-control capabilities. GitHub allowed the four of us to see each other's code and easily merge it together. Some of the key ideas around using this approach were creating a branch for each person to develop individually and making sure to continuously pull other people's finished work from the remote repository to their local machine.

Out of the four people in the group, I was a Computer Science major, two other members were engineering students, and the fourth student was a Public and Professional Writing (PPW) student. This brought together students with a variety of skill sets. Even though all of us had some exposure to code in this class or prior to it, I was the only one with actual GitHub experience. By sharing Git and version control best practices with the group, we were able to work collaboratively on a project with ease. GitHub is a great version-control and collaborative tool because of its various user interfaces. GitHub has both a web GUI and a Command Line Interface for pushing and pulling changes to and from the repository. This allowed the different members of the group to use what was most comfortable and convenient for them.

We split up the work by each of the main three pages so that everyone could contribute to the code. The PPW student also greatly contributed to the content of the site. The site in its current state can be found at Virtual Advisor (sjwhelpley.github.io). Not only does GitHub allow for easy collaboration between group members, but it makes deploying and sharing projects simple and free. Since the initial version, other groups have reworked the tool in newer versions, expanding documentation and improving usability and accessibility.

ideaHub – Esther Lui

Using GitHub as a central platform for sharing information and resources, our team developed an expanded, web-based version of the elevator pitch generator. We were first introduced to stasis theory as an invention tool in Dr. Stephen Quigley's "Introduction to Technical Writing," where the ideaHub project began. Our team of three envisioned ideaHub as a fully functional and responsive web application that would not only allow university students to come up with well-developed project ideas, but also serve additional functions like creating greater inclusion through team member recruitment, pitch browsing and filtering, as well as making connections with sponsors and mentors.

Each of the three members on our team came from varying academic backgrounds and levels of experience. Two on our team studied computer science, one with extensive practical experience with software engineering. The third member majored in professional writing, with some exposure to UI design and frontend web development. Accordingly, we determined what roles were needed — software developer, technical writer, and designer, respectively — often with overlapping tasks requiring handoff and close collaboration. GitHub was useful in these circumstances to keep track of which elements were contributed by which member as well as to maintain version control.

From the ideaHub landing page, users first create an account. From there, students can immediately create a project pitch. In line with the four stases of conjecture, definition, quality, and policy, we developed a pitch developer tool that features these four question types. Each question was further adapted into more colloquial language and broken up into sub-questions for ease of understanding. Next in this process, users are prompted to come up with a name for their idea and are given the opportunity to add an image and tags to their idea.

When users complete and upload their unique project pitch, ideaHub adds student pitches to its database. From here, students can now browse others' ideas, filtering by school or institution, or by searching for relevant tags (such as "healthcare," "non-profit," "statistics," etc.). Clicking on an idea brings up its project pitch as well as options to follow or apply to join that particular team. When a student applies to join a project, the idea owner receives a notification via email and can connect with the applicant to form a new team.

Our team was successfully able to launch a functional ideaHub prototype after the first scrum sprint that included fully functional pages for pitch development, browsing other pitches, viewing a user's own pitches, as well as viewing "followed" pitches. The final ideaHub site map is as follows in Figure 2.

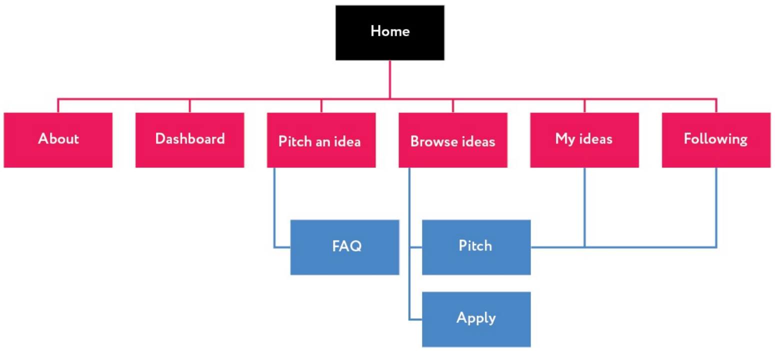
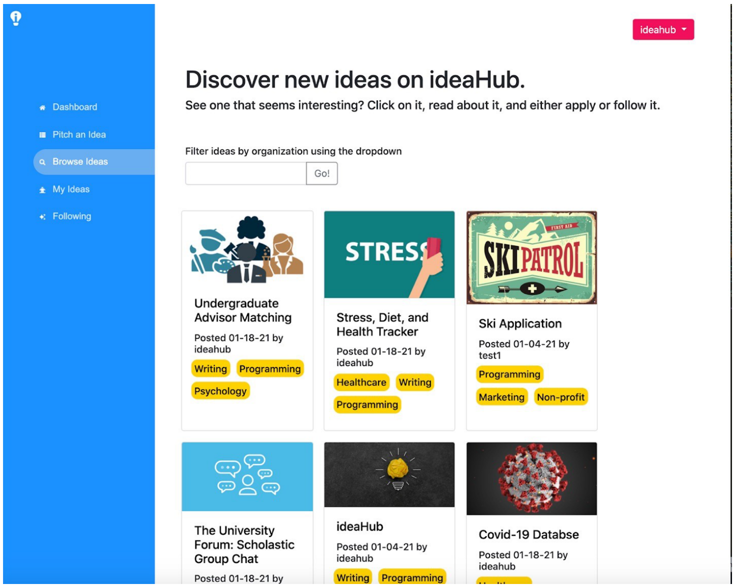


Figure 2. ideaHub sitemap displaying the hierarchy of pages

Ultimately, and perhaps most importantly, we hope ideaHub can become a useful model for other developers to interact with in the future. Because all project resources were housed in GitHub, it is also accessible for anyone to pick up, repurpose, and re-imagine any parts of the project for their own purposes. See the figure below for a screenshot of the final project pitch database.



Discover new ideas on ideaHub.

See one that seems interesting? Click on it, read about it, and either apply or follow it.

Filter ideas by organization using the dropdown

Undergraduate Advisor Matching
Posted 01-18-21 by ideahub
Writing Programming Psychology

Stress, Diet, and Health Tracker
Posted 01-18-21 by ideahub
Healthcare Writing Programming

Ski Application
Posted 01-04-21 by test1
Programming Marketing Non-profit

The University Forum: Scholastic Group Chat
Posted 01-18-21 by

ideaHub
Posted 01-04-21 by ideahub
Writing Programming

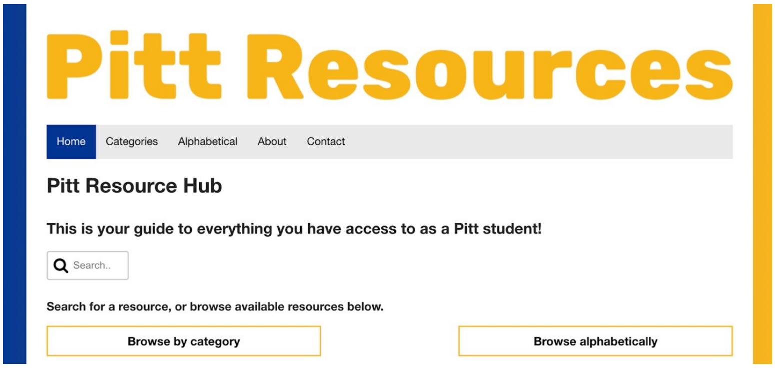
Covid-19 Databse
Posted 01-18-21 by ideahub

Figure 3. The final layout of the "Browse Ideas" page, displaying a database of all idea pitches.

Pitt Resource Finder - Joseph Flot

Unified Pitt Resources sought to create a central hub which allowed easy access and discovery of Pitt resources. Our group noticed many students did not know of these assets or how to access them. We attributed this lack of awareness to the departmental segregation of resources. Engineers know of engineering resources, but not of biology or liberal art resources. This reduces the discoverability of tools and information accessible to all students. We embarked on this project as a group with diverse backgrounds of skill and experience. I had trepidations about working on a project of this scale with four other people. I did not have much experience with GitHub nor with coding. I felt I would not be able to pull my weight with the skills I brought to the table.

To my surprise, this project was my most effective group collaboration to date and taught me several lessons. First, the diversity of skills I thought would leave me unable to contribute instead helped my own skills stand out. I could share my abilities and learn more about their own by embracing the skills of my peers. I learned the importance of proactivity. Before we began working, we all agreed to a policy of clarity and open communication. This meant that if someone had an issue, they should tell the group and we would handle it as a collective. In practicing this policy, we were able to maintain flexibility in the workload distribution. This also kept one person from becoming stressed or resentful towards another group member. We often consider emotion last when working in a professional setting, but when you work with humans, you must treat them like humans.



Pitt Resource Hub

This is your guide to everything you have access to as a Pitt student!

Search for a resource, or browse available resources below.

Figure 4. Unified Pitt Resources searchable database.

The flexibility with roles also lent itself to learning outside of my comfort zone. I served as the graphic designer but, when I had little work to do, I helped with the web design as best I could. This ability to share work between roles did not diminish the importance of roles in the first place. Knowing who does what allows for a flow of information to be established between group members. The coders can ask me for a design, and I will pass that onto the coders and let the document writers know of the change. I have learned that static roles allow for the assignment of tasks ahead of time. At the beginning of the project, the person who worked on the final presentation knew two weeks ahead of time.

My individual role as a graphic designer taught me about creating within technical constraints and ensuring accessible design. Much of the graphics on the website are icons no larger than a postage stamp. I appreciated the challenge of making a well understood creative design under these restrictions. I noticed that researching how others had overcome these challenges by looking at other sites sped up the process. I also learned the importance of real time demonstration in developing a design. On Zoom I could share my screen and make a draft of a design and take suggestions as my group thought of them. This cut hours off the drafting and revising process a normal workflow might have. Designing many visually similar graphics also forced me to rethink my personal drafting process. In most circumstances, I am working on a single piece, and no one will see any of the files. This leads me to have an erratic and sometimes nonsensical naming scheme, though now I have a much more organized naming process.

With more than a year having passed since this experience, I have had several more group projects. Each of those benefitted from the collaborative approach I learned here. Involving group members in the iterative process of project completion and applying their skills improves any sort of endeavor. Furthermore, having a group with diverse skills and backgrounds empowers each of the members to contribute their unique abilities. Learning to balance the rigidity of roles with the plasticity of human experience makes for a more creative problem solving. I hope to carry these lessons into my own field of microbiology.

Discussion

Because technical writers often work in Git-based version control writing environments like GitHub (Brewer et al. 2017, Gentle 2017, Gales 2017), TPC educators should do more to introduce these systems and technologies in their classrooms. To be clear, GitHub is both vast and imperfect. The work of Prana et al (2021), for instance, reveals that GitHub mirrors much of the inequity and inequality we see in the world. Yet GitHub provides a space for our students to test how they might attend to the problems any technology can bring to bear on both designers and users, individuals and communities, and to explore more ethical and equitable design practices that might result in more ethical and equitable outcomes. As the above group project narratives demonstrate, building open-source projects in GitHub requires student groups to think carefully about how to create and sustain inclusive work environments through both project documentation and project management. The first is theoretical, the second relates to practice. The concern for Costanza-Chock (2020) is what happens next? While designing open-source projects may invite participation and iteration, it doesn't ensure design tools and knowledge are equally distributed so that all may participate in the design. To address this need, we will lastly share our ATTW GitHub Workshop we designed to introduce individuals to GitHub practices and workflow.

The ATTW GitHub Workshop

We designed an open-source GitHub workshop that anyone can implement or adapt beyond our ATTW conference date or ATTW community. Any individual who forks the repository may alter any text or methods to suit their needs. They may also contribute to our project repository by pushing changes to the main repository using either of the two prescribed collaboration methods. Our workshop intends to achieve three specific goals providing each participant with an opportunity to learn why a modicum of code literacy is essential for Tech Comm educators, how to scaffold code literacy learning using GitHub, and finally, how to use GitHub Pages, Git UIs, and GitHub workflows.

Method #1 Forked or InterRepository Collaboration - This method allows individuals to collaborate between two or more repositories. Individuals begin by forking an existing project repository to create their own version of that product repository downstream. This allows each user to make specific changes to their own software while contributing changes to the original repository upstream. While this version of collaborating on GitHub occurs asynchronously, both individuals can benefit from changes made in the upstream or downstream repository.

Method #2 IntraRepository Collaboration - This method allows groups to work with collaborators within a single repository. GitHub allows repository owners to designate different levels of permissiveness for each collaborator. Regardless, each collaborator can clone a repository using Git or GitHub desktop UI, make edits, and commit changes. Designated members can resolve conflicts and merge changes. Working within a single repository has other organizational benefits. Teams can use a Kanban board to assign roles and scrum tasks.

Best Practices for GitHub Group Work

For Teachers

- scaffold learning by including code literacy into everyday activities
- let students take control of their learning
- encourage agile project management and product development methods
- provide examples of good design practices
- provide resources, not answers
- promote design justice pedagogies and practices

For Students

- use GitHub desktop UI or learn Git (merely uploading files overwrites others' work)
- assign roles
- distribute specific tasks
- write clear comments

- make pull requests specific to task
- designate a merge master / or set commit privileges to repository owner
- don't sit...COMMIT - make pull requests often / push requests oftener.

GitHub / Code Resources

GitHub Guides - an excellent set of tutorials for GitHub learning.
Open Fuego - code tools designed to support the things we are already doing in our classroom.
W3 - a wide range of code reference, tutorials, and a sandbox that allows you to play with code.
Stack Overflow - Code questions and aggregated answers.
Sample Developer Team Workflow - a decade-old branching model
GitHub Accessibility - view accessibility issues working with GitHub.

References

Anderson, Chris. 2006. The Long Tail: Why the Future of Business is Selling Less of More. New York: Hachette Books.

Brewer, Pam Estes, Helen Grady, and Robert Watson. 2017. "Diverging Currents: Continuous Innovation in an Engineering-Based Technical Communication Program." In 2017 IEEE International Professional Communication Conference
(ProComm) IEEE (2017): 1-10.

Brown, George W, and John W. Tukey. 1946. "Some Distributions of Sample Means." The Annals of Mathematical Statistics 17(1): 1-
12.

Bourdieu, P. 1986. “The Forms of Capital.” In Handbook of Theory and Research for the Sociology of Education, edited by J. G. Richardson, 241-258. New York: Greenwood.

Buchanan, Richard. 2001. "Human Dignity and Human Rights: Thoughts on the Principles of Human-Centered Design." Design Issues 17(3): 35-39.

Byrd, Antonio. 2019. "Between Learning and Opportunity: A Study of African American Coders’ Networks of Support." Literacy in Composition Studies 7(2): 31-56.

Cook, Kelli Cargile. 2002. "Layered Literacies: A Theoretical Frame for Technical Communication Pedagogy." Technical Communication Quarterly 11(1): 5-29.

Costanza-Chock, Sasha. 2020. Design Justice: Community-led Practices to Build the Worlds We Need. Cambridge, MA: The MIT Press.

Dabbish, Laura, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 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: 1277-1286.

Duin, Ann Hill, and Jason Chew Kit Tham. 2019. "Cultivating Code Literacy: Course Redesign Through Advisory Board Engagement." Communication Design Quarterly Review 6(3): 44-58.

Duin, Ann Hill, Megan McGrath, Jason Tham, and Nathan Ernst. 2017. "Design Thinking Methodology: A Case Study of 'Radical Collaboration' in the Wearables Research Collaboratory." Connexion: An International Professional Communication Journal: 47-74.

Fong, Jim, Peter Janzow, and Kyle Peck. 2016. "Demographic Shifts in Educational Demand and the Rise of Alternative Credentials," accessed March, 2021, https://upcea.edu/wpcontent/uploads/2017/11/Demographic-Shifts-in-Educational-Demand-and-the-Rise-of-Alternative-Credentials.pdf

Gales, Chris. 2017. Product is Docs. NP.

Gentle, Anne. 2017. Docs like Code. Austin: Texas. Just Write Click.

Gruber, John. 2012. Markdown: Syntax, accessed March, 2021, http://www.daringfireball.net/projects/markdown/syntax.

Github. 2021. The 2021 State of the Octoverse, accessed May, 2022, https://octoverse.github.com.

Hamm, Steve. 2008. "Radical Collaboration: Lessons from IBM's Innovation Factory." Human Resource Management International Digest 16(2). https://doi.org/10.1108/hrmid.2008.04416bad.009

Katz, Steven B. 1992. "The Ethic of Expediency: Classical Rhetoric, Technology, and the Holocaust." College English 54(3): 255-275.

Milligan, Sandra and Gregor Kennedy. 2017. "To What Degree? Alternative Micro-Credentialing in a Digital Age." Visions for Australian Tertiary Education: 41-54.

Opel, Dawn S. and Jacqueline Rhodes. 2017. "Beyond Student as User: Rhetoric, Multimodality, and User-Centered Design." Computers and Composition 49: 71-81. DOI: 10.1016/j.compcom.2018.05.008

Pew Research Center. 2021. “Mobile Fact Sheet.” April 7, 2021. https://www.pewresearch.org/internet/fact-sheet/mobile/

Prana, Gede Artha Azriadi, Denae Ford, Ayushi Rastogi, David Lo, Rahul Purandare, and Nachiappan Nagappan. 2020. "Including Everyone, Everywhere: Understanding Opportunities and Challenges of Geographic GenderInclusion in OSS." IEEE Transactions on Software Engineering, 2021. (note: 2020 preprint) arXiv preprint arXiv:2010.00822.

Pope-Ruark, Rebecca. 2015. "Introducing Agile Project Management Strategies in Technical and Professional Communication Courses." Journal of Business and Technical Communication 29(1): 112-133

Robinson, Joy, Lisa Dusenberry, Liz Hutter, Halcyon Lawrence, Andy Frazee, and Rebecca E. Burnett. 2019. "State of the Field: Teaching with Digital Tools in the Writing and Communication Classroom." Computers and Composition 54: doi: https://doi.org/10.1016/j.compcom.2019.102511.

Saxena, Rohit and Niranjan Pedanekar. 2017. "I Know What You Coded Last Summer: Mining Candidate Expertise from GitHub Repositories." In Companion of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing: 299-302.

Sides, Charles H. 1994. "Quo Vadis, Technical Communication?" Journal of Technical Writing and Communication 24(1): 1-6.

Simons, Tad, Arvind Gupta, and Mary Buchanan. 2011. "Innovation in R&D: Using Design Thinking to Develop New Models of Inventiveness, Productivity and Collaboration." Journal of Commercial Biotechnology 17(4): 301-307.

Stachowiak, Bonni. 2016. “Interview with Chris Gilliard; Digital Redlining and Privacy.” Teaching in Higher Ed. Podcast audio. December 8, 2016. 35:48. https://teachinginhighered.com/podcast/digital-redliningprivacy/

Stallman, Richard. 2002. Free Software, Free Society: Selected Essays of Richard M. Stallman. GNU Press.

Vee, Annette. 2017. Coding Literacy: How Computer Programming is Changing Writing. Cambridge, MA: The MIT Press.

Von Hippel, Eric. 1986. "Lead Users: A Source of Novel Product Concepts." Management Science 32(7): 791-805.

Walton, Rebecca. 2016. "Supporting Human Dignity and Human Rights: A Call to Adopt the First Principle of Human-Centered Design." Journal of Technical Writing and Communication 46(4): 402-426.

Wing, Jeannette M. 2006. "Computational Thinking." Communications of the ACM 49(3): 33-35.

Zagalsky, A., Feliciano, J., Storey, M. A., Zhao, Y., and Wang, W. 2015. “The Emergence of GitHub as a Collaborative Platform for Education.” In Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing: 1906-
1917.

About the Authors

Stephen Quigley teaches technical communication and digital media composition at the University of Pittsburgh. His research interests include basic coding pedagogy and networked infrastructure.

Esther Lui is a current MBA candidate at National Taiwan Normal University where she researches barriers to effective knowledge transfer for distributed software teams. She graduated from the University of Pittsburgh with a degree in Chinese and Public and Professional Writing.

Samantha Whelpley is a web developer at VISIMO and recent graduate of University of Pittsburgh's School of Computing and Information.

Joseph Flot is a microbiology major at the University of Pittsburgh looking to specialize in the field of applied microbiology.