

Socially Responsible Computing in an Introductory Course

Aakash Gautam
University of Pittsburgh
Pittsburgh, PA, USA
aakash@pitt.edu

Anagha Kulkarni
San Francisco State University
San Francisco, CA, USA
ak@sfsu.edu

Sarah Hug
CERC
Westminster, CO, USA
hug@colorado.edu

Jane Lehr
Cal Poly San Luis Obispo
San Luis Obispo, CA, USA
jlehr@calpoly.edu

Ilmi Yoon
San Francisco State University
San Francisco, CA, USA
ilmi@sfsu.edu

ABSTRACT

Given the potential for technology to inflict harm and injustice on society, it is imperative that we cultivate a sense of social responsibility among our students as they progress through the Computer Science (CS) curriculum. Our students need to be able to examine the social complexities in which technology development and use are situated. Also, aligning students' personal goals and their ability to achieve them in their field of study is important for promoting motivation and a sense of belonging. Promoting communal goals while learning computing can help broaden participation, particularly among groups who have been historically marginalized in computing. Keeping these considerations in mind, we piloted an introductory Java programming course in which activities engaging students in ethical and socially responsible considerations were integrated across modules. Rather than adding social on top of the technical content, our curricular approach seeks to weave them together. The data from the class suggests that the students found the inclusion of the social context in the technical assignments to be more motivating and expressed greater agency in realizing social change. We share our approach to designing this new introductory socially responsible computing course and the students' reflections. We also highlight seven considerations for educators seeking to incorporate socially responsible computing.

CCS CONCEPTS

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

KEYWORDS

SRC, responsibility, social impact, ethics, power, critical pedagogy

ACM Reference Format:

Aakash Gautam, Anagha Kulkarni, Sarah Hug, Jane Lehr, and Ilmi Yoon. 2024. Socially Responsible Computing in an Introductory Course. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*, March 20–23, 2024, Portland, OR, USA. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3626252.3630926>

1 INTRODUCTION

“Undergraduates need to understand the basic cultural, social, legal, and ethical issues inherent in the discipline of computing ... They should understand their individual roles in this process” argue the 1991 Report of the ACM/IEEE-CS Joint Curriculum Task Force [46]. Similar arguments to integrate ethics have been repeatedly made over the years (e.g., [6, 9–12, 23, 27, 33, 34, 36, 49, 51]). The growing awareness of computing systems' potential to inflict harm and injustice on society has strengthened the argument for integrating ethical responsibilities in computing curricula [21, 30, 50].

Equally importantly, education researchers have long since argued for the importance of ensuring *goal congruity*, which posits that students need to perceive an alignment between their personal goals and their ability to fulfill those goals by participating in the field of study (called goal affordances) for them to pursue a career path in that field [17]. In computing education, a greater emphasis on agentic goals, which has an inward focus on “the self, self-efficacy, and working with things (instead of people)” in contrast to communal goals, which has an outward focus on “working with, or in service of others” has been found to be a barrier in enhancing diversity and inclusion in computing [2, 3, 44]. Evidence from early CS courses that promoted affordances to meet communal goal have been shown to be successful in increasing participation of women, African American, and Hispanic students [3, 13, 38].

Despite repeated calls and well-established benefits, we continue to struggle in incorporating ethics in CS. Scholars have highlighted persistence challenges such as lack of faculty expertise [20, 42], faculty resistance [33, 42], time and resource constraints [5, 20], and the perception that ethics is outside the scope of computing [11, 43]. Efforts to integrate ethics into CS courses are ongoing, with various models and solutions being proposed, ranging from a stand-alone course to incorporating it across CS courses [11, 16, 21, 26, 29]. While we have yet to fully integrate ethics in CS, we also know that ethics integration is not enough. We need praxis-oriented computing courses that build upon ethical considerations toward encouraging students to take responsibility by understanding the power and social impact of technology — that is, engaging with socially responsible computing.

Socially responsible computing goes beyond ethical considerations. It acknowledges that computing enacts a form of power, thus computing professionals need to develop a critical understanding of how technology can perpetuate or challenge societal inequities [11, 29, 40]. The goal is to support computing students, who are



This work is licensed under a Creative Commons Attribution International 4.0 License.

going to be builders of technology in the near future, to develop a sense of responsibility toward the systems they create.

We are embarking on a long journey to incorporate socially responsible computing across major CS courses, beginning with a new CS0 course. The CS0 course was piloted in Spring 2023 with the plan of iterating on it and making it mandatory for all CS students in Fall 2023. In this paper, we share an overview of the CS0 course structure and details from the students’ reflections. The findings highlight the value of integrating socially responsible computing to facilitate students’ understanding of computing in a social context and in deepening their awareness of justice and power relations. We also show that students found the integration to be motivating and examined their responsibilities and structural issues surrounding computing. Overall, the integration facilitated a synergistic opportunity to deepen the learning of both programming and social issues. We conclude by reflecting on seven key considerations for designing a socially responsible computing course.

2 OUR CURRICULAR APPROACH

The course was structured such that students first developed a framework to examine the impact of computing and then learned to program computing systems.

2.1 Computing Around Us

The first three weeks were about programmed systems, not programming. This was to show students the value and power of computing, and with it develop ability to examine the impact of computing. We started by having students observe how their family, friends, and others used technology. Students had to ask people how they used technology for the assignment. This allowed us to discuss technology use in daily life, how designed systems affect people, and technology developers’ power and responsibilities.

We then covered ethical reasoning, power, and social impact analysis. We read and discussed Consequentialism (particularly utilitarianism), deontological ethics, and virtue ethics. The discussions underscored the inevitability of differing beliefs and emphasized the importance of understanding differences in underlying systems of values and the various forces that influence different actors. Highlighting that actions are highly contextualized, we stressed the importance of considering multiple perspectives, evaluating the impact on multiple stakeholders, and carefully considering who might be ignored or harmed by technological systems.

Students’ discussions were structured by asking questions about access and power. Four questions were often presented to the students: who has access to the system and who is ignored, whose data is being collected, who is benefiting the most, and who gets to decide what problems need to be solved. We encouraged students to consider impact on individual, communal, and societal levels.

2.2 Computing By Us and For Us

After three weeks, we began teaching Java programming as in a typical CS0 course (see Table 1). We relied on three practices to integrate socially responsible computing: using socially grounded assignments, introducing projects that intertwined social and technical issues, and encouraging individual and collective reflections.

2.2.1 Socially-Grounded Assignments. We wanted to introduce computing as a personally-meaningful and socially-relevant tool. To

Weeks	Major Topics
1–3	Stakeholders, power, ethical reasoning, and responsibilities
4–7	Programming: Variables, sequential execution, conditional statements
8	Bringing all together: Project 1: Evaluating On-Campus Housing Allocation
9–10	Programming: Repeating Actions through Loops
11	Bringing all together: Project 2: Fairly Dividing Restaurant’s Pooled Tip
12–15	Programming: Arrays and Modular Programming
16	Bringing all together: Project 3: Filtering Interview Candidates

Table 1: Major topics covered in the 16-week long course

that end, we provided multiple opportunities for students to see computing in a familiar, socially-situated context. For example, in a lecture on loops, we worked together to create a digital diary for ourselves. Similarly, in a conditional statement assignment, students were required to devise a method to assist their friends in purchasing birthday gifts. This task required students to keep track of their current preferences as well as their friend’s budget.

Many tasks were centered on family and friends. For example, when we introduced user input processing with Java’s Scanner Object, we assigned them to build a rudimentary conversational agent. We encouraged students to make prompts in their native language to motivate them to involve family and friends. We provided an example in Nepali, the instructor’s mother tongue, so that students could work on the topic in their native language. Along with English, students submitted responses in Spanish, Bahasa Indonesia, and Korean. Similarly, the freedom to steer the conversation in any direction appears to have encouraged students to include others, such as in inquiring about their Pokemon collections, multiple sports interests, and restaurant recommendations.

2.2.2 Projects to Examine Social and Technical. We introduced three two-week-long projects to help students connect their newly acquired technical knowledge to familiar social issues.

Project 1: The first project asked students to evaluate and redesign our university’s first-come, first-served on-campus housing allocation system. Students assessed whether the existing approach is fair, who benefits, and who is harmed. They were asked to speak with others outside of class, finalize key factors for on-campus housing allocation, and justify those choices. They implemented a Java program codifying their proposed system, which required them to demonstrate mastery of conditional statements.

Project 2: This project entailed determining an equitable division of a restaurant’s pooled tip amount. Some students had mentioned in previous conversations that they worked in restaurants, so we used that as the context for the project. The assignment required students to use loops to process an indeterminate pile of restaurant bills. Students were asked to consult with others to finalize the factors they wanted to consider when dividing the pooled tip amount. After justifying their decisions on what constitutes a fair division, they implemented their algorithm for tip distribution.

Project 3: Students created filters to shortlist CS job applicants for a fictitious company. Students were required to use arrays and demonstrate mastery of methods. This project provided a scaffolded exploration of some of the courses available in their pathway to graduation. We provided a hypothetical filtering system that evaluated the candidates based on their grades in CS courses. Initially, they crafted filters based solely on CS course grades. Then we

asked them to talk to people both inside and outside the class to finalize a justified, fairer evaluation criteria. They then implemented the filter for shortlisting the fictional company’s candidates.

The first and third projects were adapted from assignments by Evan Peck [37], with the second having a similar structure. We are happy to share all of our assignments upon request, and plan to follow Evan’s footsteps in making them publicly available soon.

2.2.3 Individual and Class-wide Reflections. We added reflective questions about power to most assignments and all projects to encourage students to think critically about the problem and their approach. We asked students to assess who benefits and who is harmed by their approach, as well as any limitations. In class, students reinforced these reflections by discussing these questions.

For example, near the end of Project 2, we discussed various methods for dividing the pooled tip amount. Some mentioned that they talked to people who waited tables and dealt with customers and felt they should be paid more. Others, drawing from their own restaurant experiences, advocated for equal division between front and back staff, while a small group believed in equity-based distribution, with lower earners receiving more. This sparked discussions on technology’s role in ensuring transparency and fairness as well as the pitfalls of technological solutionism, prompting one student to describe Project 2 as “a good programming practice with a backdrop of a tone-deaf¹ social issue.”

2.3 Data Sources and Analysis

We present data from a pilot class taught in Spring 2023. The course was elective for CS students and open to non-majors. We had 45 students, 21 of whom were computer science students and 17 were studying something other than natural sciences or engineering. Of the 45, 12 were female and 16 were what San Francisco State University categories as under-represented minorities (URM²). 14 students received Pell Grants and 13 were first-generation college students. Post each project, students filled an anonymous survey about the assignments and their perceived roles. The survey had 20 rating questions ranging from 0 to 100, categorized into four groups: Understanding of Computing in a Social Context, Awareness of Justice and Power Relations, Personal Relevance and Responsibilities, and Learning and Conceptual Integration. Open-ended responses were analyzed using thematic analysis [41].

3 FINDINGS

The surveys were optional and the number of responses declined over the three surveys, with 31, 17, and 10 responses respectively (see Table 2)³. The significant non-response rates suggests that there may be non-response bias [15]. Moreover, even though the survey were anonymous, the power differences in the class setting may have influenced the response. Noting these limitations, we do

Likert Scales	Project 1 (n = 31)	Project 2 (n = 17)	Project 3 (n = 10)
Understand Computing in a Social Context	77.3	82.2	79.4
Awareness of Justice and Power Relations	74.7	68.9	67.9
Personal Relevance and Responsibilities	66.1	62.2	64.3
Learning and Conceptual Integration	67.4	74.1	64.5

Table 2: Mean rating scores of the Likert scales (out of 100). not make claims of generalizability but rather provide initial insights on the synergistic value of integrating socially responsible computing in programming classes.

Overall, the survey responses indicate that the students recognized the social impact potential of computing, showed awareness of justice-related considerations, felt a moderate personal connection to the projects, and acknowledged the impact of socially relevant computing on their learning experience. We now report on the students’ reflections from the surveys.

3.1 Understand Computing in a Social Context

Students expressed appreciation for addressing relatable “real-world” challenges, prompting one student to comment, “*I think the main thing I learned was how programming is used to look into real ethical problems, which I think is really cool...*” They stated that the ability to see computing in a broader social context was helpful in appreciating the value of programming knowledge. A student aptly captured this sentiment, “*I love the idea that things go hand in hand with programming, because programming can be under a broad number of things in society.*” They also valued consulting with stakeholders, recognizing it as an opportunity to understand the constraints of computing against systemic problems.

3.1.1 Valuing working on and learning from relatable problems. Throughout the projects, students appreciated the opportunity to work on relatable real-world problems. As one student reflected on Project 1, “*It was meaningful to solve problems directly related to reality through programming.*” The importance of relatable tasks was emphasized by another student, who stated, “*I really enjoyed working on something that was somewhat relevant to us.*”

Some projects were more relatable than others. Reflecting on Project 2, a student remarked, “*This was by far my favorite one of the two, just because I could relate to it so I had a good idea of where to start.*” Similarly, on Project 3, another student noted its future relevance: “*This is going to be something I’m gonna be on the other end of when I start applying for jobs so I hope whoever wrote the algorithm was at least more forgiving than me.*” In contrast, the absence of perceived social impact was discouraging. As one student reflected on Project 2, “*I couldn’t see the social impact of it like I could in the first project. It felt like I was making something someone wanted versus making something I felt made a difference. I felt unmotivated while working and was mostly interested in making the code work rather than why I was writing the code in the first place.*”

3.1.2 Engaging with stakeholders to understand the impact of computing. Engaging with external stakeholders was central to the projects. To help with that, early assignments encouraged students to converse with various individuals and observe their technological interactions. Students generally appreciated this engagement. Yet, a few encountered challenges, as highlighted by a student’s reflection, “*the only challenges I faced when working on this project was*

¹This is an ableist word that the instructor did not catch when it was shared in the survey reflection. This underscores the importance of being mindful of language practices, a crucial aspect of teaching socially responsible computing.

²Our University designates URM as students whose race/ethnicity is “African Americans, American Indians/Alaska Natives, and Latinos”.

³Please find the summary of each individual survey here: https://osf.io/5bstn/?view_only=ba371971bcc94d44830921d3e55406fc

finding people who wanted to be interviewed. I had to introduce myself as taking a survey for student housing."

Some displayed commendable initiative. One recounted spending hours on campus, questioning passersby about housing, successfully engaging with 11 individuals. While many conversed with friends or family, a few expanded their search to community members. In reflecting on this, one student summarized, *"Some students expressed they feel like it's weird that we have to interview, but it kinda makes sense since we are trying to solve a real world issue."*

Many shared gaining new perspectives through the interactions, as expressed by a student who grappled with the complexities of on-campus housing, *"I had a harsh reality of how expensive housing on campus is and I did not know how to support financially weaker backgrounds by giving them more incentive to live on campus since it is in fact not affordable. That was a challenge that I faced while I was working on various aspects."*

3.2 Awareness of Justice and Power Relations

Overall, the students expressed positive value in learning about the social complexities tied to the project. For example, in response to the on-campus housing issue, a student wrote, *"I got [to] learn specifically housing allocation issue at SF since I was unaware being a commuter. It was interesting to learn about the process of it."* Through the projects, students grappled with power, especially developers' power and computing limitations in the face of structural issues.

3.2.1 Reflecting on developers' power-to and power-over. The social context of the projects prompted students to consider the power software developers have in making decisions. Early semester discussions, centered on feminist perspectives on power—specifically the concepts of "power-to" and "power-over" [1]—provided a foundation for students to evaluate power dynamics critically.

As students worked on the projects, they began to reflect on the power they exercised based on the factors they considered and the decisions they integrated into their programs. One student, for example, stated, *"...in this case it was a problem relevant to me because my peers are also affected by the housing situation. I learned that there are many factors within the problem, and the more factors there is then either people are drowned out or helped even more."* Another student, reflecting on Project 1, wrote about the power, *"The biggest takeaway from this project was understanding how the choices we make as programmers can influence the lives of others."*

Power dynamics became even more evident to some students during Project 2. One of them remarked, *"I learned how computing has real world power and can really affect real world people for good or bad reasons."* Drawing from previous lessons, another student delved into the manifestation of these concepts in software, expressing, *"I never really thought about the connection between power to and power over in program or how it would be excited [executed] in program. So it was interesting to see how it works."*

3.2.2 Examining the limitations of computing as a technical fix to social problems. Our discussions continuously returned to the structural challenges. Our aim was for students to recognize and critically examine the larger structures within which technology functions [30]. By foregrounding these structural considerations, we

hoped to challenge the pervasive notion of techno-solutionism and instead emphasize the importance of collective action.

As students worked on their projects, they began to assess the limitations of computing. One student, during their work on Project 3, shared the reductionism they grappled with when having to represent complex human attributes within computational frameworks: *"I was challenged to think about how to quantize [sic] factors such as a candidate's gender and whether they are a veteran. Veteran status can easily be reduced to a binary 1 or 0, but gender/sex is a much more difficult concept to describe using integers. I was forced to resort to a simplified binary view of gender/sex which is obviously counterproductive in light of modern norms around gender identity."* Such reflections were not isolated. Pondering the scope and constraints of computing in addressing complex societal issues, a student from Project 2 observed, *"The programming experience was valuable. However, the social problem in this case is much deeper than anything that can be addressed by computing - and is in fact exacerbated by computing in the real world. This is the issue of our relationship to labor. Systemic changes are required to address problems like worker exploitation and the subsidization of wages with tips ..."*

As the semester progressed, a noticeable trend emerged: students increasingly drew connections between structural issues and their technological development. Reflecting on the ethical quandaries of devising systems like candidate filtering algorithms, a student insightfully remarked on the larger societal inequities exacerbated by unchecked technological advancements, *"I think that technology is actually something that exacerbates the inequities of the world. The tech sector enriches a small group of individuals while the rest of the world continues to languish in poverty. Technology in a way is the inevitable outcome of Enlightenment philosophy where the entire natural world became something for humans to exercise power over. The laptop I am typing this on required large amounts of raw material mined from distant parts of the world by workers subject to horrendous work conditions. Human progress and technological progress have become conflated. See the enduring struggle of the American proletariat despite record-high, tech-enabled productivity."*

3.3 Personal Relevance and Responsibilities

Similarly, students deeply considered their roles in addressing societal challenges in assignments and projects. They recognized their unique positions not just as coders, but as socially responsible individuals. Many found resonance in the course's emphasis on real-world issues. One student expressed, *"... this which gave me an opportunity to apply myself and made me look at things like a human, a programmer and not a robot designed to work mechanically. It allowed for me to think for others and address different issues."*

The students' reflections suggested their evolving understanding of their social responsibilities. One student, pondering on their Project 1 experience, shared, *"While working on the project, I learnt how to put my social ideas into implementation through programming and also got to understand the power that programmers hold at various platforms in today's society. It made me understand how*

important our ethical values are.” The emphasis on developer’s ethical values indicates our success in fostering a reflective environment. Similar several reflective discussions throughout the semester revolved around choices and the decisions to balance multiple tensions.

3.3.1 Grappling with choices and decisions. Students engaged with the nuances of equality, fairness, and equity. A student’s reflection from Project 1 captures this tension, *“I learned how to create a point system in java dependent on user answers. I also learned the weight/importance I put on my factors which can easily drown out fairness. I approached the assignment valuing equity more than fairness.”* Many shared difficulties in iterating towards their vision of a fair outcome, *“Over time, through building the algorithm, it dawned on me that more factors likely provided more fairness.”*

The projects required students to navigate multiple tensions that are inherent in any complex social problem, highlighting the significant power and responsibilities accorded to computer programmers. Reflecting on their power and their internal conflict in making decisions, a student noted, *“It was difficult for me to decide on what I would give more points to and what I would penalize in the program as I was unsure what groups I wanted to benefit the most.”*

Indeed, achieving a fair balance was cited as one of the most challenging parts of the projects. Reflecting on Project 2, a student wrote, *“I was struggling with how to divide the tips among the workers provided and I still could not distribute it between the number of workers in their own respected role.”* Some others noted that they identified many factors that were necessary to consider when allocating resources but needed to simplify it to fit in their program. Highlighting this challenge of having to make a reductive decision, a student noted, *“I faced the challenge of programming something that would evaluate everyone’s situations, and narrow it down to people getting priority over others.”*

3.4 Learning and Conceptual Integration

Throughout the semester, we made specific moves to link programming with social and relatable elements from students’ daily lives. This integrated approach was well-received by many students who saw value in merging programming with social challenges as heard in a student’s comment, *“It was meaningful to solve problems directly related to reality through programming.”* Analyzing the students’ reflections, it is evident that the integration enabled synergistic gain: programming helped deepen their understanding of the social problems, while attending to complex social problems helped in deepening their knowledge of programming.

3.4.1 Programming deepened social understanding. Asking students to breakdown the problem such that a computing system can solve it created opportunities for the students to delve deeper into the social problem. Reflecting on Project 2, a student remarked, *“I thought it was an exciting way of putting our programming knowledge into an actual world situation that many have experienced before. The ability to put it into code expanded my way of interpreting problems moving forward.”* Computing, here, was perceived as a tool to understand the social problem as shared by another student, *“I think the main thing I learned was how programming is used to look into real ethical problems, which I think is really cool.”*

Responses varied on the challenges posed by the integrated problems. One student found the blend of social with programming novel and intriguing, *“I really liked the project, I think it tweaked with my brain and incorporating social issues with programming is something new to me.”* However, another student pointed out the inherent difficulty in translating intricate social issues into code, noting, *“...the social element incorporated by theory is valid but when embedded in the program it is so hard to quantify it.”*

3.4.2 Social context helped deepen programming knowledge. Embedding social challenges within programming exercises enhanced students’ grasp of fundamental Java concepts. One student highlighted this improved understanding after Project 1, noting, *“After doing the project I learned the ‘if’ ‘else’ in Java more.”*

We structured the programming tasks to emphasize the importance of planning [32]: students first outlined their approach in English before translating it into Java. This often necessitated multiple iterations, where the students refined their solutions each time, going back and forth between the social and technical aspects of the assignment. We believe that the iterative approach helps in deepening the students’ programming abilities, a sentiment echoed by a student’s remark on Project 3, *“I learned to reconcile ideas of justice with programming concepts. I became much more confident writing methods after coding the 4 ‘Moogly’ filtering methods.”*

Many students shared the resilience and personal growth they experienced while working on the programming problems. One student reflected, *“I had some struggles but I managed to overcome them with time. ... spoke aloud on what my code was supposed to do which helped me identify the problem I encoded into my algorithm.”*

Overall, the students appreciated the socially responsible computing course. A student, looking back on the semester, commented, *“I like the class a lot and wasn’t expecting learning so much about how technology and computing can affect people in society which was a nice surprise, I thought it would be more focused on programming which it usually is most of time except the first couple weeks and the last couple weeks but yet again I enjoy learning about the ethics and power computers have in our society.”* Additionally, some students expressed a desire for a more personalized exploration of societal issues. One student suggested, *“I think it would be very fun to have a project that allows [us] to pick an issue that interest us individually.”*

4 DISCUSSION AND CONCLUSION

Reflecting on our experience, we highlight some of the challenges that we faced. We acknowledge that these challenges are not exhaustive, nor do we have a generalizable solutions for them.

Challenge 0: Building trust with and among students: In examining social complexities and problems, our pedagogical approach leaned toward fostering a communal sense of inquiry. This was rooted in our recognition of the limitations of our knowledge about social problems and the importance of students exploring the principles that guide their thinking without necessarily reaching a convergent single answer. It is imperative, thus, to nurture trust both between the educator and students and amongst the students themselves to encourage open dialogue.

Building trust is continuous, demanding consistent attention to the evolving classroom dynamics. For this, we were taking cues

from Learning for Justice, employing tactics such as urging students to contribute personal experiences, thereby enriching the discussions [22]. Reflecting upon our sessions and revisiting our lectures, we recognize lapses in fostering inclusive conversations, particularly in empowering every student to voice their perspective.

Challenge 1: Being vulnerable to engage in the discussion: Classroom discussions require students to trust each other *and* the instructor so that they can remain open to divergent perspectives [25]. Encouraging discourse with differing perspectives and ensuring that the students are engaging with perspectives different than theirs is quite challenging. “To teach is to be vulnerable” [7]; we found ourselves noting that we needed to be vulnerable in the class, to say to the students that we did not know the answers and that there is no right answer. On a reflexive note, the instructor is a computer scientist with limited education in humanities and social sciences; the positivist training he had received throughout his formal education often gave rise to doubts in his ability to teach. Many other CS instructors are in a similar position [27, 39]. Reflecting on our experience, we believe that to succeed, we need to stay with the trouble [24], hold ourselves accountable [48], and acknowledge the mistakes we make on a journey of constant learning [19].

In addition, we made moves to ensure that students were carefully evaluating their approaches and considering who their approach prioritized and ignored. We also ensured that they knew about our position, and clarified that we were sharing not to indoctrinate them with a singular belief but rather to guide them in thinking about their *own* principles. For example, students had different views on what constitutes fairness and justice when deciding on-campus housing. During the discussion, the instructor needed to facilitate by steering students back to the principles that inform the differing notions of fairness, in this case, equality versus equity.

Challenge 2: Dovetailing technical problems with social issues: By aligning programming challenges with real-world social problems, we aimed to bridge the gap between abstract technical concepts and their tangible impact on society, one that would be relatable and meaningful to the students. Our findings indicate that this synergy holds promising potential. Yet, weaving these domains introduces various ethical and practical complexities. A key concern is ensuring students can delve into and master the technical aspects within the provided timeframe while also understanding the nuances of societal issues. We attempted to strike a balance by incorporating group work and offering additional guidance, such as expected outputs. But the challenge remains.

Another challenge that we faced was figuring out how the students could demonstrate that they have attended to the social issue in sufficient depth through their programs, a challenge echoed in prior work (e.g., [4, 11, 31, 35]). In the projects, we asked students to write reflections in two places, once before building their approach and once after they have developed it. Along with this, they wrote and explained test cases that would allow us to evaluate their approach. However, we struggled to evaluate some of the submissions, requiring prolonged deliberations between the graders and the instructor. Building rubrics akin to reflective essay evaluation

would help. For future iterations, we will seek advice from instructors in liberal arts and the humanities who regularly develop such rubrics.

Challenge 3: Finding community stakeholder(s): Interactions beyond the classroom helped deepen students’ understanding of the complexities surrounding societal issues. If we want our students to learn about the responsibilities that come with the power of developing computing systems, we need to foster space where students can connect with users and stakeholders impacted by these systems. However, we struggled with an ethical challenge to do so. On one hand, we wanted students to build systems that could impact community members. On the other hand, we were worried that the engagement would not be reciprocal; the community members’ time and effort may not be justifiably paid back. We, in academia, have often been exploitative and extractive [45]. To navigate this, we encouraged students to liaise with friends and family, hoping that pre-existing bonds would provide clarity on the mutual benefit.

Challenge 4: Delineating the scope of technology-driven solutions: One significant challenge we faced was ensuring students did not overstate the capabilities of their solutions. We needed to ensure that we scoped the problem such that the students could attend to it with their newly learned programming abilities. More critically, we needed to structure the assignments such that the larger problem at hand was not made reductive, allowing for complexities that go beyond technological solutions to emerge. For example, in Project 1, we asked students to come up with an approach that is fairer than the existing one, but at the end, we asked them to reflect on who their approach advantages and who is likely to be at a disadvantage. Such reflection not only allowed us to delineate the scope of the technology-driven solution but also allowed us to show the iterative process of building technological systems.

Challenge 5: Positive Societal Impact != Social Relevance: Emphasizing social relevance requires attending to the contextual intricacies, power dynamics, and underlying systemic issues that underpin societal challenges. While highlighting the positive impacts of computing can inspire students to engage actively in addressing real-world issues through technology, there is an inherent challenge for educators to not exclusively focus on these positive aspects. Adopting a balanced view is necessary [30]. We need to incorporate critical analysis that provides a structured way to understand the broader societal impact of technology [14, 18, 28, 34, 47]. Only then it will open learning opportunities for us to acknowledge the multifaceted nature of the relationship between computing and social change. Our findings further reveal that incorporating such critical approaches can help guide students toward a more comprehensive grasp of the societal impact of computing solutions, and to understand their role and responsibilities.

Challenge 6: Avoiding Responsibilization: While the emphasis on individual responsibilities for ethical design is widespread, there is notable silence on corporate accountability. Judith Butler terms this phenomenon “responsibilization” [8]. For a truly fair society, the focus must shift from responsibilization to making larger entities accountable, achieved through collective action and political change [14, 28]. This redirection from undue responsibilization is even more pressing given that many of our students are from lower-SES backgrounds and tech jobs are often presented as the

only pathway to economic mobility. In teaching socially responsible computing, a challenge lies in ensuring students grasp both the importance as well as limitations of individual responsibilities. In our course, we discussed individual responsibilities, the power developers have in refusing to build or in repairing harm, conflicts arising from such choices, and the power and responsibilities accorded to larger structures. We wish we had more opportunities to discuss these issues in detail.

In conclusion, our future computer scientists must be technically proficient *and* socially and ethically conscious, becoming justice-seeking citizens. Our students' feedback, that the weaving of social and technical motivated and empowered them, is heartening and points towards a promising direction. We hope that the insights shared in this article offer initial considerations for educators.

ACKNOWLEDGMENTS

This work is supported by NSF award 2216575 and a grant from Northeastern University's Center for Inclusive Computing.

REFERENCES

- [1] Amy Allen. 2022. Feminist Perspectives on Power. In *The Stanford Encyclopedia of Philosophy* (Fall 2022 ed.), Edward N. Zalta and Uri Nodelman (Eds.). Metaphysics Research Lab, Stanford University.
- [2] Kathryn L Boucher, Melissa A Fuesting, Amanda B Diekman, and Mary C Murphy. 2017. Can I work with and help others in this field? How communal goals influence interest and participation in STEM fields. *Frontiers in psychology* 8 (2017), 901.
- [3] Bo Brinkman and Amanda Diekman. 2016. Applying the communal goal congruity perspective to enhance diversity and inclusion in undergraduate computing degrees. In *Proceedings of the 47th ACM technical symposium on computing science education*. 102–107.
- [4] Carla E Brodley, Benjamin J Hescott, Jessica Biron, Ali Rensing, Melissa Peiken, Sarah Maravetz, and Alan Mislove. 2022. Broadening participation in computing via ubiquitous combined majors (CS+ X). In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*. 544–550.
- [5] Noelle Brown, Koriann South, and Eliane S Wiese. 2022. The Shortest Path to Ethics in AI: An Integrated Assignment Where Human Concerns Guide Technical Decisions. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*. 344–355.
- [6] Michael Buckley, Helene Kershner, Kris Schindler, Carl Alphonse, and Jennifer Braswell. 2004. Benefits of using socially-relevant projects in computer science and engineering education. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*. 482–486.
- [7] Robert V Bullough Jr. 2005. Teacher vulnerability and teachability: A case study of a mentor and two interns. *Teacher Education Quarterly* (2005), 23–39.
- [8] Judith Butler. 2016. *Frames of war: When is life grievable?* Verso Books.
- [9] Mary Elaine Califf and Mary Goodwin. 2005. Effective incorporation of ethics into courses that focus on programming. *ACM SIGCSE Bulletin* 37, 1 (2005).
- [10] Francisco Castro, Sahitya Raipura, Heather Conboy, Peter Haas, Leon Osterweil, and Ivon Arroyo. 2023. Piloting an Interactive Ethics and Responsible Computing Learning Environment in Undergraduate CS Courses. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 659–665.
- [11] Lena Cohen, Heila Precel, Harold Triedman, and Kathi Fisler. 2021. A new model for weaving responsible computing into courses across the CS curriculum. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*.
- [12] Randy W Connolly. 2011. Beyond good and evil impacts: Rethinking the social issues components in our computing curricula. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*.
- [13] Christianne Corbett and Catherine Hill. 2015. *Solving the Equation: The Variables for Women's Success in Engineering and Computing*. ERIC.
- [14] Sasha Costanza-Chock. 2020. *Design justice: Community-led practices to build the worlds we need*. The MIT Press.
- [15] National Research Council et al. 2013. Nonresponse in social science surveys: A research agenda. (2013).
- [16] Victoria Dean and Illah Nourbakhsh. 2022. Teaching Ethics by Teaching Ethics Pedagogy: A Proposal for Structural Ethics Intervention. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*. 272–278.
- [17] Amanda B Diekman, Elizabeth R Brown, Amanda M Johnston, and Emily K Clark. 2010. Seeking congruity between goals and roles: A new look at why women opt out of science, technology, engineering, and mathematics careers. *Psychological science* 21, 8 (2010), 1051–1057.
- [18] Virginia Eubanks. 2018. *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin's Press.
- [19] Bre Evans-Santiago. 2020. *Mistakes we have made: Implications for social justice educators*. Myers Education Press.
- [20] Casey Fiesler, Mikhaila Friske, Natalie Garrett, Felix Muzny, Jessie J Smith, and Jason Zietz. 2021. Integrating ethics into introductory programming classes. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*.
- [21] Casey Fiesler, Natalie Garrett, and Nathan Beard. 2020. What do we teach when we teach tech ethics? A syllabi analysis. In *Proceedings of the 51st ACM technical symposium on computer science education*. 289–295.
- [22] Learning for Justice. [n.d.]. Community Inquiry. <https://www.learningforjustice.org/classroom-resources/teaching-strategies/community-inquiry>. Accessed: July 26, 2023.
- [23] Mikey Goldweber, Renzo Davoli, Joyce Currie Little, Charles Riedesel, Henry Walker, Gerry Cross, and Brian R Von Konsky. 2011. Enhancing the social issues components in our computing curriculum: computing for the social good. *ACM Inroads* 2, 1 (2011), 64–82.
- [24] Donna J Haraway. 2016. *Staying with the trouble: Making kin in the Chthulucene*. Duke University Press.
- [25] Annette Henry. 1994. There are no safe places: Pedagogy as powerful and dangerous terrain. *Action in Teacher Education* 15, 4 (1994), 1–4.
- [26] Diane Horton, David Liu, Sheila A McIlraith, and Nina Wang. 2023. Is More Better When Embedding Ethics in CS Courses?. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 652–658.
- [27] Deborah Johnson. 1994. Who should teach computer ethics and computers & society? *Acem Sigcas Computers and Society* 24, 2 (1994), 6–13.
- [28] Mawera Karetai, Samuel Mann, Dhammika Dave Guruge, Sherlock Licorish, and Alison Clear. 2023. Decolonising Computer Science Education-A Global Perspective. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V.1*. 1097–1102.
- [29] Natalie Kiesler and Carsten Thorbrügge. 2023. Socially responsible programming in computing education and expectations in the profession. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V.1*.
- [30] Amy J Ko, Alannah Oleson, Neil Ryan, Yim Register, Benjamin Xie, Mina Tari, Matthew Davidson, Stefania Druga, and Dastyni Loksa. 2020. It is time for more critical CS education. *Commun. ACM* 63, 11 (2020), 31–33.
- [31] Matthew Kopeck, Meica Magnani, Vance Ricks, Roben Torosyan, John Basl, Nicholas Miklaucic, Felix Muzny, Ronald Sandler, Christo Wilson, Adam Wisniewski-Jensen, et al. 2023. The effectiveness of embedded values analysis modules in Computer Science education: An empirical study. *Big Data & Society* 10, 1 (2023), 20539517231176230.
- [32] Dastyni Loksa, Amy J Ko, Will Jernigan, Alannah Oleson, Christopher J Mendez, and Margaret M Burnett. 2016. Programming, problem solving, and self-awareness: Effects of explicit guidance. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 1449–1461.
- [33] C Dianne Martin and Elaine Yale Weltz. 1999. From awareness to action: Integrating ethics and social responsibility into the computer science curriculum. *ACM Sigcas Computers and Society* 29, 2 (1999), 6–14.
- [34] Gabriel Medina-Kim. 2021. Towards Justice in Undergraduate Computer Science Education: Possibilities in Power, Equity, and Praxis. In *2021 ASEE Virtual Annual Conference Content Access*.
- [35] Barbara Moskal, Keith Miller, and LA Smith King. 2002. Grading essays in computer ethics: rubrics considered helpful. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*. 101–105.
- [36] Victor Paul Pauca and Richard T Guy. 2012. Mobile apps for the greater good: a socially relevant approach to software engineering. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. 535–540.
- [37] Evan Peck. 2017. The Ethical Engine: Integrating Ethical Design into Intro Computer Science. *blog, Bucknell HCI* 5 (2017).
- [38] Nichole Pinkard, Caitlin Kennedy Martin, and Sheena Erete. 2020. Equitable approaches: Opportunities for computational thinking with emphasis on creative production and connections to community. *Interactive Learning Environments* 28, 3 (2020), 347–361.
- [39] Michael J Quinn. 2006. On teaching computer ethics within a computer science department. *Science and Engineering Ethics* 12 (2006), 335–343.
- [40] Jean J Ryoo, Alicia Morris, and Jane Margolis. 2021. “What happens to the Rastapado man in a cash-free society?”: Teaching and learning socially responsible computing. *ACM Transactions on Computing Education (TOCE)* 21, 4 (2021), 1–28.
- [41] Johnny Saldaña. 2015. *The coding manual for qualitative researchers*. Sage.
- [42] Jeffrey Saltz, Michael Skirpan, Casey Fiesler, Micha Gorelick, Tom Yeh, Robert Heckman, Neil Dewar, and Nathan Beard. 2019. Integrating ethics within machine learning courses. *ACM Transactions on Computing Education (TOCE)* 19, 4 (2019).

- [43] Jessie J Smith, Blakeley H Payne, Shamika Klassen, Dylan Thomas Doyle, and Casey Fiesler. 2023. Incorporating Ethics in Computing Courses: Barriers, Support, and Perspectives from Educators. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 367–373.
- [44] Kylan Stewart, Bruce Debruhl, and Zoe Wood. 2022. An Equity-minded Assessment of Belonging among Computing Students. In *2022 ASEE Annual Conference & Exposition*.
- [45] Randy Stoecker and Elizabeth A Tryon. 2009. *The unheard voices: Community organizations and service learning*. Temple University Press.
- [46] Allen B. Tucker, Bruce H. Barnes, Robert M. Aiken, Keith Barker, J. Bruce, Kim B. Thomas Cain, Susan E. Conry, Gerald L. Engel, Richard G. Epstein, Doris K. Lidtke, Michael C. Mulder, Jean B. Rogers, Eugene H. Spafford, and A. Joe Turner. 1991. Report of the ACM/IEEE-CS Joint Curriculum Task Force. *Association for Computing Machinery* (1991).
- [47] Sepehr Vakil. 2018. Ethics, identity, and political vision: Toward a justice-centered approach to equity in computer science education. *Harvard educational review* 88, 1 (2018), 26–52.
- [48] Alicia Nicki Washington. 2020. When twice as good isn't enough: The case for cultural competence in computing. In *Proceedings of the 51st ACM technical symposium on computer science education*. 213–219.
- [49] Laurie H Werth. 1997. Getting started with computer ethics. In *Proceedings of the Twenty-eighth SIGCSE technical symposium on Computer Science Education*. 1–5.
- [50] Aman Yadav and Marie K Heath. 2022. Breaking the code: Confronting racism in computer science through community, criticality, and citizenship. *TechTrends* 66, 3 (2022), 450–458.
- [51] Cass Zegura, Ben Rydal Shapiro, Robert MacDonald, Jason Borenstein, and Ellen Zegura. 2023. “Moment to Moment”: A Situated View of Teaching Ethics from the Perspective of Computing Ethics Teaching Assistants. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–15.