Five Pedagogical Principles of a User-Centered Design Course that Prepares Computing Undergraduates for Industry Jobs

Sean Kross
UC San Diego
seankross@ucsd.edu

Philip J. Guo
UC San Diego
pg@ucsd.edu

ABSTRACT

We present a new user-centered design course that prepares computing undergraduates for software industry jobs such as UI/UX designer, product designer, and product manager. Our course aims to bridge the academia-industry gap and innovates upon prior published HCI courses due to its targeted focus on job preparation, inclusion, and scale. Nearly 200 students (55% women) have taken it in the past two years. We developed its curriculum to align with the needs of modern industry employers and implemented five theory-backed pedagogical principles: 1) industry-relevant project prompts developed in consultation with recent course alumni, 2) final project deliverable optimized for job-seeking, 3) no coding required to foster inclusion, 4) low-stress effort-based grading to further foster inclusion, 5) weekly feedback and chances for revisions. We discuss the theoretical rationale behind these five principles and how instructors can potentially apply them to a broad range of project-based courses across many areas of computing.

CCS CONCEPTS

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

KEYWORDS

HCI education, UX design, project-based course, job preparation

ACM Reference Format:
Sean Kross and Philip J. Guo. 2022. Five Pedagogical Principles of a User-Centered Design Course that Prepares Computing Undergraduates for Industry Jobs. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022), March 3–5, 2022, Providence, RI, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3478431.3499341

1 INTRODUCTION

User-centered design is a product development process that involves observing and interviewing users to discover their actual needs, creating a series of prototypes, and performing usability testing to refine those prototypes on the path toward a finished product [10, 11, 26]. Although this methodology applies to many fields, it is especially prevalent in computing: Every widely-used piece of software today was likely developed with a user-centered design process to optimize for satisfying real user needs [18].

Despite the importance of user-centered design in the software industry, university students rarely get a chance to learn it. Within CS departments it is usually taught in HCI electives [17], but those courses traditionally focus on theoretical design principles [7, 33]. Project-based HCI courses provide much-needed hands-on experience [3, 10, 14, 31], but from surveying prior work (see Section 2) we found that *they lack explicit support for helping students to obtain relevant jobs*, which limits their perceived utility in students' eyes.

To address this limitation, we created a new user-centered design course explicitly focused on preparing computing undergraduates for technology industry jobs such as UI/UX designer, product designer, and product manager. To our knowledge, this paper is the first to describe this type of job-focused university HCI course that aims to bridge the academia-industry gap [29, 30] (see Section 2). We teach at a large public U.S. university (UC San Diego), so scale was critical: Whereas project-based courses at design schools are taught in studios with 10–20 students [1, 2], we had to create a course that serves an order of magnitude more students. Nearly 200 students (55% women) have taken our course in its first two years.

This experience report contributes to the computing education literature by presenting five pedagogical principles from our course that instructors can adapt into their own computing project courses:

(1) **Industry-relevant project prompts**: UI/UX design is a fast-changing field, so each year we use feedback from course alumni working in industry to make project prompts that are the most timely and relevant for current industry jobs.
(2) **Final deliverable optimized for job-seeking**: After finishing the project, everyone creates a final deliverable, which is a *design case study* in an industry-standard format that hiring managers are looking for. Our students have gotten jobs as UI/UX designers and product managers in software companies by presenting these case studies to employers.
(3) **No coding required**: To lower participation barriers and make the course more inclusive, we do *not* require any coding. Instead students use industry-standard prototyping tools (e.g., Figma), which are much easier for novices to learn.
(4) **Effort-based grading**: Again to make the course more inclusive, we implemented an effort-based grading policy [16] where students get full credit for simply completing each assignment, regardless of the subjective 'quality' of their design work. This way, students coming in with less prior design experience are not put at a disadvantage grade-wise.
(5) **Weekly feedback and chances for revisions**: Students receive feedback from classmates and TAs each week and then get several chances to revise their prior work in response to such feedback. Since grading is based on effort (see above), our TAs feel very comfortable giving students highly-critical feedback without it harming their grades.

2 RELATED WORK

In the early 2000s, SIGCSE held several panel discussions on HCI education [4, 8], and researchers followed-up with experience reports and surveys to document techniques for teaching HCI [3, 7, 14, 22, 25, 31, 33]. These courses varied widely in content, but two common threads emerged: 1) having students implement a team project, 2) offering regular feedback and critiques in small-group studio sessions. Our course follows this standard template but innovates beyond prior courses in its focus on providing industry-relevant project prompts (Section 4.1) and having students create a final deliverable (a case study) that they can present to employers (Section 4.2).

Aberg [3] and Hui [14] surveyed instructors about challenges in teaching HCI courses, and Oleson et al. [24] complemented those findings with student perspectives. One recurring theme was that students perceive the content in such courses to be too abstract, vague, and sometimes 'obvious' or 'common sense.' As a result, they felt that grading tended to be too subjective and unfair, which our course remedies with its effort-based grading policy (Section 4.4) and weekly opportunities for feedback and revision (Section 4.5).

Another recurring issue is that project deliverables are usually prototypes and reports that students cannot put to good use after the class ends. Aberg recounted that *"as for the report of the project work, there was a sense of an unclear aim"* [3] so students were unmotivated to write up their findings. He suggested *"to get around the problems with lack of motivation one could try to connect it better to the industry."* [3] Our course provides such motivation via industry-relevant project prompts (Section 4.1) and a project report formatted as a design case study that students can immediately put to use by presenting it to employers when job hunting (Section 4.2).

Despite the long lineage of work on HCI education, to our knowledge *there have been no papers documenting HCI courses that are designed with industry-relevant job preparation as a core goal.* Thus, we contribute to this academic literature by describing a novel HCI course that is designed to optimize for job preparation, inclusion, and scale to serve a large public university. More broadly, our course aims to bridge the academia-industry gap in computing education, which has been a recurring discussion in our community [29, 30].

3 COURSE OVERVIEW

Our HCI course teaches students to *apply a user-centered design process to create a computing-related design project that they write up as a case study in an industry-standard format.* The case study they create in our course can be used to directly apply for jobs in the software industry such as UI/UX designer, product designer, and product manager (PM). It can also augment the portfolios of students who are applying for more traditional software engineering and programming jobs. See Section 4.2 for details about what a modern industry-standard design case study looks like (circa 2022).

Student population: We teach at a large public U.S. university containing a large computer science department (over 2,000 majors) and over 2,000 computing-related students in several adjacent departments (e.g., cognitive science, electrical and computer engineering). Our students mostly come from those majors. This is a new elective course that has been offered twice, with 82 students in Fall 2019 (48% women) and 112 students in Fall 2020 (61% women).

Week	Main Concepts	Project Milestone Due
1	Intro. to user-centered design	(none)
2	User research for needfinding	project proposal, user research plan
3	Giving design critiques	user research findings
4	Sketching possible solutions	user personas, competitive audit, UX flows, UI sketches
5	Low-fidelity prototyping, usability testing	low-fi digital wireframes, initial user testing
6	(none, time for revisions)	revisions based on feedback
7	High-fidelity prototyping	hi-fi prototype, user testing
8	Reaching out to employers	refine prototype from testing
9	Writing up a case study (1)	revisions based on feedback
10	Writing up a case study (2)	case study on live website

Table 1: Our user-centered design course for computing students, with project milestones due at the end of each week.

Course calendar: Table 1 shows the flow of our course over a standard 10-week term at our university. We cover all major steps of a user-centered design process [10, 17, 26] ranging from user research to UX flows to UI prototyping to user testing. There are no theory-based homework assignments or exams; all grades come from a term-long project with milestones to turn in each week.

Project milestones: Students work either individually or on a small team to complete a term-long design project. The right column of Table 1 shows how each week they must turn in a milestone that corresponds to the step of the user-centered design process they are implementing that week. They receive feedback from both their classmates and TAs about each milestone. The final milestone at the end of Week 10 is their completed case study (Section 4.2).

Undergraduate TAs: We mostly use undergraduate TAs since they can relate better to our undergraduate students, have recently taken the same course or similar courses, and have done recent internships in UI/UX/product design roles. The pervasive use of undergraduate TAs in university computing courses has been well-documented [21, 23], and we extend that tradition to HCI. We invite top-performing students to become TAs for next term, which helps ensure a sustainable pipeline that scales as enrollment grows.

4 FIVE PEDAGOGICAL PRINCIPLES

The UI/UX design topics we cover (Table 1) are fairly standard, but what makes our course unique is a set of five principles that we developed to optimize for job preparation, inclusion, and scale.

4.1 Industry-relevant project prompts

A central challenge of project-based courses is what constraints to place on students' project ideas. At one extreme, instructors could allow students to do projects on whatever they want; this maximizes freedom but often results in bland projects because, in practice, many undergrads are not able to come up with interesting ideas. They often end up creating cliche 'college student apps' such as class selection tools, todo lists, or apps to facilitate on-campus socializing. Thus, some instructors create their own project prompts to provide constraints, but the problem here is that those prompts are not necessarily what appeal to industry employers.

To overcome this limitation, instead of us as instructors coming up with arbitrary prompts ourselves, we surveyed the top-performing students from prior years (some of whom are our current TAs) to have them come up with prompts related to the latest design themes and trends that they have seen recently in the software industry[1]. Since these top students often do internships at relevant companies and then go on to full-time jobs there after graduating, they are much more in touch with industry trends.

The exact prompt details vary by year since industry trends evolve over time, but they all follow this general template:

> **Extend or redesign a feature of a widely-used [type of software application] to help a [specific type of user] to [do some important task better].**

Critically, students must extend or redesign a *feature* of an existing widely-used app (e.g., a mobile app, web app, or desktop app that many people use) rather than creating their own from scratch.

Theoretical rationale: We use industry-relevant project prompts because of the motivating power of *authenticity*. This is a major theme of Guzdial's 2015 book on computing education theory and practice [12]. To summarize, students want to feel like they are making 'the real thing' with industry-standard practices and not just doing 'toy projects.' Guzdial relates authenticity to expectancy-value theory [32], which states that students are more motivated when they see genuine value in what they are assigned to do.

Potential benefits: Our prompt template has three benefits:

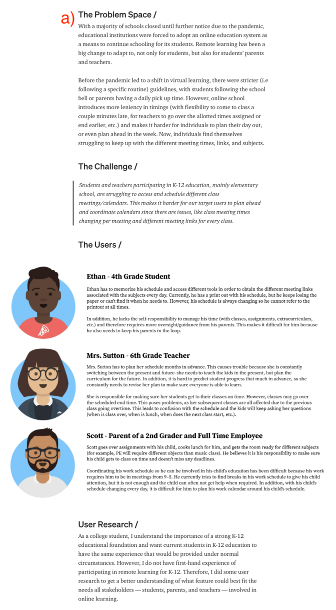
- It makes for more compelling job applications since it is a more realistic simulation of what new hires do in industry. Employers want to see how students can work within the constraints of existing apps rather than making their own app from scratch (which employees rarely do in industry).
- It forces students to work within the constraints of widely-used popular apps and to solve problems faced by real people. This prevents them from making up cliche 'toy' project ideas that look amateurish when presented to potential employers.
- Honing in on a specific feature provides focus and eliminates overly-vague goals like *"I want to redesign Skype and Zoom!"*

Each term the course staff creates around half a dozen prompts to account for a diverse variety of student interests as well as recent industry design trends. For instance, in the most recent offering (Fall 2020) we created a set of timely prompts related to improving quality of life during the COVID-19 pandemic, such as facilitating K-12 education, helping extended families remain connected, and supporting local businesses during the pandemic. Here is one example prompt: *Extend or redesign a feature of a widely-used online education tool to help K-12 teachers to [do some important task better].*

To give a sense of what students worked on, here are some examples of the most outstanding projects from the Fall 2020 term:

- adding features to Zoom and Google Calendar to help K-12 teachers to better manage their virtual classrooms

[1]Some courses connect students with real industry clients, which is even more realistic but is *very hard* to scale since the staff needs to manage those connections [10, 14, 31].


a) The Problem Space /

With a majority of schools closed until further notice due to the pandemic, educational institutions were forced to adopt an online education system as a means to continue schooling for its students. Remote learning has been a big change to adapt to, not only for students, but also for students' parents and teachers.

Before the pandemic led to a shift in virtual learning, there were stricter (i.e following a specific routine) guidelines, with students following the school bell or parents having a daily pick up time. However, online school introduces more leniency in timings (with flexibility to come to class a couple minutes late, for teachers to go over the allotted times assigned or end earlier, etc.) and makes it harder for individuals to plan their day out, or even plan ahead in the week. Now, individuals find themselves struggling to keep up with the different meeting times, links, and subjects.

The Challenge /

Students and teachers participating in K-12 education, mainly elementary school, are struggling to access and schedule different class meetings/calendars. This makes it harder for our target users to plan ahead and coordinate calendars since there are issues, like class meeting times changing per meeting and different meeting links for every class.

The Users /

Ethan - 4th Grade Student

Ethan has to memorize his schedule and access different tools in order to obtain the different meeting links associated with the subjects every day. Currently, he has a print out with his schedule, but he keeps losing the paper or can't find it when he needs to. However, his schedule is always changing so he cannot refer to the printout at all times.

In addition, he lacks the self-responsibility to manage his time (with classes, assignments, extracurriculars, etc.) and therefore requires more oversight/guidance from his parents. This makes it difficult for him because he also needs to keep his parents in the loop.

Mrs. Sutton - 6th Grade Teacher

Mrs. Sutton has to plan her schedule months in advance. This causes trouble because she is constantly switching between the present and future-she needs to teach the kids in the present, but plan the curriculum for the future. In addition, it is hard to predict student progress that much in advance, so she constantly needs to revise her plan to make sure everyone is able to learn.

She is responsible for making sure her students get to their classes on time. However, classes may go over the scheduled end time. This poses problems, as her subsequent classes are all affected due to the previous class going overtime. This leads to confusion with the schedule and the kids will keep asking her questions (when is class over, when is lunch, when does the next class start, etc.).

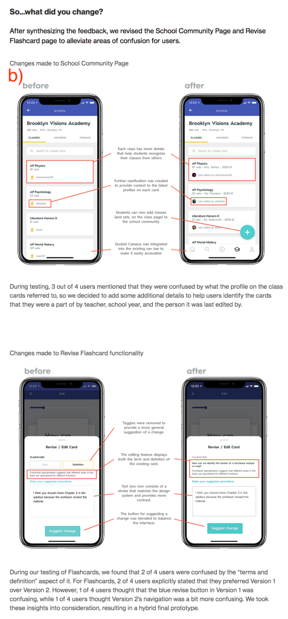
Scott - Parent of a 2nd Grader and Full Time Employee

Scott goes over assignments with his child, cooks lunch for him, and gets the room ready for different subjects (for example, PE will require different objects than music class). He believes it is his responsiblity to make sure his child gets to class on time and doesn't miss any deadlines.

Coordinating his work schedule so he can be involved in his child's education has been difficult because his work requires him to be in meetings from 9–5. He currently tries to find breaks in his work schedule to give his child attention, but it is not enough and the child can often not get help when required. In addition, with his child's schedule changing every day, it is difficult for him to plan his work calendar around his child's schedule.

User Research /

As a college student, I understand the importance of a strong K-12 educational foundation and want current students in K-12 education to have the same experience that would be provided under normal circumstances. However, I do not have first-hand experience of participating in remote learning for K-12. Therefore, I did some user research to get a better understanding of what feature could best fit the needs all stakeholders — students, parents, and teachers — involved in online learning.


So...what did you change?

After synthesizing the feedback, we revised the School Community Page and Revise Flashcard page to alleviate areas of confusion for users.

Changes made to School Community Page

b)

before
after
12:22
SCHOOL
Brooklyn Visions Academy
300 sets | BVA - Brooklyn, NY
CLASSES
MEMBERS
FORUMS
Search for classes here
AP Physics
87 sets · Mrs. DeVoe · 2020-21
Last edited by milesmorales99
AP Psychology
53 sets · Ms. Picciano · 2020-21
Last edited by stellastar
Literature Honors II
31 sets · Mr. Stallworth · 2020-21
AP World History
129 sets

Each class has more details that help students recognize their classes from others

Further clarification was created to provide context to the listed profiles on each card

Students can now add classes (and sets, on the class page) to the school community

Quizlet Campus was integrated into the existing nav bar to make it easily accessible

During testing, 3 out of 4 users mentioned that they were confused by what the profile on the class cards referred to, so we decided to add some additional details to help users identify the cards that they were a part of by teacher, school year, and the person it was last edited by.

Changes made to Revise Flashcard functionality

How can we identify the border of a functional module or map?

Functional specialization suggests that different areas in the brain are specialized for different functions.

I think you should check Chapter 3 in the syllabus because the professor revised this material.

Toggles were removed to provide a more general suggestion of a change.

The editing feature displays both the term and definiton of the existing card.

Text box now consists of a stroke that matches the design system and provides more contrast.

The button for suggesting a change was elevated to balance the interface.

During our testing of Flashcards, we found that 2 of 4 users were confused by the "terms and definition" aspect of it. For Flashcards, 2 of 4 users explicitly stated that they preferred Version 1 over Version 2. However, 1 of 4 users thought that the blue revise button in Version 1 was confusing, while 1 of 4 users thought Version 2's navigation was a bit more confusing. We took these insights into consideration, resulting in a hybrid final prototype.

Figure 1: Excerpts from two case study webpages created by students in our course (full case studies too long to fit here). a) problem motivation and user personas for extending Google Calendar for K-12 education. b) user-testing and refining an extension of the Quizlet online education app.

- enhancing popular mobile apps to coordinate access to health-care services while minimizing risks of venturing outside
- redesigning FaceTime to better connect older adults with family members to keep emotional ties when quarantined

4.2 Final deliverable optimized for job-seeking

Many project-based courses provide the raw materials for students to present in their portfolio when job-hunting, but they suffer from the last-mile problem [27]: these courses never teach students how to write up their project work in a format that appeals to employers. Instead, students are left to figure this all out by themselves. As a result, even though they have put in all the work to complete a project, many find it hard to traverse that vital "last mile" to turn it into a compelling addition to their portfolio that employers value.[2]

We address this last-mile problem by explicitly teaching students how to write up a *design case study* and upload it to a real portfolio website as the final deliverable of our course. A case study is a blog-post-like webpage that shows how a student adopted a user-centered design process from the start of a project all the way to the finished product; it explains the analytical reasoning behind all the various design decisions made along the way. Figure 1 shows two excerpts from case studies made by our students. The exact details are unimportant, but these case studies mix explanatory text with images, mock-up app screenshots, and sometimes animated GIFs or embedded videos that demonstrate user interactions.

[2]The last-mile problem [27] is a term from communications, infrastructure, and logistics design that conveys the difficulty of connecting end-to-end. For instance, a subway station may be located over a mile away from many commuter's homes or offices, so they also need to carry bicycles or skateboards with them to cover that last mile.

Employers have told us that *these case studies are the most important part of a student's portfolio* when they apply to jobs such as UI/UX designer or product manager. They read case studies to get a sense of applicants' skills as an analytical and design-oriented thinker. This is why in our course we teach students to write a case study on their project in an industry-standard format that appeals to employers. Specifically, we guide students to write up the 'behind-the-scenes' details of their design process, justifications, and rationale so they are not simply showing the final product. Our curriculum is available at *[link anonymized for submission]*, and we have adapted content from industry guides for case studies [28].

Theoretical rationale: Similar to industry-relevant project prompts (Section 4.1), we guide students to create real case studies because of the motivating power of authenticity [12]. According to expectancy-value theory [32], we believe that students are more motivated when the final product they create in our course is a case study that looks like those that have helped their peers to get industry jobs.

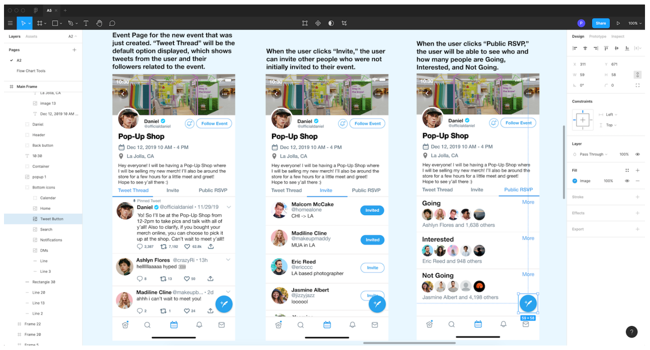
Potential benefits: Several students have already emailed us to report that they directly used the case study from our course to obtain their first design-related internship in the software industry.

In addition, although some tech-savvy students already know how to create case studies and portfolio webpages, many computing students who come from less privileged backgrounds have not done so before. Thus, by teaching them how to do this step-by-step in our course, we hope to ***improve equity and inclusion*** in our field. Here is a relevant anecdote from an end-of-term course review survey that we conducted (students consented to having their anonymized feedback potentially appear in academic publications):

> *"The best part is I finally made a portfolio/website for myself to showcase my project (and hopefully future projects) to potential employers. Even though I've always heard that a portfolio was one of the most valuable things for a college student, it was something that I never thought I would ever do. This was due to lack of knowledge and lack of projects that I felt were worthy enough to display. Now, instead of skipping the section for 'Personal Website' on job applications, I can confidently lead them to my website to showcase my abilities."*

4.3 No coding required

Students often design mobile or web applications in their project-based computing courses, so it seems natural to have them also implement their designs in code. We have taught several such courses in the past and one author, in his role as the undergraduate education chair in his department, has spoken with many students about their perceptions of these programming-heavy courses. We found that while such coding-intensive courses give students the chance to implement real software, *the complexities of coding overshadow the higher-order lessons we want to teach them about design.* What often happens is that students spend an enormous amount of time on the myriad complexities of setting up, debugging, and interfacing with libraries to implement modern mobile or web applications. And due to the short time-span of such courses, students often end up with a final product that is buggy and unpolished, which does not make for a compelling portfolio piece to show employers.



Event Page for the new event that was just created. "Tweet Thread" will be the default option displayed, which shows tweets from the user and their followers related to the event.

When the user clicks "Invite," the user can invite other people who were not initially invited to their event.

When the user clicks "Public RSVP," the user will be able to see who and how many people are Going, Interested, and Not Going.

Figure 2: Instead of implementing their projects with code, students in our course use industry-standard UI prototyping tools such as Figma. Here a student is using Figma to prototype an extension to the Twitter mobile app for iOS.

Thus, in our course we do not require any coding. Instead, students create prototypes of mobile or web apps using industry-standard UI prototyping tools such as Figma. These are GUI tools that UI/UX designers in industry use to prototype the look-and-feel of user interfaces along with user interactions and animations, all without needing to write any code. Figure 2 shows how students use Figma to prototype an extension to Twitter's iOS app UI by modifying pre-made iOS UI components; the resulting prototype looks similar to a real app and can even have simple interactivity.

This decision also aligns well with what employers are looking for since when students apply for jobs related to UI/UX/product design, they are *not* required to code; instead, employers read their case studies to assess their skills in reasoning about the user-centered design process. They understand that the app screenshots in case studies are mock-ups created using tools such as Figma (e.g., Figure 2) and thus do not expect to see a fully-functional app.

Theoretical rationale: Aside from the authenticity benefits of non-code prototypes being what industry employers expect when hiring designers (who are evaluated differently from software engineers), not requiring coding also has benefits for inclusion. Prior work has shown that one way to make courses more inclusive in terms of gender and other demographics is to not equate computing with geek stereotypes of pure coding prowess [20]. As one data point, our enrollments so far have been 55% women (200 total students), which is much higher than our CS department baseline of 25%.

Potential benefits: Prototyping tools like Figma lower barriers to entry and enable students to create much more polished prototypes than they would be able to do in code. They also promote inclusion by not putting students with less prior coding experience at a disadvantage. Here is a course review from a student who self-identified as a novice programmer:

> *"I especially liked how this class did not require coding as that is the one thing in life that I am 100% completely horrible at like [getting] 24% on programming assignments type of horrible. I like how I got to focus on the research and designing and using the prototyping tools that came with Figma."*

Even experienced programmers can benefit from *not* coding in our course since it lets them focus more on high-level design ideas rather than low-level code details. Here is a relevant student review:

> *"This course forces you to think about how effective an application is at achieving its goal. As a computer science student, this juxtaposed with the content I am used to hearing and learning, where an application's main goal is to have properly working algorithms and provide the correct output. Oftentimes, I believe people overlook the importance of an application's control flow and user interface."*

4.4 Effort-based grading

From our experiences teaching many software engineering and HCI project-based courses in the past, one perennial challenge we face is how to do grading in a fair and efficient way. One traditional way has been to use rubrics [19], which are checklists of requirements for each project component. However, some students perceive rubrics to be too restrictive on their creativity, since they must resort to 'checking off the boxes' to earn the respective points; likewise, our TAs have found grading with rubrics to be very time-consuming since they must tediously validate whether each students' submission meets each rubric item. We would also like to assess the 'quality' of student projects, but doing so is fraught with bias since each TA has different tastes as to what makes for a 'good' project. Unlike in, say, a mathematics course, there is often not a 'right' or 'wrong' answer in open-ended design projects [14].

Our theory-backed solution to these issues is to implement *effort-based grading* [16]: students get full credit on each weekly project milestone if they put in the effort to turn it in on time (we allow a few late days for emergencies). Each week students can get 2 points: 1 for turning in the milestone and 1 for giving peer feedback to classmates on the prior milestone (see Section 4.5). The only ways to get a 0 score are either not to turn anything in or to turn in something meaningless (e.g., writing a random jumble of words). This means students who put in the *effort* to complete all project milestones will earn 100%, regardless of the 'quality' of their work. Percentages translate into letter grades using a standard A–F scale.

Theoretical rationale: Our policy is directly inspired by the progressive concept of labor-based grading contracts from humanities classes [16]. The theoretical justification there is one of equity: it is hard to be 'objective' in judging student work in the humanities, especially when students come from diverse cultural and socioeconomic backgrounds. Thus, the developers of this method argue that the most equitable and ethical way to assign grades is by whether a student has put in the *effort* of labor to complete each assignment.

Potential benefits: Although readers may be skeptical of this grading policy, we have gotten tremendous benefits from it. For instance:

- It mostly eliminates subjectivity and bias in grading, since TAs are not assigning grades based on perceived quality. (There is still some subjectivity in determining whether a student has 'completed' a milestone, but that is much easier to agree on than judgments of quality. To calibrate, other TAs and the instructor will take a look at borderline cases.)
- TAs spend very little time on grading and can devote the majority of their time to mentoring students' projects and giving them honest feedback. This policy empowers them to give candid, constructive, and even highly-critical feedback to students without worrying that it may hurt their grades.
- Students do not need to follow a rubric to get credit, which gives them more creative freedom. From a student review: *"I also liked that the class gave you points as long as you clearly put in effort. It took away the stress of trying to make the case study fit a certain mold rather than make it our own."*
- It fosters inclusion by not putting those with less design skills at a disadvantage. From a student review: *"The grading scale for this class was really generous because it relieved a lot of the stress and worries I had coming into this class because I've never done a case study this complicated before."*
- Finally, having TAs not spend an inordinate amount of time on weekly grading has been a very effective way for us to scale to larger enrollments without overburdening our TAs.

Informal assessment: Why would students put in the effort to do a good job when they can still get an A grade with minimal effort? We encouraged students to put in their best effort by reminding them that they are working toward making a real case study that can land them real jobs; we presented direct evidence of their TAs and other former students getting jobs using these sorts of case studies, so that often provided sufficient motivation.

As a preliminary assessment, after the Fall 2020 term ended, TAs looked over all 112 students' final case studies and judged their quality (this was not part of their grade). They determined that 21 case studies were top-notch, with 9 more earning 'honorable mention.' That means 30 out of 112 students (27% of the class) did outstanding work even without grades as a motivator. On the flip side, our TAs found only 7 case studies that were very sub-par in quality. This is *not* a formal assessment by any means, but at least it gives us some indication that the majority of students were still motivated to put in effort despite not being graded.

4.5 Weekly feedback and chances for revisions

TAs give feedback to students each week both synchronously during Friday studio sections [15] (each TA has around 20 students) and asynchronously via comments left on their milestone Google Docs.

In addition, when students turn in each weekly milestone, they must also give feedback to two of their classmates about their prior milestone (they get assigned a different set of classmates each week). To prime students to give peer feedback that is specific, actionable, and constructive, we demonstrate examples of good and bad feedback in class. To guard against rude comments, we also have a Code of Conduct that counts for a few points in their grade.

We leave Weeks 6 and 9 free for students to revise milestones based on feedback from prior weeks (see Table 1); this ensures that they have enough time to act on the feedback they have been given.

Theoretical rationale: The process of continual formative feedback, reflection, and revision is core to the theory of studio-based design education [15, 25, 31]. We extend those ideas online by having students and TAs leave feedback asynchronously via Google Docs comments. In addition, prior work has shown that the act of giving feedback to peers helps students to indirectly reflect on and improve their own work [13]. Our students give feedback to 20 of their classmates throughout the term (2 each week x 10 weeks).

Potential benefits: Instead of just seeing a grade and then hurriedly moving onto the next assignment, students get a chance to revise based on feedback to apply what they learned. From a review:

> *"I wish other college courses were similar to this format because you get to revise your work through the TA's critique, without it hurting your grade. Far too many college courses deduct points without telling you why, and are paced in a way that you can't even reflect on your mistakes."*

Students also benefit from their classmates' perspectives (again without it affecting their own milestone grades). During the ten-week term, each student sees feedback from 20 classmates (two each week). Note that this means each student also looks at 20 classmates' projects throughout the term, which may indirectly give them design inspiration for their own projects.

Finally, peer feedback helps with scale since students can get feedback from a large cohort of classmates in addition to TAs [13].

5 INFORMAL ASSESSMENT FROM STUDENTS

Since this is an experience report and not a formal research paper, we do not have a rigorous assessment of course outcomes. That said, we have provided student anecdotes from course reviews throughout this paper to give some indicators of potential benefits. Many student comments praised the pragmatic nature of our course:

> *"Dear Future Students, TAKE THIS CLASS! It is hands-down the most practical class I have taken in my three years at [university]. [...] This is the only class I've taken so far in my three years at [university] that might actually help me find a job as it guides you through the process of writing a case study for your portfolio."*

Perhaps the best indicator of success is if students actually get jobs using the case studies they created in our course. We have some anecdotal evidence of this happening via a few student emails sent to us so far, but we have not done a formal study.

Even though student evaluations are biased [5, 6, 9], we present an overview here for reference: 97% of students in our first offering chose Yes for 'I recommend this course' and 100% did for our second offering (which is rare for courses with over 100 students). In those years, the most similar project-based HCI/UX design courses in our department had 'I recommend this course' votes of 61%, 67%, 89%, and 92%. And our department average over all courses was 85%.

Critiques from student reviews: While not requiring coding opens up opportunities for more creative and ambitious projects, some advanced CS students wanted the opportunity to learn more about the implementation details of coding up user interfaces. Also, not having a real implementation makes it harder to do authentic usability testing; students can only test using mock-up prototypes. Some students found the workload to be uneven across different weeks, so we could do a better job at smoothing out the pace. Finally, TAs varied in their design and pedagogical expertise, so we are now exploring ways of having students get personalized attention from multiple TAs rather than only their one assigned TA.

6 CALLS TO ACTION FOR CS INSTRUCTORS

Even though our course was on HCI, we believe that the five pedagogical principles we implemented can apply to other project-based computing courses as well. These include courses in sub-fields including (but not limited to) software engineering, computer systems, A.I., graphics, security, and data science. Although the specific technical content, project prompts, and concrete deliverables will differ widely across computing sub-fields, the spirit behind these principles can transfer well. Thus, here are our calls to action:

(1) **Industry-relevant project prompts**: We encourage instructors to ask the top-performing alumni of their courses (who are now in relevant industry jobs) for what the most compelling modern trends are in their field. What would alumni like to see in resumes and portfolios when hiring new colleagues at their organizations? Use these insights to create course project prompts that can help students get those jobs.

(2) **Final deliverable optimized for job-seeking**: We encourage instructors to address the "last-mile" problem by teaching students how to turn their projects into a final deliverable that can be directly presented to employers. This will take different forms for different CS sub-fields (e.g., in HCI it is a design case study), but again recent course alumni will have a good sense of what is required for a strong portfolio entry.

(3) **No coding required**: Consider not requiring coding as a way to lower barriers to entry and to make the course more inclusive to people with less experience. We acknowledge that programming is essential for some CS courses, but consider the use of alternative non-coding tools when possible.

(4) **Effort-based grading**: To improve inclusiveness, consider a low-stress effort-based grading policy. We have found that students can still do excellent work on projects if they are motivated by the outcome (e.g., if it helps them to get internships and jobs). This will also help your class scale better by enabling fewer TAs to manage higher enrollments without burning out from spending so much time on grading.

(5) **Weekly feedback and chances for revisions**: Train students to give good peer feedback to one another and provide them with time to revise assignments based on feedback. Also, it is important to decouple feedback from grades (e.g., using effort-based grading) so TAs can feel comfortable giving critical feedback without worrying about harming their students' grades. Doing so also fosters a collaborative (rather than judgmental) relationship between TAs and students.

7 CONCLUSION

We have presented a novel user-centered design course that guides students through a computing-related design project that they write up as a case study in an industry-standard format. Our course innovates upon prior HCI electives with its explicit focus on job preparation, inclusion, and scale. Our students have used these projects to get software industry jobs as UI/UX designers and product managers. While creating this course, we developed five pedagogical principles based on research in authenticity [12, 32], inclusion [20], effort-based grading [16], and peer feedback [13]. We encourage computing education researchers to use the ideas presented in this experience report as the basis for more formal studies of efficacy.

ACKNOWLEDGMENTS

Special thanks to Rajiv Sancheti and David Wu for helping us co-create the first iteration of this course in Fall 2019 and to Judy Chun and Kendall Nakai for their valuable feedback on the materials surrounding this course. Also thanks to all of the undergraduate TAs (past, present, and future!) who continually contribute to the ongoing evolution of this course.

This material is based upon work supported by the National Science Foundation under Grant No. NSF IIS-1845900 and from a grant from the Alfred P. Sloan Foundation.

REFERENCES

[1] [n.d.]. Parsons School of Design. https://www.newschool.edu/parsons/. Accessed: 2021-08-10.

[2] [n.d.]. Rhode Island School of Design. https://www.risd.edu/. Accessed: 2021-08-10.

[3] Johan Aberg. 2010. Challenges with Teaching HCI Early to Computer Students. In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education* (Bilkent, Ankara, Turkey) *(ITiCSE '10)*. Association for Computing Machinery, New York, NY, USA, 3–7. https://doi.org/10.1145/1822090.1822094

[4] Julie Barnes, Rob Bryant, Daniel D. McCracken, and Susan Reiser. 2003. Teaching Human-Computer Interaction: Reports from the Trenches. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education* (Reno, Nevada, USA) *(SIGCSE '03)*. Association for Computing Machinery, New York, NY, USA, 125–126. https://doi.org/10.1145/611892.611901

[5] Anne Boring, Kellie Ottoboni, and Philip Stark. 2016. Student evaluations of teaching (mostly) do not measure teaching effectiveness. *ScienceOpen Research* (2016).

[6] Kerry Chávez and Kristina M.W. Mitchell. 2020. Exploring Bias in Student Evaluations: Gender, Race, and Ethnicity. *PS: Political Science amp; Politics* 53, 2 (2020), 270–274. https://doi.org/10.1017/S1049096519001744

[7] Elizabeth F. Churchill, Anne Bowser, and Jennifer Preece. 2013. Teaching and Learning Human-Computer Interaction: Past, Present, and Future. *Interactions* 20, 2 (March 2013), 44–53. https://doi.org/10.1145/2427076.2427086

[8] Sarah Douglas, Marilyn Tremaine, Laura Leventhal, Craig E. Wills, and Bill Manaris. 2002. Incorporating Human-Computer Interaction into the Undergraduate Computer Science Curriculum. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education* (Cincinnati, Kentucky) *(SIGCSE '02)*. Association for Computing Machinery, New York, NY, USA, 211–212. https://doi.org/10.1145/563340.563419

[9] Justin Esarey and Natalie Valdes. 2020. Unbiased, reliable, and valid student evaluations can still be unfair. *Assessment & Evaluation in Higher Education* 45, 8 (2020), 1106–1120. https://doi.org/10.1080/02602938.2020.1724875 arXiv:https://doi.org/10.1080/02602938.2020.1724875

[10] Guiseppe Getto and Fred Beecher. 2016. Toward a Model of UX Education: Training UX Designers Within the Academy. *IEEE Transactions on Professional Communication* 59, 2 (2016), 153–164. https://doi.org/10.1109/TPC.2016.2561139

[11] Colin M. Gray. 2014. Evolution of Design Competence in UX Practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1645–1654. https://doi.org/10.1145/2556288.2557264

[12] Mark Guzdial. 2015. Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics* 8, 6 (2015), 1–165.

[13] Catherine M. Hicks, Vineet Pandey, C. Ailie Fraser, and Scott Klemmer. 2016. *Framing Feedback: Choosing Review Environment Features That Support High Quality Peer Assessment.* Association for Computing Machinery, New York, NY, USA, 458–469. https://doi.org/10.1145/2858036.2858195

[14] Bowen Hui. 2020. *Lessons from Teaching HCI for a Diverse Student Population.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3428029.3428054

[15] Christopher D. Hundhausen, N Hari Narayanan, and Martha E. Crosby. 2008. Exploring Studio-Based Instructional Models for Computing Education. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '08)*. Association for Computing Machinery, New York, NY, USA, 392–396. https://doi.org/10.1145/1352135.1352271

[16] Asao B Inoue. 2019. *Labor-based grading contracts: Building equity and inclusion in the compassionate writing classroom.* WAC Clearinghouse.

[17] Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* Association for Computing Machinery, New York, NY, USA.

[18] Steve Krug. 2014. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability* (3rd ed.). New Riders Publishing, USA.

[19] Chinmay E. Kulkarni, Michael S. Bernstein, and Scott R. Klemmer. 2015. PeerStudio: Rapid Peer Feedback Emphasizes Revision and Improves Performance. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale* (Vancouver, BC, Canada) *(L@S '15)*. Association for Computing Machinery, New York, NY, USA, 75–84. https://doi.org/10.1145/2724660.2724670

[20] Colleen M. Lewis, Ruth E. Anderson, and Ken Yasuhara. 2016. "I Don't Code All Day": Fitting in Computer Science When the Stereotypes Don't Fit. In *Proceedings of the 2016 ACM Conference on International Computing Education Research* (Melbourne, VIC, Australia) *(ICER '16)*. Association for Computing Machinery, New York, NY, USA, 23–32. https://doi.org/10.1145/2960310.2960332

[21] Julia M. Markel and Philip J. Guo. 2021. Inside the Mind of a CS Undergraduate TA: A Firsthand Account of Undergraduate Peer Tutoring in Computer Labs. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 502–508. https://doi.org/10.1145/3408877.3432533

[22] D. Scott McCrickard, C. M. Chewar, and Jacob Somervell. 2004. Design, Science, and Engineering Topics? Teaching HCI with a Unified Method. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (Norfolk, Virginia, USA) *(SIGCSE '04)*. Association for Computing Machinery, New York, NY, USA, 31–35. https://doi.org/10.1145/971300.971314

[23] Diba Mirza, Phillip T. Conrad, Christian Lloyd, Ziad Matni, and Arthur Gatin. 2019. Undergraduate Teaching Assistants in Computer Science: A Systematic Literature Review. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. Association for Computing Machinery, New York, NY, USA, 31–40. https://doi.org/10.1145/3291279.3339422

[24] Alannah Oleson, Meron Solomon, and Amy J. Ko. 2020. *Computing Students' Learning Difficulties in HCI Education.* Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376149

[25] Yolanda Jacobs Reimer and Sarah A. Douglas. 2003. Teaching HCI Design With the Studio Approach. *Computer Science Education* 13, 3 (2003), 191–205. https://doi.org/10.1076/csed.13.3.191.14945 arXiv:https://doi.org/10.1076/csed.13.3.191.14945

[26] Helen Sharp, Yvonne Rogers, and Jenny Preece. 2007. *Interaction Design: Beyond Human Computer Interaction.* John Wiley amp; Sons, Inc., Hoboken, NJ, USA.

[27] Stigo. 2017. The Last Mile – the term, the problem and the odd solutions. https://medium.com/the-stigo-blog/the-last-mile-the-term-the-problem-and-the-odd-solutions-28b6969d5af8. Accessed: 2021-08-10.

[28] Fabricio Teixeira and Caio Braga. [n.d.]. The Case Study Factory. https://essays.uxdesign.cc/case-study-factory/. Accessed: 2021-08-10.

[29] Sander Valstar, Sophia Krause-Levy, Alexandra Macedo, William G. Griswold, and Leo Porter. 2020. Faculty Views on the Goals of an Undergraduate CS Education and the Academia-Industry Gap. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 577–583. https://doi.org/10.1145/3328778.3366834

[30] Sander Valstar, Caroline Sih, Sophia Krause-Levy, Leo Porter, and William G. Griswold. 2020. A Quantitative Study of Faculty Views on the Goals of an Undergraduate CS Program and Preparing Students for Industry. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (Virtual Event, New Zealand) *(ICER '20)*. Association for Computing Machinery, New York, NY, USA, 113–123. https://doi.org/10.1145/3372782.3406277

[31] Mihaela Vorvoreanu, Colin M. Gray, Paul Parsons, and Nancy Rasche. 2017. *Advancing UX Education: A Model for Integrated Studio Pedagogy.* Association for Computing Machinery, New York, NY, USA, 1441–1446. https://doi.org/10.1145/3025453.3025726

[32] Allan Wigfield and Jacquelynne S Eccles. 2000. Expectancy–value theory of achievement motivation. *Contemporary educational psychology* 25, 1 (2000), 68–81.

[33] Lauren Wilcox, Betsy DiSalvo, Dick Henneman, and Qiaosi Wang. 2019. Design in the HCI Classroom: Setting a Research Agenda. In *Proceedings of the 2019 on Designing Interactive Systems Conference* (San Diego, CA, USA) *(DIS '19)*. Association for Computing Machinery, New York, NY, USA, 871–883. https://doi.org/10.1145/3322276.3322381