I have found mentoring and teaching to be incredibly rewarding. As an undergraduate at the University of Washington (UW), I was privileged to be mentored by Michael Ernst and David Notkin. Their inspirational mentoring led to two of my publications, my undergraduate thesis, and my interest in research and the mentoring of undergraduates. As a Ph.D. student, I was privileged to mentor a diverse set of undergraduate and Master’s students. Three of the students I mentored have moved on to pursue graduate degrees (Ph.D. at PKU, M.Eng. at Cornell, M.S. at UPenn), two more will be applying for graduate school this year, and two have accepted jobs in industry (Microsoft and Salesforce). Beyond mentoring students for research, I am also a member of AccessComputing and the Diversifying Future Leadership in the Professoriate (FLIP) Alliance, where I also mentor students. Furthermore, I was a teaching assistant at two universities (UW and UIUC) for three different classes. My responsibilities included lecturing all students, creating assignments and exams, meeting with students individually, and grading. Two of those classes were at the undergraduate level, while one was at the graduate level. The sizes of these classes ranged from \(\sim 40\) to \(\sim 100\) students. Beyond my experiences as a teaching assistant, I have also given five guest lectures for multiple professors. In the rest of this statement, I will detail my approaches to mentoring and teaching students and how my experiences have shaped my approaches and teaching interests.

1 Mentoring Approach During my Ph.D., I mentored eight undergraduate and two Master’s students. The students that I mentored have coauthored eight publications with me (with two more in submission). Of the eight undergraduate students that I worked with, two are female. One of them has gone on to pursue an M.S. at UPenn, while the other is still pursuing her undergraduate at UIUC and has already won the best presentation award for her work with me at the Spring 2019 Promoting Undergraduate Research in Engineering poster symposium [1]. After receiving the award, she wrote a card to me to say how grateful she was for the experience and advice that I provided throughout the project.

Overall, my mentoring approach is focused on two aspects: fostering interest in a project and encouraging a collaborative environment. To foster interest in a project, I assign concrete tasks to students, especially students who are just starting research. As they work on the concrete tasks, I ask them about the challenges they encounter along the way, and I encourage and help them address those challenges. Doing so enables students to gain the necessary technical and critical-thinking skills, and it also helps them find a problem that they are truly passionate about. To encourage a collaborative environment, I often work alongside my students. Namely, I provide specific hours during which the students could come to my office (or join a remote meeting in light of the pandemic), and we would work together on the project. Additionally, I also make myself frequently available in case they encounter challenges that they are stuck with. In doing so, the students who work with me often feel that I am a mentor and a vital member of the project who is there to address challenges that they encounter. I also often have students interested in the same topic pair up, especially at the beginning of a project. Doing so enables the students to learn from each other, solve much more challenging problems than they could solve if they worked alone, and be more motivated by each other.

2 Teaching Approach The main theme of my teaching approach is to have students learn not just the theoretical side of software engineering but also how to apply such theory in practice. My broad industry experience (working as a mobile software engineer for one year and an intern in five industry internships) enables me to bring unique insights into how the theoretical side of software engineering can be applied. Here, I will focus on three main aspects of my teaching
approach: working on team-oriented projects, focusing on concepts that are often used in industry, and teaching students through the use of examples.

**Work on team-oriented projects.** According to Putnam’s study on the cost of software development [2], teams with three to five members have higher productivity per member than teams of other sizes. My industry experience also showed that most software development teams have around three to five members. To improve my students’ productivity and prepare them for industrial software development, I simulate industry-like team environments for my students and meet with them frequently. I let students indicate whom they may want to work with and take that into account when assigning teams. However, I also randomly assign some students to every team. Letting students choose a few members while randomly assigning the rest simulates how team assignments typically work in industry. A developer may not join a team that contains no one that the developer wants to work with. Yet, the developer may not get to choose everyone on the team. With the rising popularity of tools (e.g., CATME [3]) that automate the team formation workflow, I also plan to customize and use such tools in my future classes. One key challenge with using team-oriented projects for teaching is that some students may not be as productive as the others in a team. To help with this challenge, I frequently meet with each team to understand who may be under-performing and ensure that every student has a clear role. In the software testing graduate-level course where I was a teaching assistant, a semester-long, team-oriented project was assigned, and I met with students to help form teams and ensure that students had clear roles between project milestones.

**Focus on industry-related concepts.** Beyond simulating industry-like teams, I also teach concepts relevant to software development, e.g., code reviews. When performed properly, code reviews not only help uncover defects, but they also provide a wide range of benefits, such as knowledge transfer, team awareness, and improved solutions to problems [4]. Another important industry-related concept is coding conventions, which help write safer, more secure, more reliable, more testable, and more maintainable code. In the two undergraduate courses that I was a teaching assistant for, we placed nontrivial emphasis on the coding conventions used in students’ homework. Doing so not only helps prevent problems in students’ homework but is likely to instill proper coding etiquette in students such that their overall code quality is improved in the years to come.

**Teaching by example.** To help students grasp complex concepts, I first demonstrate the problem and its solution using examples. In the five guest lectures that I have given, feedback from students mentioned that examples were particularly useful to help them understand the concepts. Beyond providing examples, I also employ live coding sessions. In two of my guest lectures, I not only discussed the concepts with my students, but I also demonstrated through actual coding and scripting how I would solve the problem. For example, in one of the guest lectures, I taught students how to debug flaky tests by debugging one I had no prior understanding of. My approach to teaching students by example is similar to the saying that one should lead by example. In doing so, the feedback from students after my lectures has shown that students are much more engaged, and they possess a better understanding of the concepts being taught.

### 3 Teaching Interests

My prior teaching experiences have already included classes at the undergraduate and graduate levels in two different universities. Regardless of what course I teach, I will strive to equip students with the skills to succeed in academia and industry. Based on my teaching and research experience, I feel qualified and am eager to teach undergraduate classes related to software engineering, programming languages, formal methods, security, compilers, systems, data structures, and mobile application development. Beyond teaching existing classes, I will also create new classes that bridge the gap between software entrepreneurship and software engineering.
topics. For example, my broad industry experience and my research with mobile apps make me an ideal candidate to teach a course on the entrepreneurship of mobile apps.

As for graduate classes, I would like to teach courses related to software engineering. Such courses would allow me to share various software engineering problems that I have researched (e.g., software testing, program repair, code-clone detection, and mobile application testing). By doing so, I hope to give students exposure to different state-of-the-art research problems and solutions in software engineering and help them pursue research related to software engineering. I will also look to organize seminars on software engineering. Such seminars were instrumental in exposing me to different software engineering topics during my graduate studies and pushing me to think critically about research. As long as remote meetings are the norm due to the ongoing pandemic, I will also organize seminars that would span across institutions. Overall, I look forward to teaching various classes and organizing seminars to enable the students at my institution to succeed in academic and industry settings.

References