Educating the next generation of engineers and scientists has been a fulfilling part of my career over the past five years that I hope to continue. I will use this statement to discuss my duties as an educator, my approach to fulfilling that duty, and ultimately why I will add value to your institution.

My teaching experience includes required courses, upper level electives, and capstone courses, ranging from 40 to 500 students per class, reaching a total of 1309 students. I have co-taught courses (Computer Organization; Software Engineering; Databases) and independently taught courses (Conversational AI; Compilers; Web Systems) as the instructor of record. I designed the Conversational AI course, which is now a routinely-offered capstone course. Moreover, three of these semesters were fully remote due to the COVID-19 pandemic. Overall, my average formal ratings from students are 4.84/5.0 compared to a median of 4.4/5.0 across the entire engineering school.

Teaching Philosophy

I very strongly believe in fostering student communication. While typical classroom settings can definitely benefit from think-pair-share or flipped classroom strategies, I have observed computer science students increasingly seek to engage remotely (especially due to COVID-19) as technology permits remote engagement (e.g., through lecture recordings and automatically-graded assignments). In addition, I believe that computer science education before the university level prepares students unequally, and thus I view an important part of my role to be fostering an environment in which every student can learn effectively, regardless of background. To that end, I favor (1) adapting course material to encourage inclusivity of students regardless of background, and (2) communicating strong motivation of course content as it relates to student career development.

I find that students often feel left out due to imposter syndrome — this was particularly evident to me when encountering students at UM who either came from a less prepared background (e.g., students from high schools that did not have introductory computer science courses) or had transferred in from a community college. I addressed this both by adopting certain vernacular (i.e., avoiding words like "obviously" or "clearly" when explaining topics) and by using humor in lectures to keep the tone light enough that more students would feel comfortable contributing. Students have reported, *"We love Kevin! He handled the semester with compassion and grace,"* and *"Kevin … legitimately helped engage students. Very conscious of what has and hasn't been explained."*

I place a substantial focus on motivating course material — I routinely use platforms such as Piazza for discussing the impact that material will have on student careers. This has generally been well-received by students. In my course evaluations, students have mentioned, *"Piazza posts were very helpful," "[Kevin] always answers questions on Piazza very quickly,"* and *"I like that [Kevin] actually goes on Piazza unlike most professors."* I view it as essential to clearly communicate the benefit of learning a topic with respect to student career development.

Course Goals and Expectations

My observation is that students benefit from clear course goals and motivation — I spend time ensuring that students know what tasks need to be completed for assignments and what to expect on exams. This includes revising assignment specifications based on feedback from previous semesters, and clarifying student questions so that students can focus attention on defined course goals rather than extraeneous implementation topics. For example, while teaching Databases, I encouraged a specific known-good SQL development setup, but freely allowed students to choose any development environment instead. This allowed students who were less adept with scripting to pick up the course material, while permitting flexibility to students who knew how they wanted to complete assignments. Student reported, "*[Kevin is] absolutely killing it with Piazza help, that relieves a lot of stress and anxiety for me...*" and "*There was never a moment ... that I felt like we were not being given the utmost respect and academic devotion ... or that we were in anything less than the best hands.*"

I also incorporate a focus on systems building, integration, and independent student creativity into course material and assignments that I design, being careful to address students coming from disadvantaged backgrounds. For example, in my Software Engineering course, I developed optional screencasts for students to help them get a virtual machine set up for completing the assignments in a known-functional environment. Moreover, while teaching Conversational AI, I encouraged student teams to motivate and develop their own semester-long projects to keep them engaged in a topic they cared about. Students have reported, "*Taught real world applications and was relevant to the real world,*" "*The course was absolutely great at teaching practical skills,*" and "*taught a lot of really cool topics that I'd wish I'd known for previous internships.*"

Exam Delivery and Academic Integrity

I view exams as important instruments for assessing student mastery in a limited block of time. To that end, they are useful study and practice tools. However, since they assess student mastery in a very small time window under high stress, I favor weighing exams less than traditional written assignments for grading purposes. More recently, the COVID-19 pandemic has created student anxiety over academic integrity in remote exam delivery. Moreover, students report feeling pressure to cheat to keep pace with students who decide to cheat in a remote environment. To address these student (and faculty) concerns, I have led an effort to create techniques for randomly generating exam questions unique to each student. An instructor defines a "grammar" for the type of question they want to ask, from which we can create many variants of the same question automatically and uniquely for each student. For example, an instructor can create unique coding snippets for each student by specifying parameters such as "a function with three inputs, two if statements, an else branch, and a while loop." This approach allows creating questions that are (1) fair across students, (2) easily graded, and (3) effective in reducing incentives for students to cheat. I presented this work to the Dean for undergraduate education, who agreed to fund this effort. I have been leading a team to deploy randomized exams in other courses—both CS and other engineering disciplines—in the engineering school.

Conclusion

Overall, I am fortunate to have taught so many courses and students during my postdoctoral research. I have learned many useful techniques and approaches to effectively and efficiently engage and motivate students from many backgrounds. I hope to continue contributing to the university mission of undergraduate education throughout the rest of my career.

A list of courses taught appears on the next page.

Teaching Experience Summary

During my postdoctoral research, I have had the opportunity to teach six different courses as the instructor of record at the University of Michigan, reaching 1309 undergraduate students. My teaching is summarized below:

- **Courses taught as instructor of record**
  - **Conversational Artificial Intelligence (2 semesters, 110 students total)**, a capstone elective course I designed where students work in large groups to develop a custom virtual assistant (e.g., specialized Alexa or Siri).
  - **Undergraduate Computer Organization (500 students)**, a required second year course discussing the five-stage pipeline, cache organization, and virtual memory.
  - **Undergraduate Compilers (69 students)**, an upper level elective focused on lexical analysis, parsing, code generation, and dataflow analyses for optimization.
  - **Databse Systems (200 students)**, an upper-level elective discussing SQL, relational algebra, and database implementation.
  - **Web Systems (70 students)**, a rigorous upper-level elective focused on concurrent and distributed web application design, including Flask, React, SQLite databases, and distributed computing with Hadoop.
  - **Software Engineering (2 semesters, 320 students total)**, an upper-level elective discussing practical software development methods and management techniques.
- **Service activities**
  - I led an effort across the Engineering School to build robust online exams during COVID-19. This university-funded effort resulted in a technique for randomly generating questions unique to each student sitting an exam to promote academic integrity.
  - I formally advised 150 undergraduate students during three semesters as part of the Undergraduate Advising Committee.