Statement of Teaching Philosophy

Md Imanul Huq, Ph.D.

imanulhuq@gmail.com Personal Webpage

WHAT I WANT STUDENTS TO BE ABLE TO DO

My teaching is guided by a simple standard: students understand an idea when they can explain it clearly, apply it, recognize where it fails, and articulate what they learned from that failure. I design my courses to move students repeatedly through that cycle of explanation, application, testing, and reflection.

The outcome I want to avoid is a student who can produce the correct answer without understanding why it is correct—someone who can compile a program but cannot predict what will happen when the input changes. Such a student may succeed on a familiar assignment but struggle when the problem changes. My role is therefore not to prevent students from making mistakes, but to help them develop the ability to identify, explain, and correct those mistakes independently.

I arrived at this standard by using it on myself. As a student I learned difficult material by trying to explain it in ordinary language, in the manner Richard Feynman described, and the points where my own explanation broke down were reliably the points I had not yet understood. When I began teaching, I asked students to do the same, and it worked on them as it had worked on me. I have used it ever since as a practical diagnostic: when a student cannot explain a concept without relying entirely on terminology, that usually marks where understanding is still developing.

EXPLANATION AS A FOUNDATION FOR LEARNING

After introducing a concept, I ask students to reconstruct it in their own words, identify the assumptions behind it, work through a concrete example, and recognize the limits of their understanding. That final step is especially important. A student who can say precisely where their confidence ends has identified a productive place for further learning.

In systems courses, for example, I would not want students merely to define processes, threads, or deadlocks. I want them to explain why an operating system needs those abstractions and predict what will happen under a particular schedule. In a software engineering course, I would treat requirements, version control, code review, and documentation as part of the intellectual content of the course rather than as administrative overhead attached to programming. Students should be able to explain the consequences of a design decision, not simply report that the decision was made. In compiler-related material, students who can explain how source text becomes tokens, then structured representations, and eventually an intermediate form have developed the conceptual scaffold needed to understand the formal mechanisms that follow.

The classroom practices supporting this approach are straightforward: short peer explanations, live code walkthroughs, deliberate debugging, and questions that require students to predict an outcome before seeing it. When a student cannot explain a step, that difficulty becomes useful information about where instruction should focus next.

LEARNING THROUGH PRACTICE

Explanation must be followed by construction. I therefore structure practical work as a progression. Early exercises are scaffolded and typically have a clear solution. Intermediate exercises ask students to modify, extend, or diagnose an existing system. Later exercises become more open-ended, requiring students to determine what the problem actually is before deciding how to solve it.

In an operating systems course, discussions of scheduling, synchronization, and memory management can be paired with Linux exercises in which students observe actual process behavior and compare it with what theoretical models predict. In cybersecurity courses, a laboratory can progress from understanding an attack, to reproducing it safely in a contained environment, to implementing a defense, and finally to testing whether that defense addresses the underlying vulnerability rather than only the demonstrated symptom.

I view laboratory design and management as part of teaching rather than as separate logistical responsibilities. I have developed laboratory sequences and instructional materials from the ground up for digital logic, microprocessor, and networking courses, including equipment setup and laboratory examinations. As the teaching assistant for a fully asynchronous online Network Security course at Texas A&M University, I developed a complete course package, including slides, staged assignments, and assessments aligned with stated learning objectives.

DESIGNING AROUND THE ACTUAL BARRIER

I have taught face-to-face, online, and hybrid courses, and the practical constraints differ in each.

The Network Security course described above was fully asynchronous and online, with students distributed across multiple time zones. In that setting, access to instruction became the central problem. A single office-hour schedule could not serve every student equally, and students working many hours outside the instructor's time zone could easily become isolated from support.

To address that, I built an AI assistant grounded in the course material and made it available to students throughout the term. It was not a static resource. Over the semester I extended it using the questions students were actually bringing to my teaching-assistant hours, so that the explanations it could offer moved closer to the points where students were getting stuck. The purpose was to make course-specific help available at the moment a student needed it, regardless of time zone.

This experience reinforced a broader principle in my teaching: when students are struggling, I first try to identify the actual barrier to learning and then design around that barrier rather than simply increasing the amount of instruction.

ASSESSMENT AS INFORMATION

I see assessment as a source of information for both students and instructors, not only as a mechanism for assigning grades. A small number of high-stakes examinations can show what students were able to do at particular moments, but they provide limited information about where understanding began to break down.

I therefore use multiple forms of assessment. Short formative exercises reveal misconceptions while there is still time to address them. Implementation assignments show whether students can build functioning systems. Conceptual questions ask students to explain why an approach works rather than merely whether it produced the expected output. Multi-stage projects, in which later stages depend on earlier design decisions, help assess integration and make students' reasoning visible. When appropriate, I also ask students to walk through their own code or analysis aloud. Doing so helps distinguish genuine understanding from an answer that happens to be correct.

Feedback is most useful while students can still apply it. I try to return work in time for students to use the feedback on what comes next, and I focus on where the reasoning diverged rather than simply identifying the final answer as incorrect. When many students make the same mistake, I also treat that pattern as information about my own sequencing or explanation and reconsider how I taught that material.

STUDENTS ARRIVE BY DIFFERENT PATHS

Students enter the same classroom with very different academic histories. Some transfer from other programs, some change majors, some return after years in the workforce, and some come from educational systems in which prerequisite material was taught differently or not at all. I do not see rigor and support as competing goals. The academic standard can remain high while students are given multiple effective paths toward reaching it.

In practice, this means determining what students actually know rather than assuming that completion of prerequisite courses guarantees a uniform starting point. I use early diagnostic activities to identify differences in preparation and adjust pacing where appropriate. I also emphasize predictable course organization, explicit learning objectives, transparent grading criteria, and examples that progress from simple to complex. Students who are behind should be able to identify where they need to begin rather than simply recognizing that they are behind.

I have taught mixed-ability adult learners in non-degree settings as well as university students in two different national educational systems. One lesson has remained consistent across those environments: students who appear disengaged are sometimes not unwilling to learn; they are unsure how to begin. Good course design should make that first step visible.

Access is a prerequisite for learning as well. When instruction moved online during the pandemic, the most immediate challenge for some of my students was not pedagogy but hardware. Students without suitable computers could not participate meaningfully in online instruction. I bought and collected discarded computers, repaired them, and provided more than twenty-three working computers and laptops to students who did not have adequate devices. That experience reinforced for me that teaching requires identifying the practical conditions preventing students from participating and, where possible, addressing them.

I approach accessibility with a similar philosophy. Rather than treating accessibility only as a response to an individual accommodation request, I try to incorporate it into course design from the beginning. I review instructional materials for

screen-reader compatibility, document structure, and whether instructions can be understood without relying exclusively on visual cues.

TEACHING WHEN ANSWERS ARE EASILY GENERATED

The widespread availability of large language models has changed the environment in which programming and computing are taught. In an introductory programming course, I became aware that some students were using these systems to complete assignments. The course's academic integrity policy remained in force, but I concluded that spending the semester trying to detect AI use would contribute relatively little to their learning. Instead, I redesigned parts of the work so that generated output itself became something students had to examine and understand.

For each problem, I gave the class a specific prompt to run through the model and asked students to bring back what it produced. We then worked from that output together: reading it closely, identifying the assumptions behind it, testing where it broke, and deciding what needed to change. The prompt changed with each new problem. The exercise was never about obtaining a solution—the solution was the starting point, and the work was evaluating it.

I began to see a change in the questions students asked. Rather than quietly copying code, some students began asking why a generated solution had been structured in a particular way, what assumptions it was making, and why it failed in a particular case. Those questions were educationally much more valuable. I would not treat observations from one course as controlled evidence, but the change influenced how I think about assignment design in courses where powerful generative tools are readily available.

This approach also connects naturally to my research, which examines how people interact with and sometimes over-trust automated systems. AI-generated explanations can be fluent and confident while still being incorrect. That makes them particularly useful objects for the same process I emphasize throughout my teaching: explain the reasoning, test it, identify the gap, and determine what needs to change.

CONNECTING TEACHING AND RESEARCH

My research and teaching also inform each other directly. Over several semesters, I studied how cybersecurity students used PGP encryption in Mozilla Thunderbird. Rather than studying the problem only in an artificial usability setting, the work grew from students' experiences using the technology as part of an actual course.

As a teaching assistant for the course, I was one of the people students approached when the software or workflow became difficult. Their support requests and troubleshooting experiences helped reveal recurring usability problems. The resulting work was published at IEEE PST 2025 and identified points at which an encryption system could challenge even technically prepared users.

The research also changed my teaching. The places where students became stuck were not always the places I had been emphasizing during instruction or office hours. Their difficulties revealed assumptions in my explanations that I had not previously recognized.

This experience reinforced my belief that teaching and research can form a productive feedback loop. Classrooms allow instructors to observe repeatedly where difficult ideas and technologies become confusing in authentic use. Research can help explain those problems, and the resulting knowledge can improve instruction for the next group of students.

RECOGNIZING WHEN THE PROBLEM IS NOT ACADEMIC

Teaching also requires paying attention to changes in student performance and participation. In one of my courses, an undergraduate who had been among the strongest students began submitting work late and then missing class. It would have been easy to interpret that change as disengagement or difficulty with the material. When I followed up, however, I learned that his family circumstances had changed and that he had taken a job to help support them.

The underlying problem was financial rather than academic. The university had a scholarship intended for students in such circumstances, but he did not know about it. I worked with the administration to have him considered, and he ultimately received the scholarship.

I include this experience because I consider awareness of institutional support to be part of effective teaching. Instructors often see changes in a student's performance before those difficulties become visible elsewhere. We cannot solve every problem students face, but we can recognize when a problem falls outside the classroom and help connect students with appropriate resources.

From Following Instructions to Asking Questions

I have supervised five undergraduate theses and senior projects from proposal through defense, and two of those students have continued working with me as collaborators and co-authors on ongoing research. That progression represents one of the outcomes I value most: a student who begins by following an assigned procedure and eventually learns to formulate a question, select an appropriate method, interpret evidence, and defend a conclusion.

At the graduate level, my goal is similar, although the development occurs over a longer period. I am particularly interested in teaching seminar courses in usable security and privacy, human-centered cybersecurity, and the evaluation of AI systems. In those courses, I want students to engage directly with primary research literature—analyzing assumptions, questioning methods, comparing evidence, and developing their own positions rather than relying only on summaries.

My long-term goal as an advisor is for students to become intellectually independent. Technical methods can be taught directly, but research maturity also requires learning to judge which questions are worth pursuing and how much confidence the available evidence can support. Undergraduate courses can begin developing that habit early by asking students not only for answers, but also to identify the limits of what they know.

Evaluating and Improving My Teaching

I do not want to wait until end-of-semester evaluations to learn whether my teaching is working. In one course, I announced what students believed would be a surprise test. Instead, I gave them an anonymous mid-semester evaluation asking about my teaching: the pace of the course, the delivery of material, and whether my explanations were helping them learn. The first sheet asked for their identity and required nothing to be written on it, serving only to make the anonymity visible; the second asked them to evaluate the course and my instruction.

Their most important feedback was that I was moving too quickly. They were right. I changed my pacing and began relying more heavily on evidence of what students could actually demonstrate rather than treating the syllabus calendar as the sole measure of where the course should be. That experience also led me to place greater emphasis on early diagnostics and mid-semester adjustment.

End-of-semester evaluations remain useful, but they arrive too late to improve the experience of the students who complete them. I therefore look for additional indicators throughout the semester: whether the questions students ask in week ten show greater sophistication than those they asked in week three; whether they can solve a problem that was not demonstrated in an example; whether the same misconception continues after I have redesigned the relevant instruction; and whether students who entered with weaker preparation are still progressing successfully.

Ultimately, the outcome I value most is transfer. I want students to be able to encounter a problem they have not seen before, determine which principles are relevant, reason through uncertainty, test their decisions, and explain why they chose a particular solution. A course succeeds when students leave with that ability—not simply when they can reproduce what the instructor said.