WIP:"Towards AI-assisted CS1 classrooms: An experience report on the design of learning activities and assessments in engineering computing courses"

Ayush Pandey
ayushpandey@ucmerced.edu
University of California
Merced, California, USA

Abstract

With generative AI tools commonly available to students, instructors must update the in-class activities and assessments accordingly such that AI can be integrated as a pair programmer. In this experience report, we describe the outcome of AI-assisted active learning in CS1 courses and novel assessment designs to maximize creative exploration. Conventional CS1 assessments can be solved with iterative prompting of generative AI tools like ChatGPT or completed on the fly with Github Copilot. To address this, we designed new formative and summative assessments that promote critical thinking and computational exploration. These new assessments cannot be solved in one-shot by generative AI tools but are still suitable for collaborative learning with AI for students. We present implementation details of these assessments, student outlook, and their limitations. We also discuss the effective use of generative AI for instructors teaching these courses. On reflection, we identify key gaps in learning management systems and educational technologies that present limitations in supporting and grading these AI-assisted activities. The context of this report is the implementation of these new strategies in three diverse settings — a large class of 207 engineering undergraduate majors, a moderate size class of 55 non-engineering majors, and a moderate size class of 68 engineering majors.

CCS Concepts

• **Applied computing** → **Computer-assisted instruction**; • **Social and professional topics** → *CS1*; **Student assessment**; • **Computing methodologies** → Artificial intelligence.

Keywords

computing education, engineering education, ChatGPT, GitHub Copilot, active learning

ACM Reference Format:
Ayush Pandey. 2024. WIP:"Towards AI-assisted CS1 classrooms: An experience report on the design of learning activities and assessments in engineering computing courses". In . ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Context

This experience report is based out of CS1 course designs and instruction for undergraudate students at an Anonymous University in diverse majors and years of study. A summary of the context of this report is given in Table 1 with a list of the computing courses at the university, their enrollment, and the instructional support.

2 Class activities

2.1 Live coding — Asset mindset and teaching to fail

One of the underlying pedagogy principles in using live coding as an in-class activity is "Teaching to fail". Narla [1] argues that vulnerabilities in the classroom can be a powerful tool for learning and even foster a sense of community. Live coding is a great way to show students that it is okay to fail and that it is a part of the learning process. It is important for the instructor to actively discuss this philosphy in the classroom instead of relying on the students to infer it. Live coding and the follow-up discussions around it is also a good opportunity to affirm an asset mindset in the classroom. Students are likely to think about the code in different ways and therefore see the value in their own unique perspective. This is reinforced when the instructor shows that they value these perspectives and multiple solutions to the weakly-defined problem. If the live coding problem is rigid, it prevents these explorations and the asset mindset may not be activated. To implement this activity successully in the class and maximize the learning, the instructor and the students need to be prepared.

Before class preparation for instructors: The first and the most time consuming task for the instructor is to find a problem that has multiple possible solutions. Some examples of such problems are given below:

Course	Total enrollment	Instructional staff	Year
ME 01	207	1 IOR, 4 GSI, 1 ASE	Fall 2023
CSE 01	57	1 IOR, 1 GSI	Fall 2023
EE 01	65	1 IOR, 1 GSI, 2 ASE	Spring 2024

Table 1: Summary of context. IOR: Instructor of Record. GSI: Graduate Student Instructor. ASE: Academic Student Employee.

(1) An array indexing/iteration in a loop
(2) While and/or for loop with different ways to create the iterator or termination condition
(3) Different function signatures to solve the same problem
(4) Local and global variables in a function

The next step is to code up the different structures of the same problem with conceptual blanks or errors in the code. Using GitHub Copilot, the instructor can prompt these alternative solutions to the problem and delete parts of the code to create the live coding prompts for the class. This requires minimum effort, so the main effort should be on coming up with the problem.
The instructor must prepare the slides with the live coding activity description, instructions, expected inputs, and outputs even though the main focus of the activity is the live programming in an IDE. The clear descriptions of the inputs already available in the code and the expected outputs (either errors or console outputs) from the code are appreciated by students as it provides a structure to work with. A common feedback from students working on this activity is that the "correct" answer was not provided at the end or that the "final" code was not shown/discussed in the slides. An easy way to address this is to just provide these directly to the students. But there's also space for keeping the "final" answer open to the design choice/students' perspectives on what qualifies as the final answer, especially if a strong asset mindset activation is desired.
Finally, practicing the activity with the ASEs before the class can help iron out the wrinkles in the activity and also trains the ASEs on how to help the students in class. Even if no ASE support is available in-class, the practice helps the instructor get feedback on the activity.

Before class preparation for students: The students do not need to prepare anything beyond the usual for this activity, which makes it the go-to in-class activity in CS1 education. Having said that, it's often important that the students have finished their prior week's assignment and have turned it in so that the problem choice in the activity could develop on the prior week's material. It would also help the students if they have received the grading feedback from prior week's submitted assignment, so that they are comfortable with the fluid nature of assessment in the class where they receive full credit for their correct design choices and creative exploration.

In-class set up: Introduce the activity in slides in a "problem/issue/expected output" framework, highlighting how such situations can occur in real-world applications. After a quick overview of the expected output and how the activity will be run, switch to your IDE. The online-python.com or replit.com are great cloud-based Python resources that do not require any sign-up. Using a dedicated IDE (like VS Code, Pycharm, or Sublime Text) that all students have set up has its advantages too, but require slightly extra effort on the students' part and also cannot run on iPads/tablets/chromebooks. At UC Merced School of Engineering, even though there is a school wide computer policy that states the minimum required specification of a laptop that each student must have, many students often find themselves scrambling to access a device in the class. In such cases, online platforms are suitable. Many students are working along side school and cannot carry their laptops to the class, many others just forget, and there is a fraction who are still learning to use a computer. ASEs like learning assistants can be a great resource to address these concerns as they walk around the classroom. Fully planning around these issues in advance may not be possible but a general awareness from the instructor can go a long way in designing the activity accordingly.
The instructor should make sure to project a zoomed-in version of the IDE/coding platform, especially for large classes. The instructor can start by a simple coding task with live coding its first part and then by giving students some time to parse so that they can attempt the next line. Free discussions may be allowed or facilitated by ASEs, if available. Instructor can move around to encourage questions and work individually with students depending on the size of the class. In a large class with more than 100 students, it is difficult to individually solve students' errors but with 50 or fewer students it is quite feasible. Moving around is also helpful since students are more likely to ask questions directly when they don't have to project themselves out loud, especially in larger classes.

Challenges: The live code and the instructions are both required at the same time. Use split projections or write instructions in big letters on the whiteboard while the code is projected. Using comments in the code to give instruction is also possible but may not be equally effective for all students. While on the other hand, use of the whiteboard for instructions encourages more movement. Another common challenge is students feeling that the instructor ran through their code quickly or that they could not follow the steps as the code was executed.

Summary The overall level of preparation required for this activity can be rated as "Low-Medium", or a 3 out of 10 on a subjective difficulty scale. The advantage is that no prior reading is required on the students' part since its a live interactive coding session. The time needed in-class for the activity can range from anywhere between 15-30 minutes depending on the difficulty of the activity and the extent of the possibilities that you explore. Student's critical thinking activation in this activity is "High" or 8 out of 10 since they are asked to make decisions and think about the real-world output of the program given the task (as opposed to a conceptual multiple-choice or a fill-in-the-blank). The activity promotes group work and collaboration at a "Medium" level, or a 5 out of 10 since each student is writing their own code but are still discussing the alternatives with their partners or groups. The independence and self-efficacy development in this activity is "Medium-high" or a 6 out of 10 score.

2.2 Collaborative Worksheets

Todo **Sequenced data structures, their indices, order, mutability, and uniqueness**

Loop iteration worksheet

Function calling line order worksheet Students number the lines executed in sequence as function(s) is (are) called.

Abstraction brainstorming worksheet

2.3 Index card activities

Todo **A piece of code can go a long way**(distribute index cards randomly with a piece of code that students have written, then the receiving students build on it/fix it/use it for another task): a file, a function, a class, a module.

Pre-defined index cards by instructor An index card with some code or macros and then prompts are given to the students to use the code or use the macros in a new code.

2.4 Icebreaker activities and memes

Todo

2.5 Minute paper — quick self-efficacy check

A minute paper is a commonly used active learning strategy used to promote inclusive teaching in classrooms. It can be used as a feedback tool for the instructors on student learning or as a self-efficacy check for the students. The minute paper is a quick and easy way to check if the students are comfortable with the material and the pace of the class. It is also a great to check if the students are comfortable with the assessment structure of the class. I have used minute papers mainly as a self-efficacy check in computing classrooms. A majority (around 80%) of students in my classes are first-time programmers and first-generation college students. The higher-order knowledge connection made by a first-time programmer (and often fairly new computer user) is that learning a programming "language" will be similar to learning a spoken language where the vocabulary and grammar rules need to be memorized. But learning computer programming is different because it can be iterative and paved by a path of constant errors made in a low-stakes computer environment, where unlimited tries are allowed. Therefore, feeling confident in their own abilities to program a computer is a critical step in CS1 classes.

Students are often overwhelmed by errors in computer programs or the large number of syntax they feel they need to memorize. To understand this struggle and their self-efficacy, minute papers can be used with a defined structure to prompt the students. In the next class after the minute paper activity, it is important that the instructor shares a summary of the responses from the minute paper with students to model transparency and flexibility. More importantly, seeing that others are also struggling with similar issues can also be a boost to their self-efficacy. To summarize these responses, word clouds can be used. An easy AI-based clustering using ChatGPT can also be used for more nuanced analysis and to promote the discussion on the advantages and pitfalls of ChatGPT.
Examples of minute papers: For CS1 classes, minute papers can be based on the following ideas:

(1) A one-line summary of a main concept from the class.
(2) A self-efficacy check on an upcoming "big assignment".
(3) A reflection on the most challenging part / the muddiest point of the class.

Giving prompts for minute paper is great. For example, to activate a self-efficacy check for a project, the minute paper prompt could ask the students to complete the following `"This week, I will finish ... To finish this, I will get help from ... Next week, I will finish ... To finish that I will, ... Overall, I feel ..."`

3 Formative assessments

3.1 Multi-select problems — the 5E model and cognitive dissonance

Formative assessments in CS1 take many forms but the most commonly used one is a programming assignment where the problem is described and the students need to complete the code from scratch. While programming assignments still hold value in CS1, the wide availability of generative AI tools has impaired the authenticity of student learning progress as indicated by the assignment grades. Instructors are often recommended to innovate the problems in the programming assignments but this is time intensive without much benefit as the continuously improving AI tools can solve more and more complex programming tasks. Using oral assessments during lab sessions and providing feedback to the students in-person is another commonly used remedy but it cannot be scaled either. Another alternative is multi-select multiple choice problems as part of the assignments in CS1 or as a live assessment in labs or classes. Multi-select choice questions with many options are different from a pop quiz or a graded quiz or a "quick check" multiple choice problem. They are designed to be formative assessments that promote critical thinking and computational exploration rather than problems that have a fixed answer or are conceptual in nature.
The 5E and the reflect-and-predict framework for CS1: In the multi-select assessment, the 5E model can be effectively applied. To best engage the students, the constructivist 5E model suggests five stages in a learning activity: Engage, Explore, Explain, Elaborate, and Evaluate. I use the multi-select assessment on topics that the students have some idea about but have not finished learning all the related concepts and have certainly not practiced programming assignments on that topic. In this context, students can be engaged by getting them thinking on a limitation of a construct/approach that they have learned previously (like repeating if-else branches instead of a while loop, or the repetitive computation of a task instead of using functions).

Then, the students explore a given piece of code mentally or by running the snippet in a Python compiler. They are asked to predict what would happen before running the code in one or more multi-select question(s). Students receive immediate feedback on their responses to these problems and have unlimited tries to finish. In the process when a student gets a wrong answers, a cognitive dissonance is triggered due to the incorrect prediction. With other follow-up questions, they have an opportunity to reflect on this. Finally, the students can work through the philosphy of why there are so many options to solve the same problem (often a frustrating fact reported by the students). This leads to opportunities where the students can explain their learning. At the end, the students can be tasked with the elaboration of the concept in applying it to programming task where actual code is written by the students or they fill up missing pieces of code by selecting options. In this way, the 5E model is applied to multi-select problems by purposefully designing the problems to trigger cognitive dissonance and reflection in the students.

A follow-up of this formative assessment exercise is to discuss how different perspectives and different options can achieve the same output. The class discussions can be made more inclusive by including historical perspectives on the development of the programming language over the years and how diverse set of communities and researchers have contributed in it. Discussion on best coding practices can also be built on this formative experience.

Preparation for the instructor: The instructor needs to prepare the multiple choice multiple select questions. Ideally, these are problems that the students do not yet know how to fully solve as discussed above.

Preparation for the students: Students need to be comfortable with previous weeks concepts since these problems will challenge those concepts further. Often the effective multi-select problems will lead to un-intuitive answers or point to the idiosyncrasies of the programming language.

Types of CS1 activities suitable for this activity:

(1) Array slicing step-by-step. For a long array, we can index it at different points and in different strategies.
(2) The `len()` function applies to many data structures and in different ways in Python: lists, strings, dictionaries, list of lists, and for use in an interator. On the other hand, the `len()` function can throw errors in situations where students may expect an answer. For example, for `len(2)`, some students may expect the answer to be 2 and not a long error message (type error), which they may or may not understand!
(3) Adding something new to a list or a dictionary. There are many ways to do it in Python: append, update by indexing, concatenate, an elaborated step-by-step approach.
(4) Multiple and nested if-elseif-else branches as a code example, then options on various output choices.
(5) Function code snippets with different ways to call the function and different ways to pass the arguments. Including exploration of local and global variables in the function.

3.1.1 Assessment setup. Students should refer to the reading for a brief introduction of the topic and the big picture idea on how the instructor is proceeding with the topics should be clear. The assessment could be live in-class (with online polls or LMS assignments) or take-home (timed quiz). An audience response system that can be integrated into the class slides can also be used to direct group behavior and provide live feedback on the polling. These problems are also suitable for unstructured Q&A sessions where the students can discuss the problems in pairs or groups or as oral assessments. In live implementations, the students can be prompted to answer the poll by themself after a minute of thinking. In a take-home implementation, the students can have a timer on the LMS that prompts them on thinking and acting times. Before moving to the next problem, the current problem may be locked after a certain time and the students should be shown feedback on their prediction. In live implementations, the students are asked to discuss in pairs and convince each other of their answers. Then, the instructor can open the poll again for responses, let the live responses updates be visible. Finally, each option is discussed and the instructor may or may not go for another round of polling. In a take-home assessment, each consecutive problem should build on the previous to show the students any incorrect details in their predictions of the previous problem. It is important to lower the stakes in this assessment since a key part is to let the students make mistakes and get answers wrong so that cognitive dissonance can be triggered.

Challenges: Audience response systems are usually not free, especially for large groups, so subscriptions are required. With multi-select problems, the percentage of students who chose each option is meaningless. Instead, there should be a better metric of displaying the propensity of an option being chosen by a majority of the class. Another challenge is that after the first show of the responses, a groupthink behavior starts to form and can bias the creative process of individuals. Finally, as is common with other low-stakes assessments and activities, this assessment also suffers from the lack of motivation that the students may have to actively engage since their final grade is not affected by it.

Summary Overall level of preparation required for the assessment is "Medium" or a 5 out of 10 difficulty rating. Finding effective problems that are well-suited for this activity is important. The total time needed for the activity is 3 to 7 minutes. Students' critical thinking is highly activated (an 8 out of 10 subjective score) because it can be fun for the students to think through various prompts in a low-stakes problem and also discuss the "what-if" prompts with other students. The student group work and collaboration is "Medium-high", or 6 out of 10 in a live implementation and a score of 2 in the take-home implementation. Finally, this activity does not promote self-efficacy as much because the students are going through a process of making mistakes and correcting them. Hence, the activation of self-efficacy is "Low" or 4 out of 10 rating.

4 Summative: Open-ended projects

With the general availability of generative AI, summative assessments in CS1 have to be designed with the understanding that the students can use these tools to complete the exams. Rather than switching to timed in-person or similar "stricter" settings to prevent AI use, CS educators must innovate the assessments such that generative AI tools can be used collaboratively. Open-ended projects offer a path forward. Research has shown that projects as the main summative assesssment in engineering and computing courses can have a greater positive impact on student learning and career preparation than conventional exams. With open-ended projects, the idea is to go one step further. If a project problem statement is carefully designed by the instructor and provided to the students, generative AI tools could have a much easier time finishing the project, especially in CS1. Identifying a problem, exploring possible solutions, and defining what a good real-world implementation would look like are the creative elements of any project. Therefore, I designed open-ended project-based summative assessment exams for two CS1 courses.

The open-ended projects can be designed to be creative and exploratory in nature. I gave a set of constraints to the students for their Python projects but the project goals, specific objectives, and the technical approach were assigned as tasks for the students. The

project constraints for the ME021 class were:

All students must work on an independent Python project. This project must demonstrate the following five key elements of Python that you learn in ME 021. An ideal project that grades 100 will

(1) *Control the program flow with branching and loops*
(2) *Uses correct data structures for optimal computations*
(3) *Uses functions for modular code*
(4) *Loads data from files (real-world data is preferable) and writes outputs to files (if needed)*
(5) *Documents the flow of logic with comments, docstrings, and user-friendly messages*

Beyond these constraints, the students were given the independence to propose a project and develop it for their exam. Two main challenges with this style of summative assessment are: lack of clarity for the students on "what do I do?", and the design of a fair grading strategy and rubrics. To address the first challenge, I provided a list of project ideas to the students. The students were also encouraged to propose their own project ideas. Scaffolds were designed carefully for the project so that students could make consistent progress and achieve the best results.

4.1 Milestone assignments as project scaffolds

A total of 5 milestone assignments were designed as scaffolds for this project-based summative assessment. The first scaffold was a "project proposal" assignment. In this assignment, the students wrote a Python code that asked themself questions about the project being proposed. Students submitted their code and their responses as the code ran on the terminal.

Project proposal: The assignment asks students to write a Python code with the following prompts:

(1) Ask the user to input the goal of their ME 021 project in 1 line. Instruct the user that this goal must be succinct and should describe the big-picture problem that the project is addressing.
(2) Ask the user to write down inputs and outputs of the project. That is, what are the particular objectives that this project will achieve.
(3) Ask the user to input the data that this project would need.
(4) Ask the user whether they met with their course TA to discuss the project or not (Yes or No).
(5) Check whether the user answered all four questions. If yes, then output...

The other milestone assignments addressed each of the constraint in the exam instructions (adding branching, loops, functions, files, and visualization).

4.2 Categories of proposed projects

A total of 207 projects were proposed by the students in the ME021 class. The projects can be classified into the following categories: game design (tetris, tic-tac-toe, rock paper scissor, adventure games), computational apps (computing math formula, finance calculators), data analysis (COVID-19 data analysis, stock market analysis), and simulations (UC Merced bus tracker, diet simulator, class registration system). A full list of all projects is available online on GitHub [2].

The second challenge in this exam design is the grading. I designed a grading rubric that was shared with the students before the exam. The rubric was designed to be holistic and included the following criteria: whether the code runs successfully given the desired inputs, significant contribution (subjectively assessed by graders), optimal use of data structures, and correct use of branching, loops, and files/visualizations. A demonstration of the project to a TA was a required grading component for the project. The oral exam ensured the authenticity of the exam and challenged students to present their approach. The TAs were provided with a list of generic question starters (see GitHub [2]). To summarize, the grading of the exam consisted of: 5 milestone assignments, 1 demonstration oral exam, and 1 final project submission (graded on a rubric). The time commitment for grading was high and that is the main limitation of this open-ended exam design, which calls for more research on automatic graders.

4.3 A "general" AI-autograder – a call for action

Automatic grading is a common mechanism to provide quick feedback to students in CS1 and reduce teacher workload in large courses. Autograders like those used in OK (okpy.org), zyBooks (zyBooks.com), Perusall (perusall.com), Autolab (autolabproject.com), and Gradescope (gradescope.com) are already being used by many educational institutions. They provide end-to-end assessment solutions that include code review, data analytics, automated feedback to students, and code completion. Most of these existing software are suitable for well-defined assignments where exact outputs are known and there is an expected code structure. They use unit testing frameworks to check whether the student's code passes a well-defined set of tests and corner cases. However, personalized feedback to students is limited in such autograders. Moreover, instructors design rigid assignments to incorporate autograding abilities. The rigidity needed to make autograded assignments work efficiently tends to limit the student's creative exploration and learning. Thus, there is a clear need for autograder software that uses generative AI capabilities to provide personalized feedback to students and also grade free-form assignments that do not have a pre-defined rigid structure. With such an autograder available, instructors can assess students learning as they develop independent and creative projects.

4.4 Summary

Open-ended projects for summative assessments in CS1 courses push students to explore their interests and be creative in their learning. The level of preparation required for this exam is "High" or a 9 out of 10 on a subjective difficulty scale. The two main challenges with this exam design are (1) coming up with the description of projects, or guiding students to propose an effective project such that the projects are fair for all students in terms of the work required, and (2) the grading. To partially alleviate the first challenge, a list of Python projects is provided on GitHub [2]. A structure with five milestone assignments also addressed the challenge with students feeling lost. It works best if the scaffolds of the projects

are provided in advance, therefore, the total time needed for the project was 5 weeks. Students' critical thinking is highly activated in this project, a 10 out of 10 subjective score. Many students in ME 021 were excited to explore their hobbies and built wonderful Python projects because of the flexibility and the independence that they were offered. For example, a student extracted data from their phone on their social media usage and built a simple activity statistics app. Another student built a detailed database of asteroids and stars in galaxies, with neat visualizations that are found often in related scientific literature. Some of the games built by the students were close in design to actual games like a graphical tic-tac-toe, or a clickable adventure game. Students who put in the least effort ended up with projects that were quite simplistic but still went beyond the assignments in the class, such as an interactive Q&A style prompts and summarized outputs, or a file reading and information extraction. Student group work and collaboration is "Low", or a score of 2 out of 10 since this is an individual project while other conventional projects tend to be team projects so it is a disadvantage of this exam. However, further developments of this exam design could actively intergrate generative AI tools like GitHub Copilot as a pair programmer, which would increase the collaboration score. The independence and self-efficacy development in this activity is "Very High" or 9 out of 10 score since students go out of this exam with a renewed confidence in their Python abilities. The grading involves multiple manual elements — grading of project proposals, the milestone assignments, the oral exam in the labs, and the grading of code in the project (as it is different for each student). An AI-based autograder could address this challenge in the future.

5 Summative: Engineering design projects

For engineering-focused CS1 courses, an engineering design and analysis problem could be used for project-based summative assessment. Since the ME021 class is primarily for students interested in majoring in Mechanical Engineering, we designed project problem statements that were based on some of the core courses in Mechanical Engineering that are required for the students. Using a fun ice-breaker activity at the start of the lecture, I collected the names of courses that the students are most excited about in their mechanical engineering degree. I grouped these course suggestions into four categories: thermodynamics, renewable energy, image processing, and material science. The course instructional staff collaborated for around 4-6 hours to carefully define the problem statements in each of these categories. The students were then asked to choose a problem statement from one of these categories and complete the project that achieves the engineering design goals described in the problem statement. The main advantage with this summative assessment is that students The main advantage with this summative assessment is that the students get a sense of what real-life engineering tasks look like in an industry.

5.1 Engineering design projects

A brief description of the project problem statements is given below:

Thermodynamics of refrigeration: Students were asked to analyze the efficiency of refrigeration (in air conditioning) and correlate these thermodynamic metrics with the costs. Data for the ambient temperatures and costs in Merced was provided to the students. Instead of directly providing a CSV file with clean data, we chose to provide the data as an image. This is a common situation in engineering research where data must be extracted from an image or a PDF. This strategy also limits the use of AI tools that can directly read CSV files and provide the output metrics. Although with more advanced AI tools, reading the data from images is also possible, we asked the students to show their work in extracting this data. Finally, with the data, the students were asked to implement interpolation tasks that involved heavy use of array indexing, iteration, and built-in functions for interpolation. This is another common mathematical operation that engineers use in their daily work in the industry or research. Students were asked to visualize the thermodynamic metrics (like coefficient of performance, thermal energy, work input, and energy costs) in graphs.

A GUI app for efficient solar panels: In this project categoy, students were asked to design a graphical user interface app. Akin to how an engineer would design an app in the industry, a rough sketch of the desired app was provided as a specification. The students were asked to create the GUI based on this specification but were asked to independently explore their creativity in designing the app. Real-world data on solar panel cleaning was provided to the students as a CSV file. The tasks in this project included creating graphical buttons to load the data, to compute metrics related to solar panel efficiency, and sliders for changing various parameters. The graphs were to be displayed in the app itself. Using the GUI app, the students analyzed the data and proposed the parameter choices that would lead to the most efficient solar panels.

Image processing for temperature measurement in LIF images: A project on image processing was designed for students interested in computational aspects of mechanical engineering. In this project, the students explored the singular value decomposition for image compression. Real-world LIF images (one reference, and one data image) were provided to the students so that they can appreciate the power of image processing algorithms in computing physical quantities. Students implemented array indexing for images, loops for pixel-wise operations, functions for image compression, and comparison of compressed vs un-compressed images in this tempearture computation algorithm. The students were asked to visualize the temperature distribution in the LIF images. Similar to the previous two project choices, use of LIF images serves two purposes: first, it is a common real-world task for engineers to interpret images, and second, working with images is a complex task that is not easily solved by generative AI tools but rather suitable for collaborative work with AI.

Design and analysis of materials and composites: The final project category is materials engineering. In this project, data for the stress-strain curves of multiple materials was provided to the students as an image. The stress and strain values were not directly provided to the students but were embedded in the image. The students were asked to extract this data and plot the stress-strain curves for the materials. Real-world data for 10 different materials was provided and students were asked to choose only 4 out

of them but were constrained to choose at least one metal, one polymer, and one composite material. Students were asked to compute tensile strength, toughness, and thermal expansion related properties for the materials that they chose. For thermal expansion, students solved a differential equation using the thermal expansion coefficient values. These parameter values needed to be extracted from the literature, a challenging task for engineers. Students visualized their computations in graphs. Finally, students designed a composite material by constructing synthetic stress-strain data by combining the data for the materials they already had. Then, they studied the properties of this new material design.

The last element of each project included an open-ended exploration task that asked the students to explore at least one element of the project further based on their interest. The full project problem statements are provided as part of our open educational resource on GitHub [2].

5.2 Challenge

Similar to the open-ended independent projects, this project design also suffers from time intensive grading. The grading workload could be minimized by defining the projects in a more specific manner where the expected outputs are well-defined and fixed numbers/equations. However, to promote independent thinking and to counter the use of generative AI tools, we chose to define the problem statement with many open-ended parts where students were asked to show their computational thinking, outlook, and exploration of engineering design.

5.3 Summary

In summary, this summative assessment strategy involves proposing a set of engineering design projects to the students. This is in contrast with the conventional approach where one fixed project is assigned to the whole class. With generative AI tools, such project problem statements might be easily solved by or assisted by AI tools. Moreover, if each student is working on the same project then each student is not necessarily activated at a similar level. The choice of the projects was based on the students' interests and the core courses in Mechanical Engineering, which further increased the engagement of the students.

The level of preparation required for this exam is "Medium-High" or a 7 out of 10 on a subjective difficulty scale. The preparation workload is cut down since this assessment design takes advantage of course staff and their diverse expertise. Each course TA could take lead on a project category, one that is closest to their research. In fact, two out of four projects that we assigned to the students were extracted from the research projects of two of our GSIs. This lowers down the workload, makes the problem statement realistic (and not the same every year), lets students work with real data, and provides active research mentors to them. The main challenge with this exam design is the reliance on the course staff's interest and enthusiasm in designing the problem statement. The grading is not a huge challenge but still significantly higher than conventional grading of a single project.

The total time needed for the project was 5 weeks but a common theme in students' feedback was the limited time they had in exploring the project fully. Students' critical thinking in this summative assessment is highly activated, an 8 out of 10 subjective score. Student group work and collaboration is "Medium-Low", or a score of 4 out of 10. The group work is higher than fully independent project because there are many students who end up working on the same project. The independence and self-efficacy development in this activity is "High" or 8 out of 10 score since students work on real-world engineering design tasks.

6 Summative: Hardware modeling

To-do.

7 Discussion

7.1 Fun activities for small classes

7.2 What works in large classes?

7.3 What does not work in large classes?

8 Conclusion

Acknowledgments

The author would like to thank the students, the graduate student instructors, and academic student employees at UC Merced School of Engineering for their invaluable contributions in defining the in-class activities and the summative assessments.

References

[1] Avaneesh Narla. 2022. Teaching to Fail: Creating Vulnerable Learning Communities to Facilitate Students' Growth. (2022).
[2] Ayush Pandey. 2024. CS1 activities. https://github.com/pyEdTools/CS1-activities

A Research Methods