Interactive Embedded Systems Learning using the Prairie Learn Framework

Design Document

Team Number: 33

Client: Phillip Jones

Advisor: Phillip Jones

Notetaker: Rachel Druce-Hoffman

Technical Lead: Justin Cano

Quality Assurance: Joey Krejchi

Project Manager: Caden Otis

Consultant: Devin Alamsya

Team Email: sdmay25-33@iastate.edu

Team Website: https://sdmay25-33.sd.ece.iastate.edu

Revised: 05/04/2025

Executive Summary

Our project came from a need to minimize instructor grading time in the course CPRE 2880, while maximizing student success. It uses the open source PrairieLearn framework (PL) to provide the structure for a homework solution, including course sections, assignments, homework questions, and options on calculating grades. The CPRE 2880 PrairieLearn project has been in development for three years total by three senior design teams. Each group has made strides in understanding how to set up PrairieLearn to work in the Iowa State environment and translating assignments, which are currently given to students in Word document form, into PrairieLearn questions.

Our client required the use of the PrairieLearn framework to implement the 12 current homework assignments used in lectures as questions, each consisting of Python, C, and HTML code. These questions span the material of CPRE 2880, including GPIO, UART, ARM assembly language, and more. We were required to have as many questions to be graded automatically, without the need for instructor review. Questions were to be broken into component parameters that can be randomized so that students can have "infinite" question variants generated to practice concepts.

The design of this solution centers around PrairieLearn's functionality and compatibility. The course must have a corresponding Git repository, which houses the code, test files, and configuration files for each question and assignment. We have received a Linux machine from Iowa State's ETG which acts as the PL server for this course. Students will be able to access the server through a URL with their email and complete assigned homeworks on the website. We integrated PrairieLearn with the Canvas API to allow grades to auto populate in the gradebook, and with ISU's SSO to provide secure and efficient logins.

Since we are continuing on a project that previous teams have worked on, we iterated on the previous group's design. Their design didn't randomize and/or autograde every question when possible, so we added that to every problem (as much as possible) in our design. There are a total of 12 homework sets, which we worked on in two batches of six. Our previous plan (as of the fall semester) was to deploy the first 6 homework sets in a beta test with current CPRE 2880 students for this semester. That did not come to fruition as we hoped, but this semester's 2880 professor (Dr. Diane Rover) is testing our platform and is distributing it to the course's teaching assistants as part of an Alpha test.

Weekly meetings with our client, the primary professor user, have ensured us that we are staying aligned with their requirements and expectations for this project. These meetings have confirmed that the current progress meets the client's goals. While the impact on student success hasn't been tested yet, the Beta version we plan to be released in the fall 2025 semester will provide critical feedback to evaluate how well our project assists student learning.

The final deliverable of our project consists of nine homework sets, from which every problem is fully autograded, randomizable, and thoroughly tested to ensure it is bug free. We have also created documentation (written and video) to document everything that we learned and have accomplished for this project, hoping to make the onboarding process for the next team as seamless as possible. Our application is ready for future senior design students to take the Beta test in the upcoming school year.

Learning Summary

Development Standards & Practices Used

- ISO/IEC/IEEE 14764:2022, Software Life Cycle Processes - Maintenance
- ISO/IEC/IEEE 42010:2022, Software, Systems, and Enterprise - Architecture Description
- ISO/IEC 27001:2022, Information security, cybersecurity and privacy protection

Summary of Requirements

Requirements that our senior design project must have are:

- All existing CPRE 2880 homeworks implemented
- Questions that are engaging and interactive
- Questions that can be randomized for unlimited practice
- Questions that are autogradable, providing student with quick and specific feedback
- Makes learning difficult CPRE 2880 concepts easier
- Is free for students and professors to use
- Documentation that can teach TAs and future developers how to use and continue developing our project
- Save CPRE 2880 professors and TAs time on grading assignments to focus on other areas within the course

Applicable Courses from Iowa State University Curriculum

The courses that have helped us brainstorm and develop our project are:

- **CPRE 2880:** Since our project is ultimately to make learning easier for CPRE 2880 students, a lot of what we learned from CPRE 2880 is applied to make questions that cover the main concepts from the course and also make the most sense for students. We also need to understand how to get the solutions to the questions we make so that we can create the autograding capabilities for most questions.
- **CPRE 3090:** This course taught software development practices that are useful for a project like this. Our team is using GitLab for source control and planning tasks, which we learned how to use in 3090.
- **ENGL 3140:** The technical writing class is very applicable to a senior design project because of all the documentation involved. It is important to know how to explain technical ideas clearly in writing so that one who reads your writing can understand what you are talking about. A large part of our project is documentation (such as this document) so our ENGL 3140 knowledge has been very useful.
- **EE 2850:** This course taught basic software development using the C programming language, such as the different variable types, method declarations, pointers, structs, and even some recursive algorithms. Our project uses a lot of these basic components of C to create software that can calculate the correct solutions for problems existing in our project.

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your
Going into this project, the skills/knowledge that the team acquired are:

- Using and developing in the PrairieLearn Framework
- Programming in Python
- Using ISU SSO and Google OAuth for authentication
- Setting up a server
- Programming with Canvas API

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Problem Statement

The current way that students in CPRE 2880 learn and engage with concepts presents several challenges that hinder effective learning. One of the main challenges that students face is not getting enough practice of the material due to limited opportunities and resources provided. The students may receive very little feedback on homework submissions due to the TAs not having enough time to provide meaningful feedback for every student in the course. This causes students to be left questioning what they did wrong, preventing any sort of improvement and understanding. When students would like to seek out the professor or TAs to get feedback or to learn more about concepts they don't understand, they are met with limited office hours and other students competing for instructor attention. Additionally, there is limited lab seating and microcontrollers (Cybots) for the students to practice programming, further impeding hands-on learning.

Department faculty is committed to improving the student experience. Instructors have limited time, but it is not due to lack of caring. Professors must balance classes, research, and more, while TAs fit their responsibilities between their own classes. The obstacles the students face are not as simple as poor curriculum or instruction. For many classes, especially these large core classes, there are simply too many students for current teaching methods. Some students will slip through the cracks in these technical courses, but the CPRE 2880 professors are looking to minimize that as much as possible. These obstacles are not unique to this course either; throughout the department and even the whole campus, there are professors aiming to improve student understanding. There are some tools already commonly used, but each has its own flaws and many fall short in technical classes. What is needed is the adoption of a new technology, infinitely customizable, and infinitely randomizable to generate instructor-approved questions for students.

From all of the challenges that students face during their time in CPRE 2880 (and similar courses), it is clear that students need a way to access learning opportunities whenever they want that will provide feedback based on their mistakes. An effective solution would ensure that students can learn at their own pace and receive instant feedback to guide their progress. Additionally, professors and TAs need a way to significantly reduce their time spent on writing questions or grading homework and exams so that they can focus on teaching and offering personalized support to their students. By optimizing these aspects, both students and educators can benefit from a more efficient and effective learning environment.

1.2 Intended Users

The product that we've created is for the benefit of making the CPRE 2880 course easier to learn and manage. The main users of our product will be the students, professors, and teaching assistants in CPRE 2880. Each user of our product has their own unique expectations for the course, and we have made sure to incorporate each of their needs to ensure every user will be satisfied with our product.

The students of CPRE 2880 are usually sophomores who have little to no experience with embedded systems and embedded programming. Depending on their major, they may have never done any type of programming before. They will also experience different types of emotions throughout their time in the course, like feeling frustrated or overwhelmed with some of the more difficult topics taught in the course. We assume that they want to take CPRE 2880 to learn the basics of embedded systems and aim to do well in the course. In order for them to get a satisfactory grade in the course and have a solid understanding of the concepts taught, they will need specific and quick feedback on assignments. This is imperative for students to learn from their mistakes and improve their knowledge. To improve their learning of concepts that they don't fully understand, they will need access to questions that are engaging and interactive. Furthermore, they need questions that can be randomized for unlimited practice. With our product, students will get access to questions that offer much more interaction than what they currently get from their homeworks. Interactive questions will be more effective at capturing student attention, and making sure even unmotivated students engage with the material. Each question hosted from our product will also allow students to create unlimited variants for every question, allowing them to practice difficult questions as many times as they would like. Overall, our product will enhance students' learning in CPRE 2880.

For the professors of CPRE 2880, they are often busy juggling different things, such as their research, multiple classes, and their personal life. These responsibilities make it hard for professors to spend more time explaining difficult concepts and having one-on-one time with students. However, professors want their course to be successful and their students to grasp every concept taught. To help professors multitask, they need homework and quiz questions to have an autograde function to save them time on grading. They also need a way to make concepts easier to understand for their students. This improvement of teaching material will lead to excellent student performance in their course. To help students learn concepts at their own pace, professors also need to provide students with questions that can be randomized. Instead of a static set of questions, students are able to practice until they are satisfied with their grade and their understanding. With our product, we will provide professors with the tools they need to make their course more successful for students.

The TA's for CPRE 2880 are typically juniors, seniors, and graduate students majoring in a similar field and have experience with embedded systems. They are most likely very busy with other schoolwork and research, and depending on their status, undergraduate TAs are only allowed to work 10 hours a week while graduate students are allowed 20 per week. This means they have limited time to help students and grade assignments, which is why they need a way to spend less time grading assignments and focus on answering questions and giving detailed responses to student work. Our project plans to have almost all questions be auto-graded, which will help reduce the time that TAs spend grading each week. This allows the TAs to spend more time grading the non auto-graded questions and allows them to give better feedback to the students.

Finally, we aim for our project to be a model for other courses to follow. We hope to inspire other instructors to revitalize how they teach material and use their time more efficiently. Through the new approach this technology brings, we aim to improve all our users' experience- we do not want a product that assists the teachers but makes students' lives harder. Students, TAs, and professors have very different approaches and goals when it comes to a course. Through our implementation of our PrairieLearn solution, we aim to keep the user experience in mind and develop a product that will satisfy our users' needs.

2. Requirements, Constraints, And Standards

2.1 Requirements & Constraints

To make sure that our project meets the needs of our users, we have defined many requirements of various types. Creating these requirements ensures that we cover all aspects necessary to create a robust, user-friendly, and efficient learning platform.

Our functional requirements include the need to implement all homeworks that have been used for CPRE 2880 in the past, where all document-format questions are coded into the Prairie Learn course. Additionally, all but a handful of questions should be able to be autograded, including those involving student-written code segments. The only questions that won't be autograded are questions that involve paragraph answers, which will need to be manually graded. We also require the randomization of almost all parameters in questions so that each question can be practiced an unlimited number of times. This helps enhance the engagement and learning experience that our project will provide students.

For our aesthetic requirements, it is crucial that our project is free of bugs and typos. This means that when no questions will be confusing for students, nor will any problems occur when a student tries to type in an answer and submit their answers. This ensures a polished and professional appearance, which is essential for maintaining high user satisfaction for the students and professors that will utilize our project for effective teaching and learning.

Our user experience requirements focus on creating an engaging and interactive experience, which is key for our student users. New question types need to be thought of and implemented with a strong emphasis on interactiveness. This is what will set our project apart from others. Questions also should be formatted in a way that is easy for the user to understand and interact with, making the learning process more intuitive and enjoyable.

Lastly, our resource requirements include the implementation of the Virtual/Emulated Cybot interface within our project, allowing students to practice more with embedded programming without having to be in the lab room, especially when there are only a limited number of Cybots available for all CPRE 2880 students. Documentation must be written about each aspect of our implementation to support ongoing development for our project and provide clear guidelines. Additionally, we aim to create tutorials for other classes that want to set up their own PrairieLearn server.

2.2 Engineering Standards

2.2.1 Importance

Engineering standards are important because they ensure safety, reliability, and consistency when designing and creating new products and protocols. Engineering standards are defined protocols that can be followed by everyone because they provide a common language across different engineering disciplines, further ensuring any product in any area continues to follow the safety protocols defined by the engineering standards. This is extremely important because everyone uses multiple products on a daily basis, such as driving a car or using a wifi connection. Consumers need to be able to rely on products, and standards help ensure trustworthy engineering practices.

2.2.2 Standards and Descriptions

The first relevant standard we chose was ISO/IEC/IEEE 14764:2022, Software Life Cycle Processes - Maintenance. This standard defines processes for the maintenance of software throughout its lifecycle. It outlines activities and tasks associated with maintaining software, such as planning, implementing changes, and managing resources. The primary goal of this standard is to guide people to keep maintaining their software which is critical, as it ensures that software remains functional, secure, and up to date, especially as new vulnerabilities or bugs are discovered.

The next relevant standard we chose is ISO/IEC/IEEE 42010:2022, Software, Systems, and Enterprise - Architecture Description. This standard focuses on defining and describing the architecture of systems, software, and enterprises. It sets guidelines for documenting architecture decisions, using viewpoints and models to represent different aspects of the system. Its goal is to provide a structured method for capturing and sharing architectural information, ensuring that all members (or as the document describes them, stakeholders) have a clear understanding of a system's structure and behavior. This helps in making the communication between system design and implementation easier and clearer.

The final standard we chose was ISO/IEC 27001:2022, Information security, cybersecurity and privacy protection. This standard gives a framework for groups to manage the security of their information. It focuses on ensuring the confidentiality, integrity, and availability of our users' information. It also tells us to identify risks to the confidentiality of information and take appropriate actions to ensure the security of it. The intent of this standard is to implement and constantly improve our security system, thereby reducing security risks and boosting the confidence of our users.

2.2.3 Relevance

Software life cycle processes -- Maintenance

As our project has been developed by two teams before us, software maintenance has affected every aspect of our work. We are simultaneously maintaining legacy code, while also creating new code that must remain maintainable for the senior design groups and other Iowa State course developers that come after us. This standard goes in depth on types of maintenance and how problems should be documented. As programmers, we need to document our code and any bugs we find to make it as easy as possible for others to pick up where we leave off.

Software, systems and enterprise -- Architecture description

The architecture design standard is relevant to our project because architecture design is what is used to express the architecture of our project. The architecture of our project helps us to understand the properties of the project we are working on. The architecture descriptions allow us to cooperate and communicate better as we work to integrate all of the architectures of our project. As a team, we want to be thorough in our communication and understand the architectures of our project and we can accomplish that with architecture descriptions. We will make sure to follow the architecture designs that have already been created by Prairie Learn, and we will make sure to create more architectural designs and diagrams that we stick to as the development of our project progresses. We need to follow current architectural designs to ensure that the front-end and back-end of our project will be easy to understand and make changes too. We also want the UI for our project to be intuitive and easy enough for students to understand as they interact with it.

Information security, cybersecurity and privacy protection

The information security standard covers things such as the process for assessing risk, evaluating mitigation effectiveness, security documentation, and improvement. These are things that are certainly relevant to our project, since we will have sensitive information in our application such as student grades and homework answers. It is important to assess the risk our platform has for leakage of such things so that we can come up with solutions to improve the platform's security and prevent issues proactively.

2.2.4 Additional Standards

From everyone on the team, we chose the standards:

- Standard for Configuration Management in Systems and Software Engineering
- ISO/IEC/IEEE International Standard - Software engineering - Software life cycle processes - Maintenance
- ISO/IEC/IEEE International Standard - Software and systems engineering --Software testing --Part 1:General concepts
- ISO/IEC/IEEE International Standard - Software and systems engineering - Software testing -- Part 2: Test processes
- IEEE Standard for Software Verification and Validation

All of the standards we chose are roughly the same, where each standard focuses on one aspect of software and system design and development. We decided to focus on the Maintenance, Architecture, and Security standards as they have the least amount of overlap in the subject matter covered.

2.2.5 Modifications and Incorporation

After reading the standards we found, we have a set of modifications that we intended to incorporate into our product. One such modification is a security risk assessment of the platform, which is outlined in ISO 27001:2022. A risk assessment allows developers to prevent undesired effects, ensure the intended outcome of the product, and help continuous improvement. Risks are anything that affects the confidentiality, integrity, and availability of the platform, and assessing risk is based on the potential consequences of a risk and the likelihood of that risk happening. To implement this modification, we surveyed our product and compiled a list of risks based on these ideas.

Another modification is an increased emphasis on documentation. IEEE 42010:2022, the standard on architecture description, outlines how we are to document the architecture of the system. Our architecture description features system elements, relationships between those elements, the system's relationship with the environment, system behavior, and the principles behind the design. This description will help future developers understand the design of the system and allow for easier improvement.

The IEEE/ISO 14764-2021 standard on maintenance will change our approach to maintenance of our software. We have written code with a focus on readability, adaptability, and scalability rather than just writing code that functions. We have accomplished this through our code conventions and documentation to explain our work. Furthermore, this standard provides terms to classify types of maintenance, such as corrective compared to additive maintenance. Even just being aware of this taxonomy helps us consider the importance and purpose of changes we make to the code base. This approach has helped us organize our tasks and understand how our work fits into the system as a whole.

3 Project Plan

3.1 Project Management/Tracking Procedures

For our Senior Design project, we adopted a hybrid style of agile and waterfall to help achieve our goals. Since the primary foundation for this project was already laid by the previous year's team, our style was partly agile focused as we did iterative development, where each team member worked on different parts of the project. We also received regular weekly feedback from our advisor as we implemented, tested, and improved features. Since we have a structured foundation for the requirements of our projects requirements and design, we also made use of the waterfall style. We had a clear image of what the final project should be and had hard deadlines for certain features to be implemented. This methodology helped us adapt to any unexpected issues that arose as the project progressed, giving us flexibility in meeting our ever-evolving understanding of the project, while also adhering to the wishes of our advisor.

To track our process efficiently throughout the year, we used a suite of tools for project management. This included Git, which is where our project is almost entirely hosted. PrairieLearn has a feature that will automatically sync any changes made to our git repository to the server, allowing for easy modification of code. We also used Git for issue tracking, which helped us organize and manage bugs, tasks, and new features. For team communication, we made use of Discord to coordinate tasks, share updates, resolve issues, and ask for feedback and help. Discord's channel-based organization allowed us to have dedicated channels for different project components to ensure we remained organized.

3.2 Task Decomposition

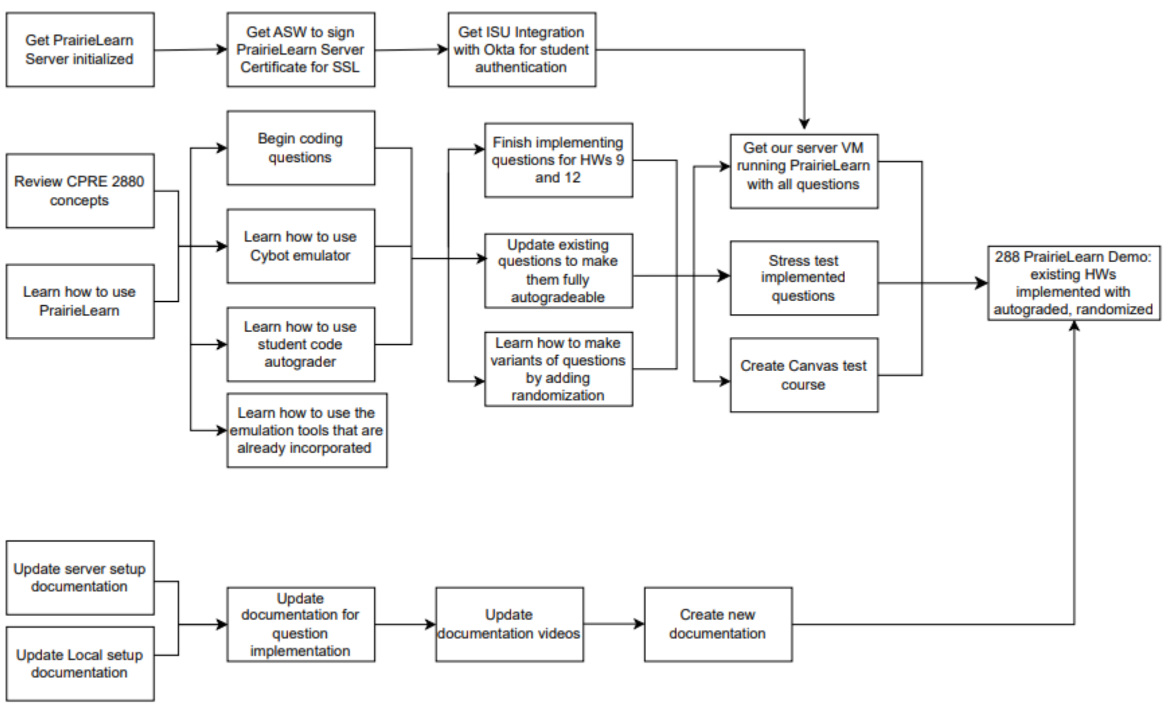

Figure 1: Task Decomposition Diagram

In our task decomposition, we have many subtasks that go into making our finished product. With our project management approach being Hybrid between Agile and Waterfall, we have steps to get to a certain point but then sprints inside of the subtasks in order to complete the project by the desired due date.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

By the end of the Fall 2024 semester, we wanted to reach several key metrics. The first metric that we wanted to reach is to have our PrairieLearn application be completed in a beta version, where half of the homework problems have been implemented into our application, and every question is randomized and autograded. With these metrics planned, we wanted to have a working version of our application ready for use by the CPRE 2880 students in the Spring 2025 semester. That way, we could gain helpful feedback that can better our application. However, we weren't able to complete this metric for the Fall 2024 semester. We were able to implement half of the homeworks (6 homeworks in total into PrairieLearn), but we weren't able to test any of the homeworks to make sure they were fully randomizable and autogradable, and also just made sense for the students.

For the Spring 2025 semester, since we weren't able to get our project ready for a beta test, we changed our metric to create an alpha test before the end of the semester. This alpha test would involve the first nine homeworks, which are fully randomized and autograded, to be tested by Dr. Rover and her TAs. Even though this isn't a beta test, we are still able to get some feedback on our project that we can either implement or give to next year's team. Our next metric is to finish updating documentation from the previous team and complete any new documentation for functionalities that we used or created. This is important for professors that will use our application, as well as any future teams that continue expanding on our project. We also want to have full canvas integration incorporated into PrairieLearn, where grades assigned from homeworks in our application will be synced with Canvas.

3.4 Project Timeline/Schedule

Fall 2024:

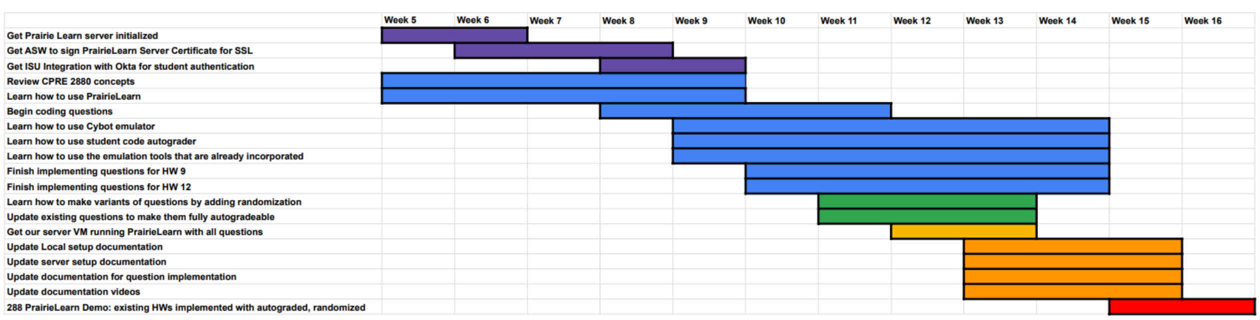

Figure 2: Fall 2024 Gantt Chart

In our Fall 2024 Gantt chart, we detail the tasks and subtasks along with their expected completion dates. The dates are recorded by the weeks in the semester. This Gantt chart and the tasks within detail only the fall semester. We have our tasks coordinated by color with different subtasks within them. The purple blocks show our setup subtasks so that we can use PrairieLearn. The blue blocks show our learning and implementation subtasks. These include things like learning how to actually use PrairieLearn and get comfortable with it and also finishing implementing questions that the previous years team didn't implement. The green blocks show our subtasks for improving upon what is already put in place. This includes randomization of answers so we can create new question variants and make questions graded automatically to give immediate feedback. The yellow block is for our team's server to be set up and running. The orange blocks have subtasks that complete the update of all documentation from previous years. This includes documentation for questions, server setup, videos, etc. And finally, the red block is the end product with our project being completed with all of the homework questions implemented, graded automatically, and completely randomizable.

Spring 2025:

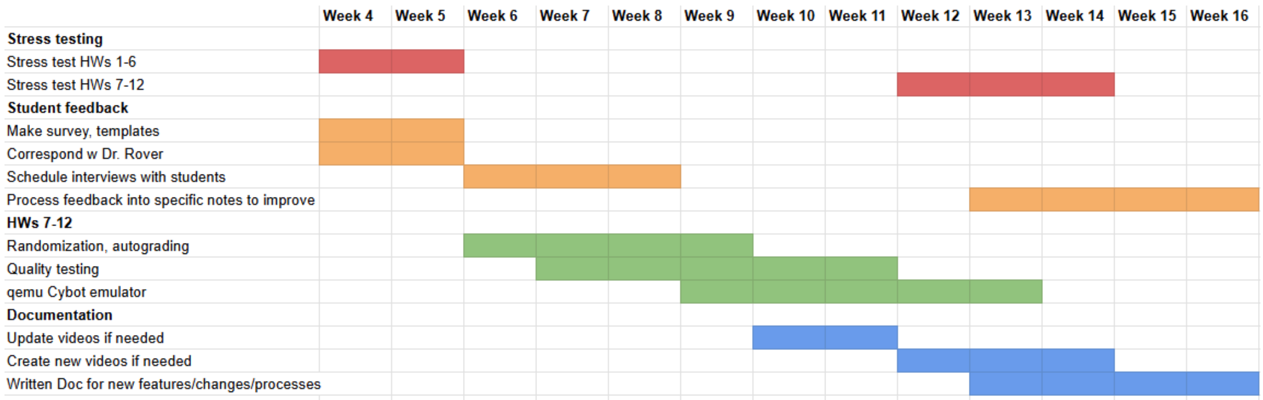

Figure 3: Spring 2025 Gantt Chart

In our Spring 2025 Gantt chart, it outlines our plan of testing the first six homeworks that we implemented in the Fall 2024 semester, and then finishing the implementation of the rest of the homeworks. This gantt chart also outlines us preparing for a beta/alpha test of our project, where we will have a survey prepared to capture feedback from students and/or TAs and Dr. Rover. For testing, we will also need to have a test Canvas course set up to simulate our project working with Canvas and having grades from our project sync to Canvas. Lastly, we outline that we need to update existing documentation and create new documentation to help with the onboarding process of next year's team.

With us using a hybrid style development model with both Agile and Waterfall, our Gantt chart most accurately depicts our plan throughout the year. We will have sprints inside of our tasks and subtasks to ensure that our project is completed. Inside of our overall plan though, we do take a linear approach. With a lot of setup starting first, to then get into learning and creating, then to updating and upgrading our work, to then updating documentation, all ending in a finished product.

3.5 Risks and Risk Management/Mitigation

Get Prairie Learn server initialized

We could run into an issue with ETG where they can't provide a Virtual Machine for us to use as our server.

Probability of risk: 0.05

Outcome of risk: We were able to quickly obtain a Virtual Machine from ETG shortly after requesting one, so no problems occurred doing this process. This allowed us to start working on the setup of our production server right away.

Get ASW to sign PrairieLearn Server Certificate for SSL

Only professors at ISU can submit requests to ISU ASW to sign an SSL certificate request. If a professor won't submit this request, we wouldn't be able to allow students to connect to Praririelearn via HTTPS, meaning malicious actors could snoop on their internet traffic.

Probability of risk: 0.01

Outcome of risk: We gave our client/advisor details on how to request a signed SSL certificate. Within a couple of days, our client/advisor got a response back from ASW with a signed SSL certificate. This allowed to quickly get back to setting up our server to host our application.

<u>Get ISU Integration with Okta for student authentication</u>

Once again, only professors at ISU can submit a request to allow ISU students to use SSO for our application. If a professor won't submit this, students won't be able to sign into our application, making it useless.

Probability of risk: 0.01

Outcome of risk: Instead of Okta, we switched to using Microsoft OAuth as the primary method for student authentication onto our PrairieLearn application. We have had issues for a while with trying to set this login method up, such as Dr. Jones only being able to submit requests for certain actions to be performed with ISU IT. However, we have been making great progress on this issue so far by having constant communication with Dr. Jones.

<u>Review CPRE 2880 concepts</u>

To review the concepts from CPRE 2880, each member of the team needs to take the time (individually) to go through CPRE 2880 material and review concepts that they don't remember. This task can be delayed if members of the team aren't willing to put in the additional time to review concepts from CPRE 2880.

Probability of risk: 0.1

Outcome of risk: There was no issue with each team member reviewing CPRE 2880 concepts on their own. If a team member did have a misunderstanding or lack of knowledge on a concept while working a homework problem, then they would either ask another team member on our Discord channel or ask Dr. Jones during our weekly advisor meetings.

<u>Learn how to use PrairieLearn</u>

It might take more time than needed to learn PrairieLearn if teammates are not putting in the time to dive through PrairieLearn and learn the different services that it provides. This can happen by team members not exploring PraireLearn on their own or not reading through any documentation from the basics of PrairieLearn to how the previous team created their application.

Probability of risk: 0.1

It would further prohibit this task from completion if members on the team aren't communicating and sharing their findings for others on the team.

Probability of risk: 0.1

Outcome of both risks: This risk never became a huge issue for the team. To make it easier to learn PrairieLearn, everyone on the team would dedicate a certain amount of hours each week to play around with and understand the PrairieLearn framework. Then during our weekly advisor meetings, everyone would share their findings and make the learning process go a lot more smoothly.

Begin coding questions

This task can be delayed from the projected timeline in our Gantt chart if team members can't access either the team's server or can't get a local version of PrairieLearn running on their machine

Probability of risk: 0.4

Outcome of risk: Everyone was quickly able to get a local version of PrairieLearn running on their computers. This made it super quick for everyone on the team to dive into PrairieLearn, learn about the framework, and start developing homework problems.

Learn how to use Cybot emulator

This task could become harder to complete in our projected timeline if there is no documentation about the emulator from the previous team, or if the documentation that does exist isn't thorough enough.

Probability of risk: 0.3

Outcome of risk: There was thorough documentation made on the Cybot emulator from the previous team. However, we didn't get much, if any, time to work on the Cybot emulator. So due to time constraints, this risk never affected us.

Learn how to use student code autograder

Similar to the previous task, learning how to use the QEMU ARM autograder could become a more time-intensive task if the previous team didn't write detailed documentation. This is because the ARM autograder was created by the previous team, so the documentation is our main resource for learning about this specific autograder.

Probability of risk: 0.3

Outcome of risk: This risk did have a pretty big impact on our progress for the ARM autograder. The documentation that was made for the ARM autograder did a decent job on explaining how it was created, but didn't do as great of a job on how to actually use the autograder to autograde ARM programming questions. Since this autograder is needed for just one homework, we decided to move on and work on the rest of the homeworks that didn't need to use this autograder.

Learn how to use the emulation tools that are already incorporated

Similar to the previous task, this task can become delayed and difficult to complete if the documentation written for the QEMU ARM autograder is not detailed enough.

Probability of risk: 0.3

Outcome of risk: Similar to the previous risk, we never fully learned how to implement the ARM autograder into PrairieLearn to autograde ARM-based questions Since the documentation wasn't super detailed for this part, we spent a lot of time just trying to get the autograder to work. After many trial-and-error sessions, we decided to move on from this autograder and instead improve on other homeworks,

Finish implementing questions for HW 9

This task can be delayed if team members are not completing their portion of work, not communicating with the team, and/or not showing up to weekly team and advisor meetings.

Probability of risk: 0.4

Outcome of risk: The completion of HW 9 has been going smoothly. Team members have done an excellent job completing work that they say they'll do every week and quickly implementing changes to homeworks. HW 9 is no exception, and it has been completed with full randomization and autograding to meet the needs of our users.

Finish implementing questions for HW 12

Similar to the previous task, this task can be delayed if team members are not completing their portion of work, not communicating with the team, and/or not showing up to weekly team and advisor meetings.

Probability of risk: 0.4

Outcome of risk: As mentioned before, team members have put an amazing effort into the project and are making great progress every week. However, this risk became a bigger issue because of our lack of understanding with how to use the ARM autograder to finish implementing the questions on HW 12. Because of this and time constraints, we weren't able to finish HW 12 and will have to be picked up by next year's team

Learn how to make variants of questions by adding randomization

For some questions, we may struggle to determine parameters to randomize, or with coding the randomization. This requires learning new coding techniques and implementing bug-free questions.

Probability of risk: 0.4

Outcome of risk: This risk never grew to become an issue for the team. While learning how to use PrairieLearn, we were able to grasp the different techniques that we could use to make the formatting of each question unique and engaging to students. We also would ask our advisor for ideas on how we could improve the randomization of questions if we were ever stuck.

Update existing questions to make them fully autogradeable

This task can be delayed from our projected timeline if we can't get the C autograder or the ARM autograder to work as expected.

Probability of risk: 0.2

This task can also be delayed if team members are not completing their work and doing their portion of the autograding for certain questions.

Probability of risk: 0.3

Outcome of risk: This has not been a huge problem in the development of our project. Even though we weren't able to figure out how to use the ARM autograder, most questions rely on the built-in C autograder for autograding questions. The team was able to vary quickly learn how to use the built-in autograder, and allowed us to autograde all questions on the HWs that we've completed so far.

Get our server VM running PrairieLearn with all questions

We could get behind on schedule with implementing all questions on the server if team members have not contributed to implementing all questions from HWs 9 and 12 into PrairieLearn

Probability of risk: 0.3

We could also be prevented from implementing all questions on our server if we can't create our own server from ETG

Probability of risk: 0.05

Outcome of both risks: There hasn't been an issue with team members not contributing to the project, nor has there been a problem with ETG. Team members have made adequate progress on each HW of our project and our server has been up and running almost perfectly since the beginning of our project. All questions that have been implemented are currently on PrairieLearn and can be accessible by anyone, there just isn't a reliable authentication setup to allow only ISU students and staff to login to our project.

Update Documentation (Local setup, server setup, question implementation, videos)

All documentation could be hindered by the time necessary to develop a cohesive visual design standard.

Probability of risk: 0.2

All documentation could be hindered if previous team documentation is missing more detail than originally evaluated.

Probability of risk: 0.3

All documentation could be delayed if a team member does not complete their portion of the work timely, or their work is not up to standard.

Probability of risk: 0.1

Outcome of all three risks: To make sure that we have enough time to update and/or create new documentation, we decided to prioritize the second half of the Spring 2025 semester for working on the documentation. By making this plan early, team members put in the time and effort to get most of the design and HW implementation done before the second half of the semester so that they could shift their focus to documentation.

2880 PrairieLearn Demo: existing HWs implemented with autograded, randomized

Product may perform worse than Canvas in beta testing.

Probability of risk: 0.5

Risk mitigation plan: Use student and professor feedback to optimize PrairieLearn during the spring semester, reworking whole sections if necessary.

3.6 Personnel Effort Requirements

Estimated efforts:

Task	Description	Effort (Hours)
Planning	Initial planning of our project. Understand the needs and requirements of our client. We've allocated 10 hours to ensure we all have a solid understanding of what we need to make.	10
Research	Perform product research on alternative products, view the previous team's project and gain an understanding of it. 15 hours should be enough for us to research other options as well as gain an understanding of how the previous team's project works.	15
Learning software	Spend time learning how to create questions, randomize answers, and auto grade them. 15 hours should be enough for each member to understand how PrairieLearn works and how to use it to create questions.	15
Server Setup	Get the server that PrairieLearn will be hosted on setup and ready for hosting. This task mostly relies on us waiting for members of the ISU IT team to get back to use, so we allocated 15 hours.	15
Finish implementing questions	Finish implementing homeworks that last year's team didn't finish. We only needed to implement homeworks 9 and 12, so we allocated 20 hours.	20
Get questions autogradable and randomized	Modify last year's teams questions so they are auto-graded by PrairieLearn and randomized so students can have multiple attempts. This task is possibly the most daunting of them all, so we allocated the most amount of time, 35 hours.	35
Bug testing	Perform testing of our project to ensure no bugs will harm our users' experience. Since we don't know how many bugs we'll have to fix, we allocated 15 hours just to be safe.	15
Updating documentation	Some parts of the previous team's documentation is either inaccurate or needs to be updated. As such we allocated 20 hours just to updating documentation and videos, to ensure that those who follow us could easily set up a PrairieLearn course.	20

Final demo	Have a working demo of all homeworks, with each question being auto-graded and randomized by the end of the semester. Since everything should have been done in previous tasks, we allocated 10 hours just to making sure our demo is ready.	10

Table 1: Estimated Effort Requirements

Actual efforts:

Task	**Description**	**Effort (Hours)**
Planning	Initial planning of our project. Understand the needs and requirements of our client. We took more time on this task to really understand the needs of our client and users.	20
Research	Perform product research on alternative products, view the previous team's project and gain an understanding of it. Similar to the estimated efforts, we did spend about 15 hours researching other options as well as gaining an understanding of how the previous team's project works.	15
Learning software	Spend time learning how to create questions, randomize answers, and auto grade them. Throughout the entire year, we would learn new aspects of the PrairieLearn framework, so this task was done parallel with our other tasks.	30
Server Setup	Get the server that PrairieLearn will be setup and ready for hosting. Due to complications of getting Microsoft OAuth to work, the final completion of our server took a lot longer than expected.	35
Finish implementing questions	We thought that we just needed to implement questions for homeworks 9 and 12, but all other homeworks were in an incomplete state as well. This required us to thoroughly test each HW and make adjustments to the questions.	35
Get questions autogradable and randomized	Most of the questions implemented by the previous team didn't have either great or no randomizing and/or autograding functionality. Because of this, we spent a lot more time than expected making sure each question of each HW was randomizable and autogradable, and fixing questions that didn't meet these requirements	45
Bug testing	Perform testing of our project to ensure no bugs will harm our users' experience. There were a decent	25

	amount of bugs/imperfections in the HWs implemented by the previous team than was expected, so we spent longer than planned on this task.	
Updating documentation	Some parts of the previous team's documentation is either inaccurate or needs to be updated. We also learned quite a bit about PrairieLearn that hadn't been documented already, so we had to create more documentation than what was expected.	25
Final demo	Have a working demo of all homeworks, with each question being auto-graded and randomized by the end of the semester.	10

Table 2: Actual Effort Requirements

Observations:

Comparing our estimated efforts with our actual efforts, we first spent a lot more time planning our project design. We also spent more time than expected learning how to use the PrairieLearn framework because of how vast and complex the framework can be. Because of our issue with Microsoft OAuth, we spent quite a bit more time on getting our server fully set up and functional with PrairieLearn. Implementing questions and making them autogradable and randomizable also took a lot more time and effort than we expected because we had a different idea compared to the previous team of how the HWs should look and function. We had a big emphasis on having each question being randomized as much as possible and having full autograding capabilities so the students could get instant feedback.

3.7 Other Resource Requirements

Time, Canvas page, Server, youtube channel, VMs on personal computers

Many of our remaining resources were more intangible. For help with development, we used the PrairieLearn git forum and the PrairieLearn Slack server to learn more about the PrairieLearn framework. Being able to read posts from other developers or even pose our own questions will help us when we get stuck. Furthermore, we also consulted the official PrairieLearn documentation, the previous team's documentation writeups, and videos that they made on the team's youtube channel for guides on how they implemented questions and used PrairieLearn.

To be able to run and interact with the PrairieLearn framework, we needed to install VMs onto our personal computers to run local instances of PrairieLearn for question development. We also needed to use a VM (obtained from ETG) to host our project so that our users can login and access the different HWs.

To be able to test our project and make sure it works well with Canvas, we needed to set up a test Canvas page to see how well grades obtained on PrairieLearn can be synced to the gradebook in Canvas.

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Examples
Public health, safety, and welfare	Our application is beneficial to the public in this respect by helping to reduce stress on students enrolled in 2880 by providing a better learning environment. Student grades will improve rather than degrade if our application works as expected. It also relieves the burden placed on TAs and the instructor by autograding assignments, preventing manual labor needed to grade. The application is also fair to all students by giving randomized questions, which in turn will curb copying and academic dishonesty.	Reduced student stress, reduced burden on TAs and professors, improved learning environment that is equally available to all students.
Global, cultural, and social	We believe our application will help 2880 students learn material and prepare them for their future careers. Students should have a better learning experience than they had with previous teaching methods as well. It is also tailorable to an instructor's needs because the design can be modified at any time through simple code changes. All students will have equal access to the platform as well helping to level the playing field for time or resource-challenged students.	Increased student learning, better-prepared graduates, more time for TAs and professors to do more important tasks.
Environmental	If our homeworks are online, it saves paper from being used for physical copies, reducing the number of trees that need to be chopped down for paper. Also, since our servers are hosted by Iowa State on campus, they are sourced by more renewable energy than if they were hosted elsewhere. These are a few ways our application has a positive impact on the environment.	Decreased paper production, decreased landfill use, decreased use of non-renewable resources
Economic	Our application is free for all 2880 students to access, unlike some platforms used in other courses that may require a subscription or access key. This will prevent students from having higher tuition bills than they already have every	Decreased student fees, decreased student tuition, more successful students that make more money

	semester. Our application also serves as a template for other courses to make something similar which saves students money for those courses for the aforementioned reason. And additionally, this course will be free to all students, so no specific group will be left out because they cannot afford it. Overall, these aspects will allow our project to create a positive impact rather than a negative one.	

4.1.2 Prior Work/Solutions

One of the first steps for our project was research into similar products that are on the market. The most similar products we found were Coursera, zyBooks, LinkedIn Learning, and Quizlet. All of these are teaching digital teaching platforms with features that make them unique. Quizlet simulates flashcards and allows users to create card sets that they share with friends and the community [2]. LinkedIn Learning focuses on teaching career skills [1]. ZyBooks simulates the feel of a textbook but with the addition of interactive features [3]. Coursera is similar to LinkedIn Learning but with a bigger catalog [4]. One thing that all these programs have in common is that most of their functionality is behind a paywall.

There are a lot of advantages to our platform. For one, it is tailored to CPRE 2880 and eventually integrable with Canvas. It is also open source and changeable by the course instructors. The questions are robust and autogradable, and it is free to students. These are the advantages our application offers. There are some definite drawbacks as well, however. The system will likely have bugs since it's made by students rather than professionals. It also requires technical skills to make new questions since they are written in Python and C. It also requires some work for TAs and instructors to learn the platform.

Something worth mentioning is that we inherited this project from two previous senior design teams. This introduces some unique aspects to our project compared to teams who started fresh. Some aspects of the platform were already complete, which saved us a lot of time. There is also a somewhat extensive documentation repository that answered a lot of our questions about the project. However, there were some drawbacks to our situation. It was sometimes hard to understand the previous group's work, especially when the documentation is missing. All the easiest aspects of the project are already done as well, leaving us with more challenging work.

[1] Coursera, "Coursera | Online Courses & Credentials by Top Educators. Join for Free," *Coursera*, 2024. https://www.coursera.org/
[2] "zyBooks - Build Confidence and Save Time With Interactive Textbooks," *zyBooks*. https://www.zybooks.com/
[3] LinkedIn, "LinkedIn Learning: Online Courses for Creative, Technology, Business Skills," *Linkedin*, 2025. https://www.linkedin.com/learning/
[4] "Quizlet," *Quizlet*, 2023. https://quizlet.com/latest

4.1.3 Technical Complexity

Internally, our platform has a huge set of applications that work together to support it. Our application is built with the PrairieLearn framework, an open source framework for educational platforms created at the University of Illinois. The OS running our PrairieLearn server is Linux, which is contained inside a Docker container. Using Docker allows for easy setup of the application keeps the program isolated from the rest of the system. The Linux server is hosted as a virtual machine on one of Iowa State's servers. This server is accessible via HTTPS and SSH.

Inside of the application is a set of homework assignments. Each assignment is composed of Python scripts, an HTML file, a JSON file, and potentially C code depending on the type of question. These components are altered for each question to implement the design desired. The content of these files is specified by the PrairieLearn framework and necessary for a functioning application.

One of the more interesting technical aspects of our platform is the autograding functionality. Simpler questions (such as math-based and fill-in-the-blank questions) are graded using lines of Python code. More complex questions, such as ones where users are required to write C or Assembly code, are graded by isolated autograder programs. These programs run separately from PrairieLearn in their own docker containers and are given "jobs" through shared folders in the base Linux server. The PrairieLearn foundation provides a C autograder that compiles and runs C code inside the container and compares its output to the desired output. This is good, but for our project, we want to emulate an actual CyBot running code, so we have containers developed by the previous team that have virtual machines emulating the actual CyBot hardware. This way, the student can have an experience that is the same between our application and the physical lab.

There is also notable external complexity in our project. It is not easy to design questions that are truly engaging for a user. It also is difficult to find ways to randomize questions, especially when they involve code. We also are writing questions on material that we have not learned in several semesters. These are some of the things that make this project difficult.

4.2 Design Exploration

4.2.1 Design Decisions

Our first key design decision that we made for our project was to implement auto graded questions. This enables instant feedback for students while saving professors and TAs time on grading. This decision is crucial as it supports one of our goals of providing students with immediate feedback about their assignments, while also allowing professors and TAs to focus more on teaching rather than grading. Another key design decision was to incorporate randomized questions by using randomized parameters within questions, making each attempt unique for students. This is essential, as it ensures students have unlimited practice opportunities with varying question parameters, allowing them to gain a further understanding of core concepts. Finally, our last key design decision was using Prairielearn as the project's main platform, as it allows for us to use question randomization and autograding. Choosing Prairielearn for our project allowed us to utilize its fully modular capabilities to create questions that are randomly generated with randomized parameters that can be automatically graded by Prairielearn, unlike alternatives like Canvas.

4.2.2 Ideation

Options to autograde questions:

- Option 1: manual grading by professor or TAs
- Option 2: not asking code questions that can't be autograded
- Option 3: write own compiler + unit tests to run in PL
- Option 4: all multiple choice
- Option 5: use PrairieLearn's built-in C and Python autograder
- Option 6: create our own custom autograder container in addition to PrairieLearn's version
- Option 7: make harder autogradable questions, like programming ones, graded based on participation

- Ex: if you submit code, no matter if it's right or wrong, you will get participation points for that question

The decision we had the most choices with was the goal of having almost all questions able to be autograded. Although an achievable goal, we had to determine if it was worth the risks and costs, and if so, how it should be implemented. The obvious alternative is to allow manual grading, and forego our goal. Or, we could remove all questions that cannot be simply autograded- mainly paragraph response and student-written code questions. We also considered adapting them by taking the core concepts of these questions and turning them into multiple choice, or other format, questions.

Autograding the C and ARM coding questions is the most complex part to implement. Some choices for achieving this included using PrairieLearn's built-in autograders for C and Python, or using these versions in addition to adding a custom autograder container. If given more time and materials, we could even have decided to build our own compiler and unit testing solution from the ground up to run in PrairieLearn. Finally, we could have utilized a "challenge question" approach, where more difficult programming questions are asked, but students receive participation points as long as they tried in good faith.

4.2.3 Decision-Making and Trade-Off

To identify the pros and cons of which options to use based on our list of brainstormed options, we had to think about our ultimate goal with our design and user needs. We want to have questions that are autogradable so professors and TAs don't have to worry much about manually grading assignments, but we also want to develop complex questions that are unique and engaging for students.

	Pros	Cons	Trade-offs
Option 1	• Professors and TAs have more say in how a question gets graded • Don't have to worry about making tests and making questions autogradable	• Doesn't need our professor's and TA's need of questions being autogradable • Students won't get instant feedback when answering a question • Answers can be mistaken and points can be wrongfully deducted	By picking this option over the rest, we are giving up: • saving time for professors and TAs • Instant feedback for students • Incorrect grading by human error for: • More say in how a problem gets graded • The convenience of not having the extra step to autograde questions
Option 2	• Saves time in developing new strategies for autograding difficult coding	• Can only ask coding questions in a certain format, which can hinder uniqueness	By picking this option over the rest, we are giving up: • Uniqueness of problems for:

	questions ● Don't have to worry about creating custom autograder containers for questions that PrairieLearn can't handle		● Not having many autograder problems
Option 3	● We know the whole process from submitting an answer to how it gets autograded ● We can customize how questions get autograded	● Would take a lot of time to get working properly	By picking this option over the rest, we are giving up: ● Time that we could use to develop and improve problems for: ● Making our own method to autograde questions
Option 4	● Easy to autograde ● Will have almost no autograder issues or debugging	● Questions will only be in a multiple-choice format ● Hinders the engagement of questions	By picking this option over the rest, we are giving up: ● Problems that our engaging for students and encourage learning for: ● No difficulties with making problems autogradable
Option 5	● This is great at autograding C code submitted by students ● Also works for calculation based questions	● Does not support ARM assembly language input ● Does not simulate the actual hardware that students use in lab	By picking this option over the rest, we are giving up: ● ARM assembly question autograding ● Simulations of real 2880 hardware for: ● Ease of implementation
Option 6	● More types of questions can be autograded, such as ARM assembly questions ● We'd know how to use the custom	● It would take some time to create a functioning autograder container	By picking this option over the rest, we are giving up: ● Time that could be used for problem development or improvement for:

	autograder		• More variety in questions that can also be autograded
Option 7	• Easier question design since you do not need to write code to grade the question • All students will get points if they attempt the question, regardless of correctness	• This would not provide feedback to students to let them know if their solution is correct • Students may submit low-effort and non-functioning solutions.	By picking this option over the rest, we are giving up: • Instant feedback to students. for: • Easier question design and participation grading.

4.3 Final Design

4.3.1 Overview

Our current design is a server with PrairieLearn running as a Docker container. Users access the server through their web browser with the server URL. Inside PrairieLearn, we have a list of homework assignments and their respective questions. A question is composed of a JSON file with basic information about the question, an HTML file that structures the visual component of the question, and a Python script that supports the internals. Questions can get autograded through either the Python script or an autograder image that checks programs written in C or ARM assembly. TAs can view student responses and provide feedback if answers are not autograded.

4.3.2 Detailed Design and Visual(s)

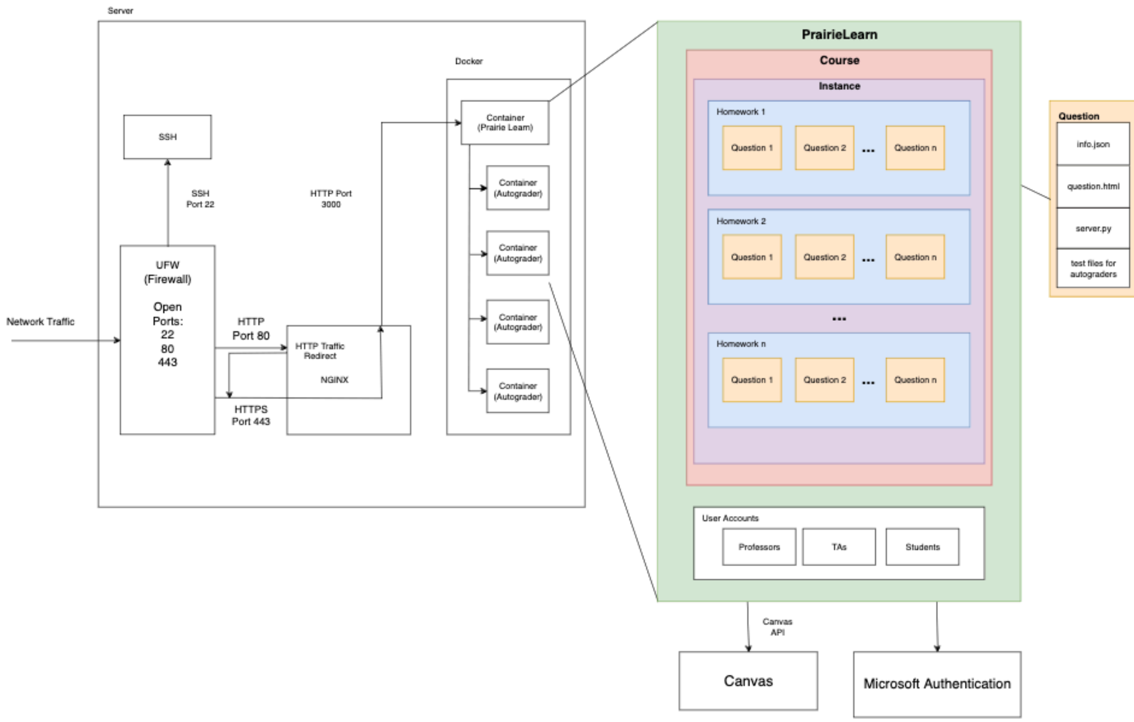

Figure 4: Block diagram of PrairieLearn components

The server hosting our PrairieLearn application is an Ubuntu 22.04 Linux machine. It has port 22 open for SSH which allows us to access the server for maintenance. Ports 80 and 443 are also open for HTTP/HTTPS traffic to allow users to access the site through the URL. User traffic is redirected with NGINX to port 3000, the PrairieLearn application port. PrairieLearn is hosted in a Docker container, which functions like a virtual machine. The ARM assembly and C autograders are separate containers connected to the PrairieLearn container through a socket and input/output directories.

PraireLearn is structured with courses with instances. An example of how those are used is "CPRE 2880" as the course listing and "Spring 2025" as the instance. Inside each instance are homework sets with individual questions. Each question has a "info.json" file that includes a unique ID, a title, a category, additional tags, and autograding options. Questions also have an HTML that allows the designer to create the visual components of the question, like diagrams, prompts, and entry boxes. There is also a "server.py" that randomizes question parameters and grades the question if it is not programming based. If it is programming based, there is a separate folder called "tests" that has a correct program and scripts to initialize the autograder.

There are three types of users for PrairieLearn: students, TAs, and professors. Students can view their assignments and submit answers. TAs can view those answers and provide feedback if necessary. Professors can design and modify the questions/course. Users login with their Iowa State credentials through Microsoft Authentication.

PrairieLearn is also integrated into Canvas so that grades are automatically synced and the PrairieLearn application window is embedded into the Canvas site. By integrating Canvas into our project, it makes it easy for student grades to be tracked both in PrairieLearn and in Canvas.

4.3.3 Functionality

Our design would ideally become a staple in CPRE classes. Professors with little technical skill would be able to set up a virtual machine to host a PrairieLearn instance just by following our documentation. Professors, TAs, or senior design groups could then implement course specific questions. Both professors and students are able to log in with an account connected to their university account, making access restriction easy. Professors can then release a course for their students to access, customize which questions will be graded, and publish assignments.

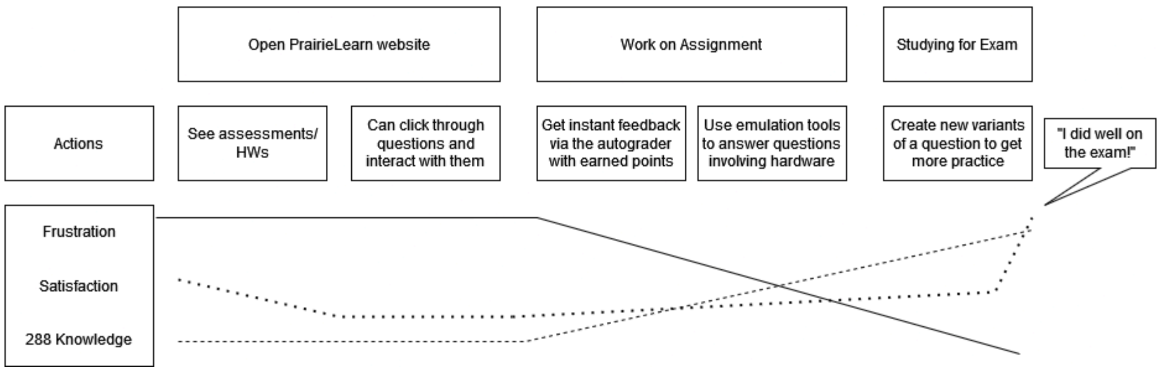

Figure 5: Journey Map describing student use of PrairieLearn

PrairieLearn should be an effective platform for students to complete homework assignments and practice concepts. Questions are more than just short answers or multiple choice- we implemented questions that are more interesting and require more effort so students must think through each question, even if they have seen a variant of the problem before. We aim for our solution to become a standard for a flexible engineering homework platform that teaches students well. Our design should have minimal bugs and cause minimal frustration as students work through problems. Courses interface with Canvas to automatically update student grades. Overall, the effect will be reduced stress on professors and students, reduced time spent grading, and improved student test scores.

4.3.4 Areas of Challenge

At the moment, our platform is well developed and ready for Beta testing. However, it did have some noted faults we had to work through. One is that all homework questions were not implemented or are implemented improperly. There were also more manual graded questions than preferred, since manually graded questions create more work for TAs that we wish to avoid. Additionally, many questions were simple and needed to be more engaging for users. We overcame these issues by going through each homework set, testing the problems, taking notes on possible improvements, and then implementing those changes. We then tested each other's work and made additional improvements as necessary.

Another struggle that we had initially was in inheriting a project from a previous design team. It was hard to jump in on a technical project that has been ongoing for two years. We overcame this by carefully reading the documentation from the previous years, and we have created new documentation that should help the future design team/s begin their work.

4.4 Technology Considerations

The technologies used in our design include Prairielearn, QEMU ARM emulator, and Git. Our first technology, Prairielearn, integrates well with our project as our goal is to create a dynamic, customized, interactive learning environment. Through use of its completely customizable questions, we can provide auto grading and question randomization, which will help students learn through practice. However, Prairielearn does require a certain degree of technical skill to program a course in Prairielearn. In order to create questions in Prairielearn, there's no simple interface you can use, you have to hard code it from scratch, which could dissuade potential users. Another option we could've used is the technology that was originally used, Canvas. Professors are already familiar with it, and it's easy to create homework assignments. However, the autograding abilities of Canvas are mediocre and the question customization is almost non-existent, meaning Prairielearn is better for our project.

Our next technology was the QEMU ARM emulator, which enables hardware emulation of a TM4 microcontroller. This technology allows for students to interact with embedded systems without needing a physical device such as the lab robots. Some drawbacks of this is the difficult setup of the emulator and the maintenance required to ensure compatibility with Prairielearn. An alternative to using the QEMU ARM emulator would be to use the Cybot emulator which was made for CPRE 2880. Since it was specially made for this course, it has all the features that the QEMU ARM emulator has. However, it has less documentation than QEMU while also having issues syncing to Prairielearn, meaning that the QEMU ARM emulator was the better option for us. However we haven't given up on using the Cybot emulator, and we have hopes the future design group will finish the work we and previous design teams have started.

The final technology we used was Git. Prairielearn has the ability to pull courses from Git repos, meaning any code we've worked on for our project can be pulled by the server whenever a new update is pushed. It also helps us with tracking code changes and managing our project as a whole. The only downside of Git is that it could be challenging for professors unfamiliar with Git. An alternative to using Git would be to just host the course files on the Prairielearn server, but this makes it harder to make changes to the course, as you will have to manually upload the files. For these reasons we chose to use Git to host our course files.

5 Testing

5.1 Unit Testing

The main thing that our project consists of are homeworks that contain engaging questions to help students learn tricky-to-understand concepts throughout their CPRE 2880 journey. Because of this, we wanted to test and make sure that each question of each homework makes sense to students, has an easy interface for answering the questions, autogrades questions correctly, gives the correct solutions, and provides students with feedback that helps them understand why they got a particular question wrong. We also wanted to test that the server that hosts our application of homeworks has been set up correctly so that students and professors can access our project very easily via ISU SSO.

Before deploying our project to CPRE 2880 professors and students, we stress tested each homework by going through each question to make sure questions have a good format, are consistent, provide feedback, are randomizable, and autograde the questions correctly. There was no special tool that we needed to test our project, just continued to use a local version of PrairieLearn on our machines to perform these tests.

Once those tests were finished, we had Dr. Rover and her TAs experiment with our project. They were tasked with interacting with the homeworks and answering questions to see if things made sense for them. They were also tasked with using the PrairieLearn application as if they were viewing and/or manually grading HWs, and reading guides that we wrote to better understand the

PrairieLearn framework. They then provided us with feedback on what went well, what didn't go so well, and what could've been improved during their experience. This feedback will be used to improve on our project, which will be implemented by next year's team.

5.2 Interface Testing

Within our project design, there are two interfaces that will be used by our intended users. The interface that students will interact with is the "student view". For each question of a homework assignment, this interface will show students the question, the number of points of the question, what they missed upon submitting their answer, and feedback to help them understand what they missed. CPRE 2880 professors and TAs will interact with an interface called the "dev view". In this interface, professors and TAs will see the correct answers for each question, can regrade questions that students have answered, and can access the software that creates a question and its correct answers, therefore allowing them to modify questions or create new questions.

In section 5.1, it similarly describes how the student and professor interfaces were tested by current professors and TAs. We were first able to view each interface on our side and stress test everything to make sure things work as expected. Once we felt confident that everything worked as expected, we then had a professor and TAs experiment with our project and give us feedback on what they liked and didn't like. The feedback that we received will then be used to make improvements by next year's team.

5.3 Integration Testing

The most critical integration path in our design is getting our project to connect with Canvas so that assignment grades on our application will be automatically synced to the gradebook in Canvas. This will allow professors and TAs to not have to manually enter grades into Canvas, thus saving them time and allowing them to focus more on helping their students. This integration was tested by making a mock/test Canvas course, and then seeing if points earned in our project were able to sync to assignments made in Canvas.

5.4 System Testing

The PrairieLearn system was tested by testing each question of each homework to make sure that they work well on the student's end, where each question has a good set of randomizable parameters and autogrades correctly. Then, we tested if grades/points obtained by students in our application sync to the corresponding assignments made in Canvas. By using this testing method, it ensured that all aspects of our system work.

5.5 Regression Testing

To ensure that any new additions added don't break the old functionality of our project, we created thorough documentation to explain everything that we've developed thus far. We also performed rigorous testing of each homework to make sure all questions had proper functionality within our project. A critical feature that we ensured doesn't break our project is that every component of our system is scalable. By having everything be scalable, this allows for anything that we've already developed to be adjusted in almost any way, especially if next year's team decides to make any improvements to the stuff we created. Since PrairieLearn is a constantly growing framework, it is almost guaranteed that older functionality will need to be improved upon, thus having scalability is driven by PrairieLearn requirements.

5.6 Acceptance Testing

We have demonstrated that our functional design requirements have been met by ensuring that each question of all homeworks has a consistent and understandable format, are randomizable and autograded, and provides instant feedback upon missing a question. We also demonstrated our non-functional design requirements have been met by showing our advisor new changes/additions that we've made to the user interface and system properties. We have made sure to involve our client/advisor in the acceptance testing by giving them weekly demos on our findings and improvements, allowing the client to see all functionalities of our design and provide us with feedback of their own.

5.7 User Testing

Security is an important aspect of our project, as we don't want students and professors to have their data compromised, nor do we want unexpected people to hack into our project and mess with our software. To make sure only ISU professors and students can access our application, we are using encryption and authentication features handled by Microsoft OAuth and ISU SSO. This also allows us to protect user information by the security measures built into Google and Microsoft. To prevent traffic from being mishandled, we are using HTTPS to encrypt the traffic between the users and our server.

To secure the server, we use SSH with public key authentication, implement multi-factor authentication for password access, and configure a firewall that only permits traffic through SSH, HTTP, and HTTPS ports. NGINX is employed as a reverse proxy for the PrairieLearn server, enabling HTTPS and encrypting client-server communication. Furthermore, all HTTP traffic is redirected to HTTPS to ensure encryption across the board.

5.9 Results

By going through all of the questions implemented by the previous team and checking to see if each question is randomizable and autogradable, we encountered quite a few problems with a bad user interface and/or missing features that make the problems confusing. For example, Figure 4 below shows a question that was created by the previous team. For this question, there was no randomization or autograding present. This problem only accepted short answer responses from students, which required the problem to be manually graded, which goes against our design ideals. So we improved this problem by switching from short answer responses to multiple choice questions, as shown in Figure 5. By having students select from different choices, this allowed us to autograde the problem since we were now easily able to assign correct answers. Due to the nature of the problem, we weren't able to make this problem randomizable, but our advisor was okay with this problem just being autograded.

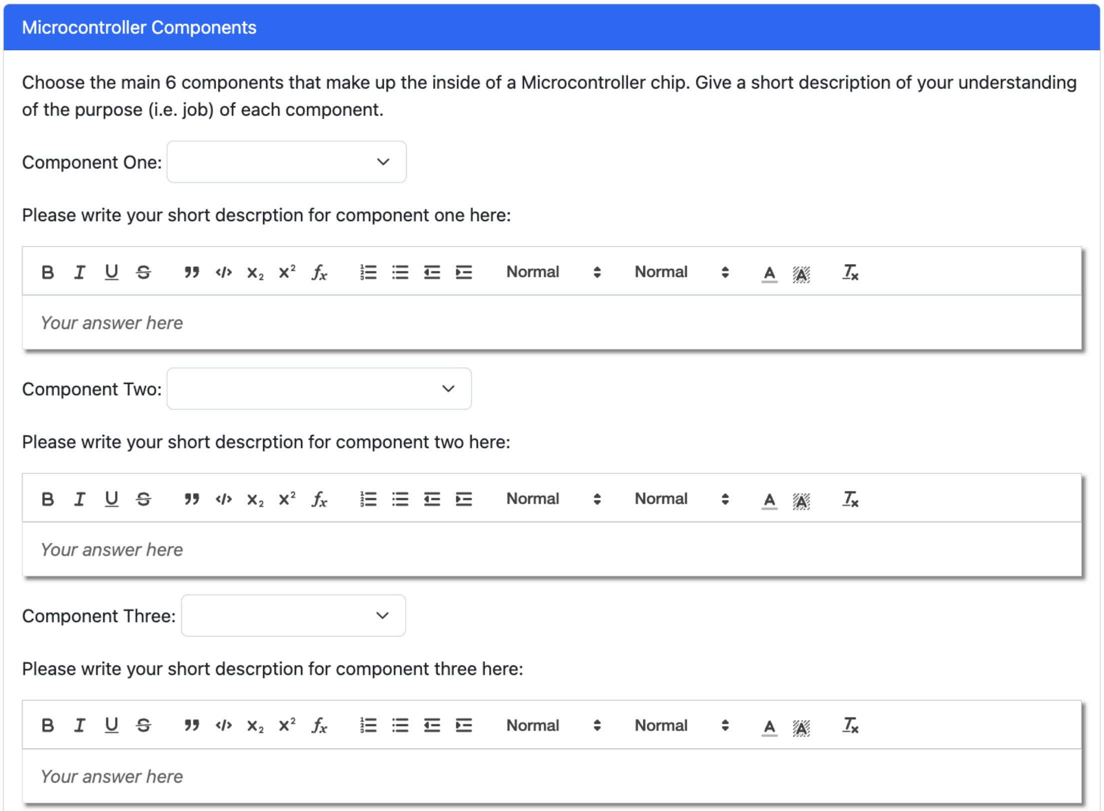

Figure 6: Old problem that has no autograding or randomization

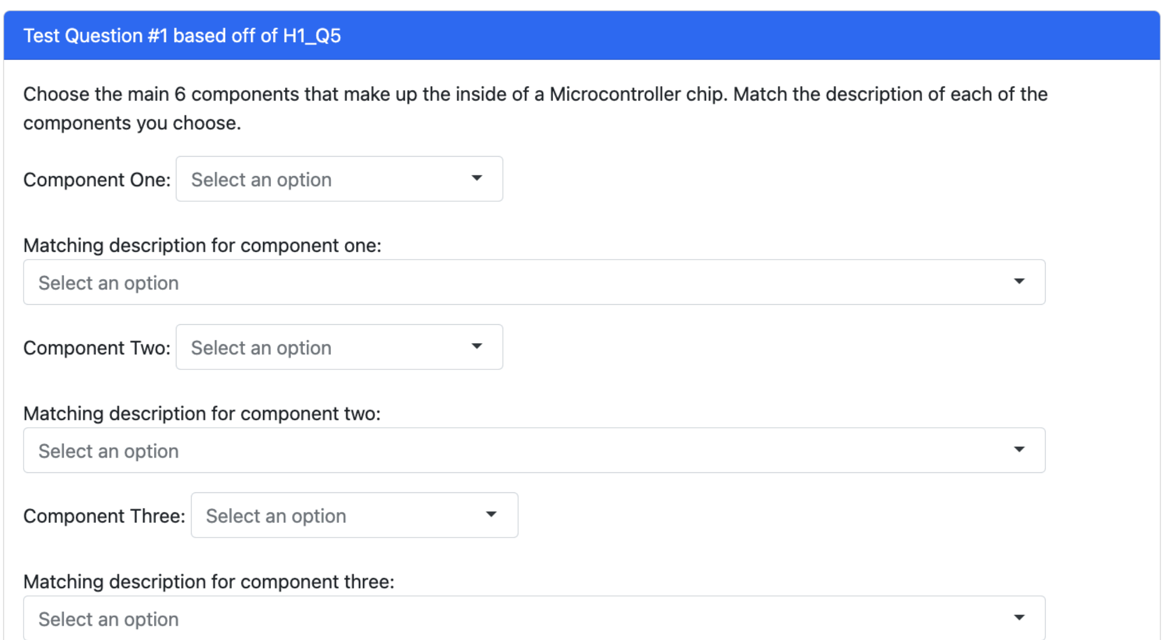

Figure 7: Improved problem that uses autograding

6 Implementation

The final implementation of our project contains a series of 12 homework assignments that run on the PrairieLearn framework, where we have a production server that hosts our project so that students, professors, and TAs can access our project with a secured login via Microsoft OAuth. Most questions within our project are similar to those used in previous CPRE 2880 courses, but we expanded off them to make sure that they are randomizable (to allow students to get a lot of practice out of each question) and autograded (to give students instant feedback on mistakes made).

The final implementation of our project mostly matches our final design. We were able to fully implement HWs 1-9 with each question being randomizable and autogradable. However, we weren't able to finish implementing the rest of the HWs, HWs 10-12 due to a lack of time and not being able to understand how to use the QEMU ARM autograder to autograde questions in HW 12.

6.1 Design Analysis

Our implemented design works very well. As mentioned before, all questions for HWs 1-9 have been thoroughly worked on and tested to make sure that they have a good chunk of randomizable parameters, are autograded with correct and sufficient feedback, and have a good formatting. This design makes it easy for students to learn and engage with these homework problems. We know that our implemented design works well because we have met with our advisor/client every week to give demos on changes that we made and request feedback on if our advisor/client liked what we did or if they wanted us to go a different direction. By having this constant communication with our advisor/client, we have been able to align our final design implementation with the needs of our users and advisor/client.

The main thing that doesn't work so well with our final design is the last three homeworks, HWs 9-12, not working at all. These homeworks are in a state where students could see a question and interact with it, but the question won't be autograded nor will they be able to randomize the question. This design flaw occurred mostly due to time constraints and having to rework or finish implementing HWs 1-9, but it also had to do with not being able to understand how to use the QEMU ARM autograder for autograding the questions in HW 12. To have these homeworks in a better state, we could've dedicated more time each week to the project to be able to accomplish more and not have time be as much of an issue with our final implementation. We also could've reached out to members from the previous team to better understand the QEMU ARM autograder instead of trying to just learn about it from their documentation that just made things more confusing.

7 Ethics and Professional Responsibility

We want to make our project an effective teaching tool for students in CPRE 2880 to use as an additional resource. With that, we need to pick a correct difficulty level that won't be too easy nor too hard for students. If the homeworks are leaning more on the easy side or hard side, then students will have a tough time really learning and grasping the material that our project aims to better teach and reinforce. We aim to strike the right balance in the difficulty level of the questions and exercises, ensuring that they challenge students appropriately without overwhelming them. By carefully calibrating the complexity of our content, we hope to reinforce their understanding of course concepts while building their confidence in problem-solving.

One challenge that we face with our project is students being dishonest and cheating on homeworks. Not only will this cause students to get better grades than they deserve, but it will also cause those students who cheat to not actually learn the material within the homeworks. This undermines the purpose of our project, which is to provide a meaningful learning tool to help students better understand complex topics in CPRE 2880.

7.1 Areas of Professional Responsibility/Codes of Ethics

Area of Responsibility	**Definition**	**Relevant Item from IEEE Code of Ethics**	**Team Interaction**
Work Competency	Perform work of high quality, integrity, timeliness, and professional competence.	To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations.	Our team has worked to provide high quality work that is up to the standards of our client and the intended end users.
Financial Responsibility	Deliver products and services of realizable value and at reasonable costs.	To avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist.	Our project will give students the ability to access our services free of charge.
Communication Honesty	Report work truthfully, without deception, and understandable to stakeholders.	To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, to be honest and realistic in stating claims or estimates based on available data, and to credit properly the contributions of others.	We have a weekly report and meeting with our advisor to monitor our progress and truthfully report our work.
Health, Safety, Well-Being	Minimize risks to safety, health, and well-being of stakeholders.	To hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others,	As our project doesn't deal with physical health, safety, or well-being, the main benefit would be to help with the emotional and mental health, saftey, and well-being of our

		and to disclose promptly factors that might endanger the public or the environment.	advisor and intended users.
Property Ownership	Respect property, ideas, and information of clients and others.	To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, to be honest and realistic in stating claims or estimates based on available data, and to credit properly the contributions of others.	We utilize Git and have logs of everyone's commits and changes so they are given credit for the work they've done. As a team we also recognize the work of the groups that have come before us.
Sustainability	Protect environment and natural resources locally and globally.	To hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment.	Our project would be hosted all online, making it unnecessary for students to print out their homeworks moving forward.
Social Responsibility	Produce products and services that benefit society and communities.	To treat all persons fairly and with respect, and to not engage in discrimination based on characteristics such as race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression.	Our product will benefit our end users by providing less stress in grading and more practice opportunities for students, helping them feel better prepared for future assignments, quizzes, and exams.

Table 3: Areas of Responsibility

One ethical area that we are doing well in is how we approach the financial responsibility of our users. Our project is entirely free for CPRE 2880 students, ensuring that they have access to all of the tools and resources they need to succeed without incurring any additional financial burden. Professors also benefit from being able to host their courses for free using the PrairieLearn platform. Additionally, custom emulators and tools can be integrated into our project without requiring any additional expenses. Any future tools or resources incorporated into our project can be done at no cost as well. The only potential costs involved with our project are related to labor and server maintenance, which are minimal compared to the overall value our project provides.

One ethical area that we aren't doing so well in is in our work competence. While our project is high quality and professional in terms of formatting, functionality, and visual presentation, and also maintains the integrity of the course and its homeworks, we have fallen short in meeting our original timeline and goal. Our initial goal was to have a full beta version of the project completed and ready for students to test a couple of weeks into the semester and then spend the rest of the semester tweaking our project. Where we ended up with our work was having homeworks 1-9 completed and ready with the other three incomplete. This was due to a number of difficulties faced inside of completing homeworks 1-9. At a certain point in the semester, we had to switch over to documentation of our project and setting up the next team to work on this project so that took away from our ability to finish the homeworks.

7.2 Four Principles

	Beneficence	Nonmaleficence	Respect for Autonomy	Justice
Public health, safety, and welfare	Design helps reduce stress for learning complex 2880 topics	Our project aims to help students improve in 2880 and won't negatively impact their learning/grade	Allows students to work and get extra practice (with feedback) without needing TA or professor help (relieving burdens on graders)	Each student will get randomized versions of the question, showing no favoritism toward any student in the class
Global, cultural, and social	Helps CPRE 2880 students learn course content and achieve high homework, quiz, and exam scores	Students should have a better learning experience than with previous teaching methods	Design can be modified at any time to follow or include the different beliefs of professors and students	All students will have equal access to the platform so no student has advantage over another.
Environmental	Homeworks will be online and autogradable instead of having students use paper for homeworks, quizzes, or exams	Since our project is hosted on ISU servers, because ISU sources a majority of its energy from renewable resources, it causes less harm to the environment	Helps preserve the forests and reduce the burden of paper demand from trees	With preservation of trees and using renewable resources to run servers, we treat all aspects of nature equally better
Economic	PrairieLearn is free for 2880 students to access	Our design will not increase students' cost of tuition since PrairieLearn is free to use	Our design and documentation can serve as a template for other courses to make their own similar instances at no cost to them	Since prairielearn is free, no specific groups of people will have to spend money on it to access it

Table 4: Four principles and broader context relations

One broader context-principle pair that is important to our project is the principle of beneficence within the public health, safety, and welfare context. The goal of our design is to reduce the stress that CPRE 2880 students experience when learning difficult concepts in CPRE 2880. Stress can negatively impact a student's ability to focus and retain information, so our project aims to address this issue by creating online, autogradable homeworks that provides instant feedback. This approach gives students more learning opportunities by allowing them to practice concepts as many times as they need. To ensure this benefit for our users, our team prioritizes a user-centered design for our project. This ensures our project truly supports student success and reduces stress, contributing to their overall learning in CPRE 2880.

One broader context-principle pair where our project is lacking is in addressing the principle of respect for autonomy within the economic context. Currently, the documentation that we have covers most of the development side of making our project, but there is no documentation for how TAs and/or professors can use our project for areas that don't involve development, such as manually grading homeworks and providing further feedback. This lack of comprehensive documentation limits the autonomy of professors and TAs, as they may struggle to fully utilize the platform without additional guidance. We would push future teams to create well-rounded and detailed documentation that addresses the operational aspects of our project. This can be in the form of detailed step-by-step guides and further video tutorials that professors and TA's can follow along with.

7.3 Virtues

The first virtue that is important to the team is determination/tenacity because without it, it would make it hard to make much progress on our project. When we are continuing on a project that a previous team has worked on, we were thrown right into the deep end of the project. The whole team had to spend a lot of time learning how to use the PrairieLearn framework, as well as learning what the previous team all accomplished. Because the team is determined to make a project to help CPRE 2880 students, professors, and TAs, we stuck with the uncomfortableness and persevered through the challenges. This determination allowed us to push past the initial hurdles, such as understanding the complexities of the framework and deciphering the previous team's work. Instead of being discouraged by the steep learning curve, we used it as an opportunity to grow and collaborate effectively.

The second virtue that is important to the team is having compassion for ourselves and for our users. By being compassionate and showing kindness and patience to one another, our team will work better and problem solve more effectively. Being compassionate also allows us to create a productive working environment for the entire team, where every team member feels valued and supported. Compassion also makes us think in the perspective of CPRE 2880 students to better understand their needs, allowing us to develop and modify questions that make the most sense for them and will actually be helpful.

The third virtue that is important to the team is creativity. Because of the nature of our project, we have to come up with questions ourselves that are engaging and interesting for the students. We need to have creativity in order to come up with unique questions that really test a student's knowledge and understanding. We have already used creativity to fix questions developed by the previous team, where these fixes improve the questions and make more sense to the students.

The fourth virtue that is important to the team is cooperation because it is the foundation of successful teamwork. Cooperation ensures that all team members can work together most efficiently, leveraging each other's strengths and supporting each other's weaknesses. Throughout our project, we have been using cooperation to share knowledge of different areas within PrairieLearn. Whenever one team member encounters a problem, we rely on open communication to come up with a solution in a timely manner. To make the progress that we did in finishing the first 9 homeworks,, we divided and worked on separate homeworks based on our individual strengths and areas of expertise. This approach allowed us to make significant progress while ensuring that each homework assignment was developed with care and attention to detail. By relying on cooperation, we have been able to overcome challenges more efficiently and ensure that our project meets the needs of CPRE 2880 students, professors, and TAs.

Caden Otis

One virtue that I have demonstrated the best so far in my senior design work is determination/tenacity. Determination is important to me because without determination, it would be difficult for me to overcome difficulties. I have demonstrated determination by always thinking about the positive effects our completed project will have on our users, which keeps me focused and motivated when encountering any obstacles along the way. When I first started working with PrairieLearn, I knew nothing about it. I had to rely heavily on my determination to figure out how it worked. After reading through lots of documentation, going through already developed questions, experimenting with my own questions, and troubleshooting, I finally understood the inner workings of PrairieLearn. My determination helped me push through the frustration and uncertainty of using a completely new platform, ultimately allowing me to create meaningful and effective questions for CPRE 2880 students

One virtue that is important to me that I haven't demonstrated well is creativity. Creativity is important to me because it allows teams to develop innovative solutions and approach problems from fresh perspectives. It enables us to think outside the box and come up with unique ideas that can add value to our project and make it stand out from similar platforms. Inside of the aspects of the project that I worked on, I feel like I was able to design and develop engaging and innovative questions that enhance the learning experience for CPRE 2880 students, even if it wasn't necessarily displaying creativity. Looking back, I would've liked to display my creativity more in my aim to bring creative solutions to any technical or design challenges that we face, ensuring that our project is not only functional but also impactful and user-friendly for our users.

Joey Krejchi

A virtue that I have demonstrated is commitment to quality. This is a virtue from Pritchard's list in lecture, and it is something I've strived to practice throughout my life. It is important for us to prioritize quality in our work so that we create things that exceed expectations and stand the test of time. When you focus on making a quality product, that extra time you put towards quality pays back when you create something great and lasting. I have demonstrated this virtue as Quality Assurance manager for our team. I make sure that our application meets the expectations of our client and that we are making something we can be proud of.

One virtue that I could've put more focus on is that of optimism. It seemed like there were big expectations for our project, and there were tough problems I wasn't sure if we could solve. I think optimism is important because without it, you will struggle to find enjoyment in what you are doing. Looking back, I think I could've found optimism in our project by appreciating what we have done and accepting that we did the best we could. I think some more optimism on my end would have a positive impact on the work I do in future projects.

Devin Alamsya

One virtue that I demonstrated is creativity. Creativity is important to me because working with a creative mindset allows me to think outside the box and solve problems in new ways. We can deliver project milestones that are innovative and engaging, not the same things as before. A way that I have demonstrated this is transforming short answer questions into questions with a new format, completely randomizable, and completely autogradable. This was able to happen because I viewed the problem with a creative mindset.

One virtue that I should've focused more on is having clear and thorough documentation. Having clear and thorough documentation is important to me because it helps me keep track of all I've done, but it also helps the team, future senior design teams, and users to know what changes were made, how to make them, the thought process behind them, and the implications of any changes. Although I've made progress on the project, documentation (documents, videos, slideshows) should be made for the progress accomplished. To demonstrate this virtue more towards the end of the project, I created documentation for changes to help external parties understand the changes made.

Justin Cano

One virtue I demonstrated during my work is perseverance. Perseverance is important to me because it shows my dedication to overcoming obstacles, even when the obstacle is complex or time consuming. I've demonstrated perseverance while addressing the issue of setting up OpenID Connect (OIDC) to allow ISU students to use Single Sign On (SSO) with PrairieLearn. Since PrairieLearn does not by default allow you to set up OIDC SSO, you can modify the source code of it to allow it. However, there is almost next to zero documentation out there explaining how to do it, coupled with the fact the ISU just recently swapped from using OKTA to Microsoft for SSO made last year's teams documentation less than useful. However despite this, I worked through this challenge by dedicating myself to this obstacle until I overcame it.

An important virtue that I struggled to demonstrate is communication. Communicating is essential for projects, as it helps ensure that others know what tasks you're working on and helps keep the team on schedule. I found that I often struggled to let the team know what I was working on as my part of the project was very much so different from their aspects of the project. To help improve my communication skills moving forward in every other project, I will work to share my progress updates more often with my future teams, even if it's a small achievement or issue.

Rachel Druce-Hoffman

I feel that I have embodied the virtue of perseverance this semester. This virtue is an important skill for everyone to have in order to overcome adversity in life, or even to learn and move on from failures. As is to be expected with a capstone project, we have faced challenges and technology that we have not worked with before. It has been intimidating, especially when we are not sure if our vision for a homework question is possible to implement. Personally, I have not worked extensively with Python before, and so I have had to practice resilience with learning this language. It is frustrating knowing I could easily solve a problem in a different coding language, but having to struggle to translate my thoughts into new syntax. This hardship has taught me a lot, and my ability to put the time in on these problems has paid off multiple times.

A virtue that I had not really demonstrated until the end of the semester was maintaining good documentation habits. There is very little point to learning and engineering solutions unless you document your work to teach others what you learned. This semester, I have been more caught up in trying to meet our team's beta test goal to do much documenting as I go. I regret this, as I want to help future developers save time and understand how to use PrairieLearn better. I have noticed that I tend to get too focused on the problem at hand to document the steps I take in the moment, and I will work on this. Towards the end of the semester I got much better at documenting my work and making clear documentation of beta test scripts and canvas integration.

8 Conclusions

8.1 Summary of Progress

This project was started under the vision of improving the course CPRE 2880 by transitioning homework assignments to the PrairieLearn platform. Our primary goal was to help minimize the grading time for instructors without harming the students' success in the course. We did this by using the modularity of PrairieLearn to transform the previous homework assignments into randomized and fully autogradable questions. So far, the first 9 homework assignments are fully functional and are ready for beta testing in the Fall of 2025, while the development of the remaining assignments are still ongoing. We have also spent a good chunk of time developing clear and concise documentation for future teams so that they can pick up the project really well and hit the ground running instead of taking a ton of time to figure things out.

As far as our testing, we've tested the first 9 homeworks in house thus far (tested by our own team) and have conducted a test with the professor to get their feedback. We were hoping to get a beta test of some sort done this semester, but that proved to be difficult due to our progress and the lack of response in our inquiry to get a beta test done.

We fell short in our goals of having all the homeworks completed and tested by students currently in CPRE 2880. This was mainly due to a complexity in the problems that we didn't realize was there which led to increased work time on them. As far as not getting the beta test out, we just weren't met with a response from the professor until it was too late to get a beta test pushed out to students. Although our plans for a beta test didn't pan out, during the Spring 2025 semester, we had Dr. Zambreno, Dr. Rover, and her TAs use and experiment with the final state of our project as an alpha test. Some of the feedback that we got related to the usefulness of the autograder and making it quick and easy for students to get grades without the need for TAs or professors to step in. We also got good feedback on the overall layout of homeworks and how questions were formatted, especially with C and assembly programming questions, having a simple and intuitive design that students can easily grasp. However, we also got feedback on factors that could be improved, such as allowing for incomplete submissions and making conceptual-type questions more unique than a typical Canvas format.

8.2 Value Provided

We believe that we have created a product that meets the needs of our users. Through the first 9 homeworks our product gives students the ability to practice questions that can help them study and understand the material. These questions also give them instant feedback that allows them to learn from mistakes or successes in the moment.

For the professor's and TA's, it meets their needs by being a product that has full randomizable and autogradable capabilities. That means that question creation and grading responsibilities are no longer something that they need to spend time with. Freeing them up to help students the best they can so they can give lots of in depth feedback to students that need the help instead of being stuck grading a ton of assignments.

From the feedback of our alpha test, it was made clear that the autograder would save a lot of time from professors and TAs having to manually grade student submissions. It also provided a quick way for students to get small feedback. The feedback that we received also made it clear that harder topics in CPRE 2880 had very unique and engaging questions to help students' overall learning with the material. The results of our alpha test showed that we were able to meet our users' needs of having an application that could improve overall student learning while also allowing professors and TAs to focus less on grading and spend more time helping students.

8.3 Next Steps

The next steps that future teams working on this project can take are finishing the last three remaining homeworks, doing a full beta test, and finishing out all of the thorough documentation of the project. The first 9 homeworks shouldn't need to be touched but the last three will need to be completed, tested, and revised. With that comes the full beta test that they can give to students to test the full functionality of the projects found in all homeworks, and then they would need to document all the new technology and processes found in the final three homeworks they complete. It would also be good for them to document results of a beta test. Some other new projects could be creating a homework platform like the one we've created for CPRE 2880 for other classes. Using our documentation, I believe that this is something that another project could potentially accomplish quicker than if they tried it without using our documentation. It would set them up well to set everything up and make great strides of progress in a potential new project.

9 References

[1] LinkedIn Learning: Online Training Courses & Skill Building, https://www.linkedin.com/learning/ (accessed Dec. 6, 2024).

[2] Quizlet, https://quizlet.com/ (accessed Dec. 6, 2024).

[3] zyBooks, https://www.zybooks.com/ (accessed Dec. 6, 2024).

[4] Coursera, https://www.coursera.org/ (accessed Dec. 6, 2024).

[5] "PrairieLearn," ReadTheDocs.io. https://prairielearn.readthedocs.io/en/latest/ (accessed Dec. 05, 2024).

10 Appendices

Appendix 1 – Operation Manual

- https://git.ece.iastate.edu/class/ece-prairielearn-documentation/-/tree/main?ref_type=heads
    - Documentation Repository
- https://docs.google.com/document/d/1_U9zlj2IjJdGGZJ0iNjF5MABmVICrgQY/edit#heading=h.e6q0nau3hzgf
    - Set-up documentation

Appendix 2 – Alternative/initial version of design

- https://git.ece.iastate.edu/sd/sdmay24-33
    - This is the repository for the previous team in this legacy project. Our project has built off of this previous project.

Appendix 3 – Code

- https://git.ece.iastate.edu/sd/sdmay25-33

Appendix 4 – Team Contract

Team Members

1. Caden Otis
2. Rachel Druce-Hoffman
3. Justin Cano
4. Devin Alamsya
5. Joey Krejchi

Required Skill Sets for Your Project

- Embedded Systems Programming
- Knowledge of various programming languages.
    - C
    - Python
    - JavaScript
    - HTML
- Security techniques to ensure our production server is secured.

Skill Sets covered by the Team

- Embedded Systems Programming
    - Every member
- Programming languages
    - Every member
- Security
    - Rachel Druce-Hoffman
    - Justin Cano
    - Joey Krejchi

Project Management Style Adopted by the team

- Hybrid of Waterfall and Agile

Individual Project Management Roles

Project Manager (Caden Otis): Coordinate with the team to develop clear project goals and metrics. Facilitate the progress made on tasks every week to make sure the team is on track with deadlines.

Technical Lead (Justin Cano): Sets up the production server for our project and ensures it is hardened and secured. Work with ISU IT, ETG and ASW for various aspects of the project.

Notetaker (Rachel D-H): Takes notes during advisor meetings

Quality Assurance (Joey Krejchi): Ensures that the product is meeting the expectations of our client-advisor Dr. Jones. Attentive to details that are crucial to a successful project.

Consultant (Devin Alamsya): Work to meet project goals and provide high quality work, be a sounding board for ideas, and be there to assist in any problems that arise within the project.

Team Contract

Team Members:

1) Caden Otis

2) Rachel Druce-Hoffman

3) Justin Cano

4) Joey Krejchi

5) Devin Alamsya

Team Procedures

1) Day, time, and location (face-to-face or virtual) for regular team meetings:
   a) We will have in person team meetings on Wednesday at 1pm and virtual team meetings as needed on Friday at 12pm via discord.
2) Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
   a) Discord and SMS messaging
3) Decision-making policy (e.g., consensus, majority vote):
   a) Majority vote
4) Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
   a) Rachel Druce-Hoffman is our official notetaker and her notes are stored in a google drive.
5)

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
   a. All members will attend meetings on time, unless there is an acceptable reason for their absence. Members will notify the team ahead of time if they are unable to attend. Finally, each member will also be attentive and participate in each meeting.
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
   a. Each team member will contribute on team assignments to ensure we complete our work before its deadline.
3. Expected level of communication with other team members:
   a. Members should view the team Discord at least once a day so they're aware of progress being made by fellow team members and what tasks are being worked on.
4. Expected level of commitment to team decisions and tasks:
   a. Each member should also contribute about an equal amount of work per task to ensure no one member is doing significantly more work.
5.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

   a. Each team member will share leadership responsibilities, as we don't want one specific person to be the team leader.
2. Strategies for supporting and guiding the work of all team members:
   a. We will maintain a judgement free working environment and always be open to helping one another.
   b. We will also have a standup during our weekly meetings to check the progress of each team member where they can discuss any issues or breakthroughs they have found.
3. Strategies for recognizing the contributions of all team members:
   a. During our weekly team meetings, we will have a standup so each member can discuss what they did the previous week so we can recognize what everyone has done.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
   a. Our team's majors are as follows:
      i. Caden Otis: Electrical Engineering
      ii. Rachel Druce-Hoffman: Cyber Security Engineering
      iii. Justin Cano: Cyber Security Engineering
      iv. Joey Krejchi: Cyber Security Engineering
      v. Devin Alamsya: Software Engineering
   b. Each team member has taken CprE 2880 so we all have knowledge of the course content we'll be using.
   c. Each team member also has a variety of coding knowledge, including C and Python, which makes up the majority of our project.
   d. Rachel, Justin, and Joey all have knowledge of security practices from their Cyber Security Engineering coursework.
2. Strategies for encouraging and support contributions and ideas from all team members:
   a. We will make sure that each member's thoughts and ideas have been heard in a discussion so we can promote equal contribution.
3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)
   a. We will be upfront with each other and foster a constructive environment. This way members can discuss potential issues and resolve them before they occur.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
   a. Meet all class and advisor deadlines on time.
   b. Go above and beyond for our weekly goals.
   c. Have a cohesive final project that we are happy and proud of.
   d. Have a product that can be beta tested by the beginning of next year.
2. Strategies for planning and assigning individual and team work:

    a. We will focus on being fair with each member's workload while also playing to each member's individual strengths.
3. Strategies for keeping on task:
    a. Incorporate breaks into our work time and set aside any distractions.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
    a. We will have a three-strikes-and-you're-out policy for handling infractions. For the first two infractions, the team will discuss the infraction and determine the best case of action. If the problem persists after the first two breaches, we will then bring it up to Dr. Shannon and Dr. Fila.
2. What will your team do if the infractions continue?
    a. We'll contact the course instructors and advisor to get advice on how to proceed with the infracting student.

****

a. I participated in formulating the standards, roles, and procedures as stated in this contract.
b. I understand that I am obligated to abide by these terms and conditions.
c. I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

NAME Joey Krejchi DATE 04/30/25

NAME Caden Otis DATE 04/30/25

NAME Justin Cano DATE 04/30/25

NAME Rachel Druce-Hoffman DATE 04/30/25

NAME Devin Alamsya DATE 04/30/25