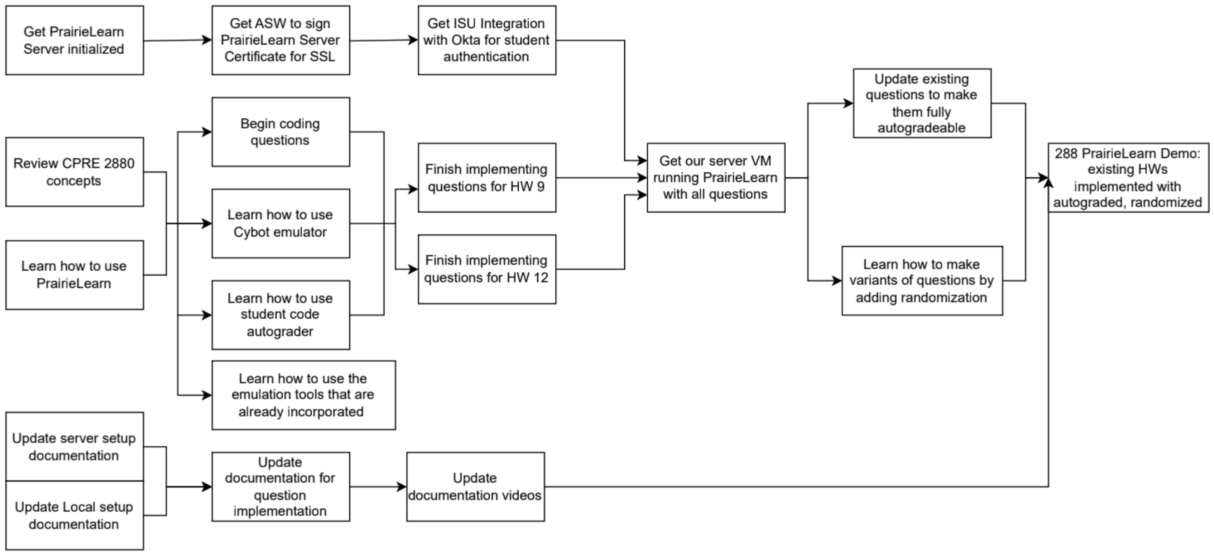
3 Project Plan

3.1 Project Management/Tracking Procedures

For our Senior Design project, we adopted a hybrid style of agile and waterfall to help achieve our project's goals. Since the primary foundation for this project has already been laid by the previous year's team, our style will be partly agile focused as we're doing iterative development, where each team member is working on different parts of the project. We're also receiving regular feedback from our advisor each week as we implement and test new features. Since we have a structured foundation for the requirements of our projects requirements and design, we're also making use of the waterfall style. We have a clear image of what the final project should be and have hard deadlines for certain features to be implemented. This methodology will help us adapt to any unexpected issues that arise as the project progresses, 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 this semester and the next, we're using a suite of tools for project management. This includes 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 can also use it for issue tracking, which will help us organize and manage bugs, tasks, and new features. For team communication, we're making use of Discord to coordinate tasks, share updates, resolve issues, and ask for feedback and help. Discord's channel-based organization will allow us to have dedicated channels for different project components to ensure we remain organized.

3.2 Task Decomposition



Get PrairieLearn Server initialized
Get ASW to sign PrairieLearn Server Certificate for SSL
Get ISU Integration with Okta for student authentication
Review CPRE 2880 concepts
Learn how to use PrairieLearn
Begin coding questions
Learn how to use Cybot emulator
Learn how to use student code autograder
Learn how to use the emulation tools that are already incorporated
Finish implementing questions for HW 9
Finish implementing questions for HW 12
Get our server VM running PrairieLearn with all questions
Update existing questions to make them fully autogradeable
Learn how to make variants of questions by adding randomization
288 PrairieLearn Demo: existing HWs implemented with autograded, randomized
Update server setup documentation
Update Local setup documentation
Update documentation for question implementation
Update documentation videos

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.

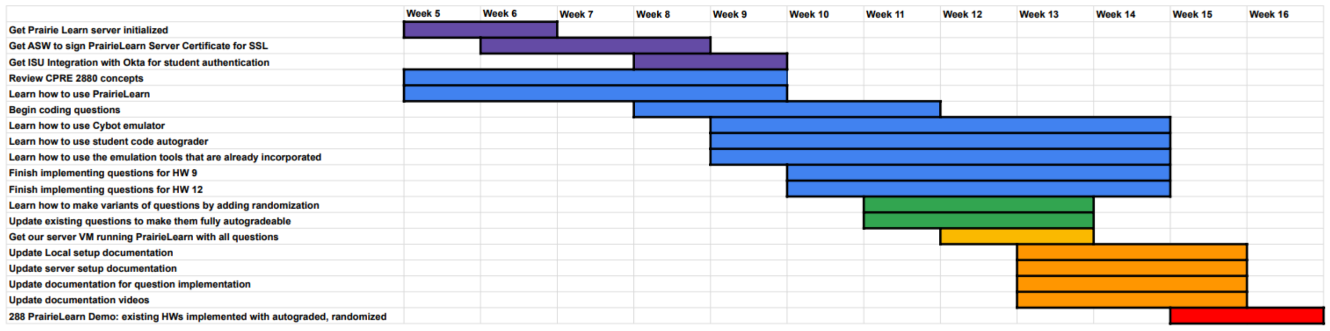
3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

By the end of the Fall 2024 semester, we would like to reach several key metrics. The first metric that we would like to reach is to have our PrairieLearn application be completed in a beta version, where all homework problems have been implemented into our application, and every question is randomized and autograded. With these metrics planned, we want to have a working version of our application ready for use by the CPRE 2880 students in the Spring 2025 semester. That way, we can gain helpful feedback that can better our application.

We also have developed several key metrics that we would like to accomplish by the end of the Spring 2025 semester. We first want to finish updating documentation from the previous team and complete any new documentation that we create. 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

Gantt Chart:



In our 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 fully autogradeable 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, fully autogradeable, and completely randomizable.

With us using a hybrid style development model with both Agile and Waterfall, our gantt chart most accurately depicts our plan. 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

Get ASW to sign PrairieLearn Server Certificate for SSL

Only professors at ISU can submit requests to 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

Get ISU Integration with Okta for student authentication

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

Review CPRE 2880 concepts

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

Learn how to use PrairieLearn

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

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

<u>Learn how to use Cybot emulator</u>

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

<u>Learn how to use student code autograder</u>

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

<u>Learn how to use the emulation tools that are already incorporated</u>

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

<u>Finish implementing questions for HW 9</u>

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

<u>Finish implementing questions for HW 12</u>

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

<u>Learn how to make variants of questions by adding randomization</u>

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

<u>Update existing questions to make them fully autogradeable</u>

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

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

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

288 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

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	10

	sure our demo is ready.	

3.7 Other Resource Requirements

Many of our remaining resources are more intangible. For help with development, we will turn to the PrairieLearn git forum or the PrairieLearn Slack server. Being able to read posts from other developers or even pose our own questions will help us when we get stuck. Furthermore, we will consult the official PrairieLearn documentation, as well as the previous CPRE 2880 PrairieLearn's documentation for guides that are more tailored to our needs.

Beyond our advisor Dr. Jones' feedback, we may consult the other CPRE 2880 professor, Dr. Rover, in case she has additional perspectives or features in mind. Both of these professors are instrumental in providing us with resources from their classes, including homeworks and lecture materials, as well as guiding our implementation through the expectations they hold for students. Finally, after we deploy our beta version to students in the spring semester, we will utilize their feedback to polish and guarantee PrairieLearn's effectiveness. The 2880 students will be a crucial resource for us to understand the student experience and making our solution effective for them.