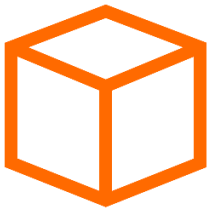

CS 463: Senior Software Engineering Project
Project Retrospective

Prototype a web-based tool for creating and executing task-delineated, collaborative, AI-assisted assignments

Group 28

Team Roles

Name	ONID	Role
Oliver Zhou	zhouo	Project Manager
Trent Matsumura	matsumut	Developer - Backend
Ethan Lu	luet	Developer - AI Integration
Collin Kimball	kimbacol	Developer - Web UI
Sai Meenakshisundaram	meenkass	Documentation

Project partner: Sanjai Terrazas Tripathi

Project partner email: sanjai.tripathi@oregonstate.edu

Contents

Introduction

In this project retrospective, all aspects of the development process throughout the entirety of the Senior Capstone Engineering Project was conducted. The retrospective was conducted internally on a call session with the entire team, where we all reflected on the past three terms and answered all of the required prompts for the retrospective assignment. We wrote down our answers in bullet points to each section, and then sections were divided between the team and we began expanding on each section in writing on this document. We would help each other writing and reflecting on each section of this project retrospective. The rest of the documentation is a comprehensive summary of our reflections and answers to our design process, development process, communication, documentation, and much more.

1. Project Overview

Introduction

Our project, Task-Delineated AI-Assisted Assignments, or TD3A for short, is a web-based application that functions as an enhanced Learning Management System, similar to websites such as Canvas, which is a standard tool in educational settings. The purpose of this project is to solve the issue regarding integrating LLMs into the learning environment. Students are encouraged to use AI chatbots as a crutch or shortcut tool, so this project aims to use the technology more productively as well as giving teachers the tools to design their assignments around this paradigm.

Goals

Our key goals in this project are to serve a Learning Management Platform which serves a productive experience while addressing both problems of inappropriate AI use and ineffective tools to break down tasks.

Scope

The project functions as a web-based Learning Management System (LMS) with standard industry features that additionally integrates Large Language Model (LLM) technology to improve both the student learning process and the educator’s ability to design and manage

assignments. Our project focuses on a single organization structure and starts off deployed in a single OSU classroom setting under supervision of our project mentor.

Deliverables

- An all-in-one Learning Management Platform with a friendly interface for both students and teachers.
- AI Metacognitive bot that enhances student learning via LLM models.
- Assignment objects that can be broken into sub-tasks to fulfill task-delineation.
- Student logging tools and AI summaries for teacher use for analyzing their class performance.

---

2. Team Members and Mentor

Name	Role
*Sanjai Tripathi**	**Project Mentor**
Oliver Zhou	Project Manager
Trent Matsumura	Developer - Backend
Ethan Lu	Developer - AI Integration
Collin Kimball	Developer - Web UI
Sai Meenakshisundaram	Documentation

---

3. Timeline Review

Our team created timelines for each term. These timelines layed out the order of our development process and were occasionally adjusted through development. The fall term mostly followed the structure of writing groundwork documentation according to the class

requirements. We started off with basic team documents, and moved onto writing about requirements and design. For that reason, most of our timeline discussion revolves around the development and presentation terms.

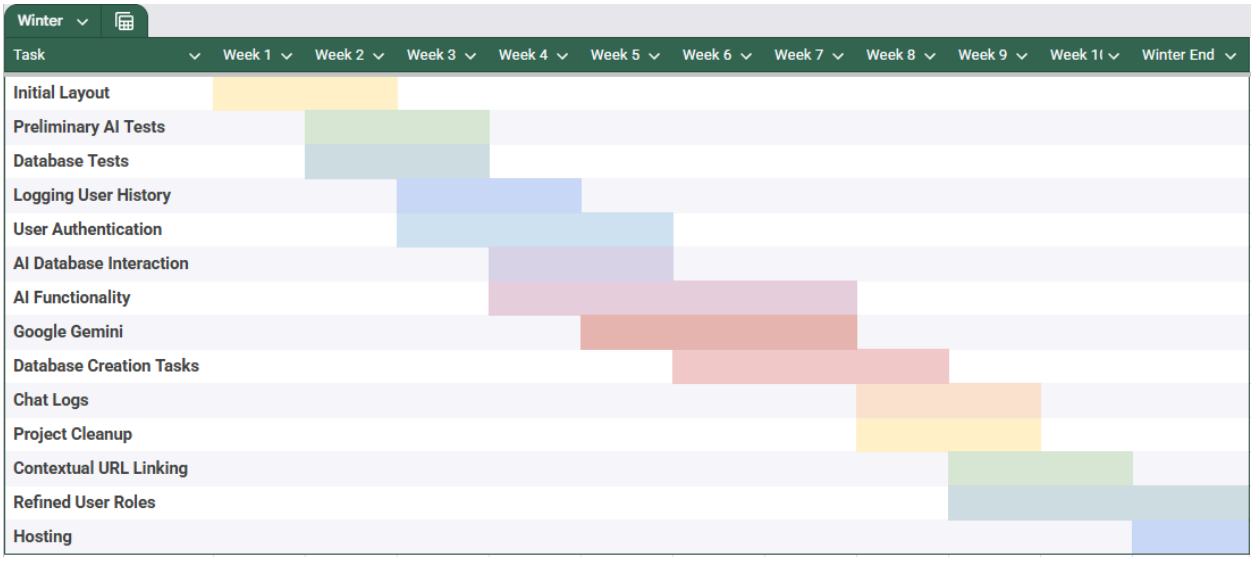

Figure 3.1 – Winter Term Development Concurrency Timeline

Some key milestones in our winter timeline was the AI functionality requirement being completed in week 7, or around our 4th sprint. The AI functionality of our project was difficult to implement and required a lot of preliminary testing starting at the beginning of development with our Ethan going off to test AI models independently.

Our mentor never gave us hard deadlines on many of the features, but he did recommend an order of features that he wanted to get completed. Earlier in winter, we were expected to complete task-delineation, and we worked hard to lay the fundamental groundwork for addressing that aspect of the application in time so that more time could be spent on AI features, and instructor tools.

In the winter term, some delays inevitably occurred because we were still getting used to the development cycle. Because of the need to create the fundamentals of the website, some features could not be worked on until others were completed. For example, instructor analysis or student chat log summaries would require our AI to work, but AI implementation took a long time before it got off the ground.

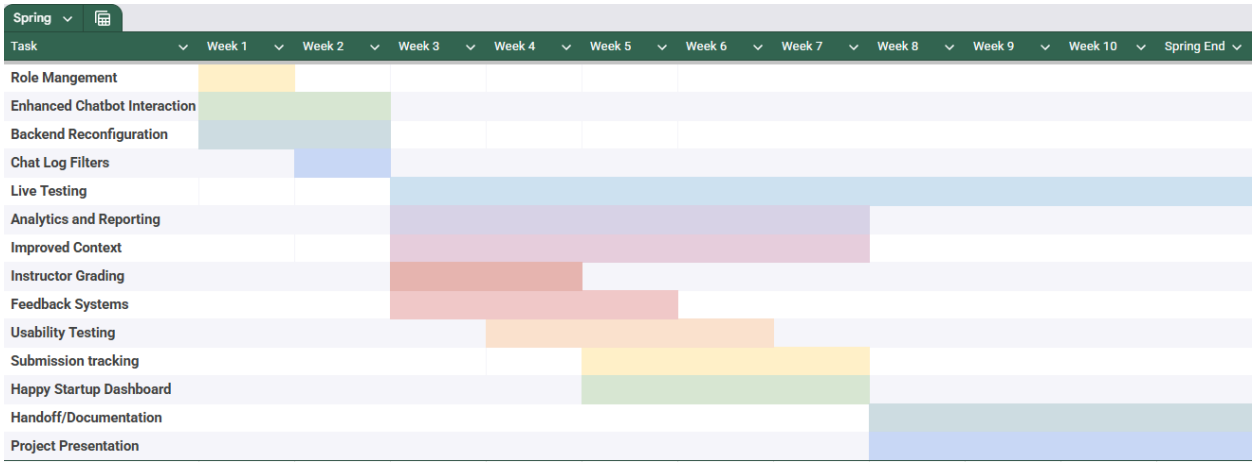

Figure 3.1 – Spring Term Development Concurrency Timeline

The spring term continued on with the development of the project, though many of the keystones and requirements were focused on different things in this term. The major key milestone we accomplished this term was live deployment in a real world testing environment with students from the business department. Our project mentor gave them extra credit for interacting with our project and giving feedback to us. This was a huge accomplishment for our team and we were excited to see our project working in action.

Throughout the last three sprints in the spring term, we finished off extra miscellaneous features to enhance each feature according to our project mentors wishes, such as improved context and analytics. These had no hard deadlines, but we simply aimed to complete as many of them as was reasonable before development ended. Development ended shortly after the 3rd sprint demo, around week 7 in the spring term.

Any delays we faced in spring were minimal, as development had few blockers. Though, there were waits between receiving feedback or suggestions on how to proceed form our project mentor, but things were discussed in communications.

---

4. Requirements, Design, Implementation, and Deployment

Requirement Gathering

Our requirement gathering and documentation process was thorough and iterative, with only minor changes needed after initial reviews. We documented most of our requirements and decision making on our requirements document in our team drive, which kept things organized effectively. Regular reviews with our team with help from email exchanges with our project mentor helped us align expectations and refine the timeline for better focus and concurrency.

Ongoing discussions with our mentor helped clarify misunderstandings or redirect features whenever we needed. As a team, we ensured prioritized achievable goals were within our project scope, and our documentation process of requirements benefited greatly because of it. When we completed revisions of our requirements documentation, we added on additional features that were recommended by our project mentor.

Design Process

Our design process included many iterations of prototypes designed on draw.io which helped map out the structure and mockups of the user interface for our project. We covered all aspects of design, including the backend technologies we chose to implement AI features and the frontend user interface in these models.

The way we decided on design principles or decisions was as a group in our meetings and text discussions about implementation. We also gained valuable suggestions and design requirements from our project mentor which guided our decision making.

This section includes a few samples of our design mockups which drove the design process of our application.

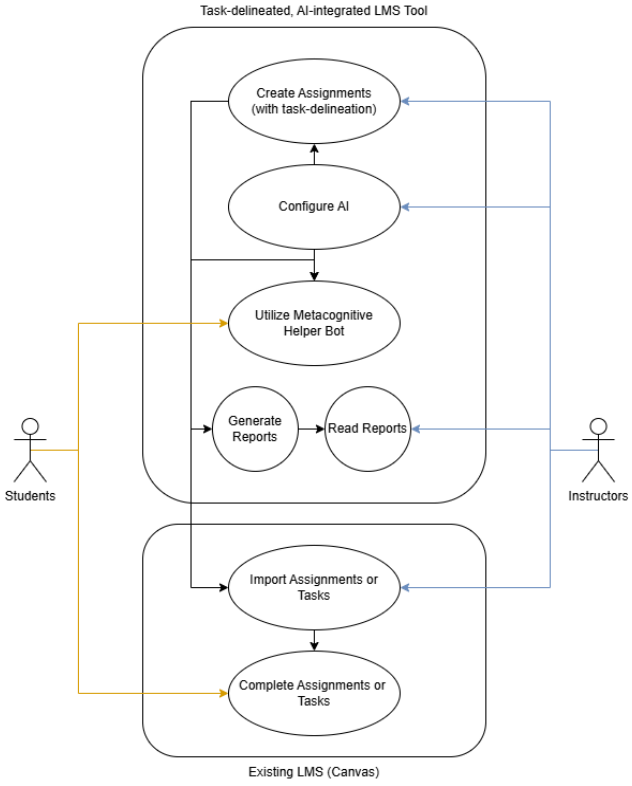

Figure 4.1 - User Design Diagram

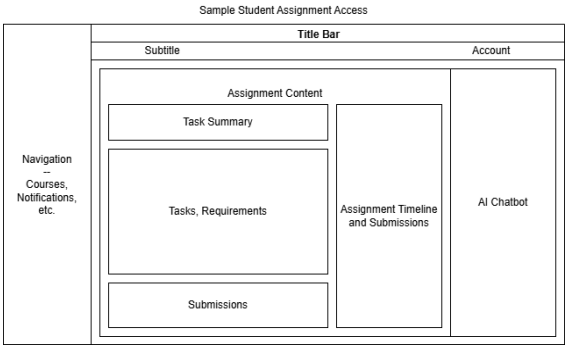

Figure 4.2 - Sample Student Assignment UI Mockup

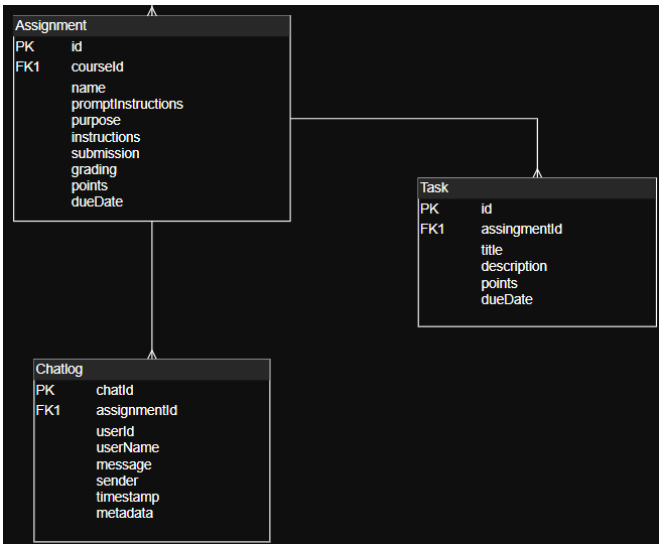

Figure 4.3 – Sample Portion of Database Schema

Changes and Adaptations

Much of the project remained the same, as we followed many of the features with strong principles and developed them mostly to requirements. However, it is necessary for us to find other methods when certain implementations did not work out during development.

One thing we had to change was our AI model. We originally wanted to use a smaller third-party open source AI model which would be more secure as a product. However, our AI-expert found that it wasn't powerful enough to implement into our project. As a result, we discussed which AI models we could use instead, and came to a joint decision led by our AI expert to use Google's Gemini AI model for our project. This model was able to integrate its API better with our project and was easier to implement.

Another significant change that we were debating was whether we made this tool an extension of existing LMS such as Canvas. We acknowledged that we would be unable to directly compete with existing Learning Management Systems such as Canvas due to the scope of the project, and for that reason, we needed a solution to make our project provide value while stating within a scope that is realistic, and still provides value to the classroom. We decided that we would simply make our assignment objects similarly structured to Canvas, as well as its user interface. This would mean that assignments are easily transferable between the two platforms, and that students could be linked to our project through existing LMS to try out our additional AI or delineation features. This adapted to the existing standards used in the classroom setting, and found a place for our project to exist. This is further supported by our testing with users in our project mentor's class.

Implementation

The implementation phase of our project worked efficiently. Each member would respectively code and work on their assigned role for the project. Then, after everyone was completed, we would join a team meeting where we would collaborate and combine work together. Separately, independent tests by each team member were conducted to make sure there were no issues. Because a lot of development went smoothly, there usually weren't any blockers or issues. Future iterations as we progressed through development were facilitated by the project mentor, who gave feedback and features and inspired our improvements, which were implemented by each development expert accordingly.

Deployment

Our deployment process used Vercel and Render to host the frontend and backend, making it easy to manage both parts of the application. We set up separate production and development environments to keep things stable and allow for thorough testing before changes go live. The process itself was straightforward. Once you link the GitHub repo, deployment happens automatically whenever relevant code is pushed to the main branch.

One challenge we faced was Render’s free tier shutting down the backend server after 15 minutes of inactivity. To get around this, we set up UptimeRobot to regularly ping the backend and keep it running continuously. Another limitation is that only the repo owner can deploy directly, so anyone else who wants to deploy needs to fork the repo first. While these free-tier restrictions required some extra steps, they didn’t prevent us from maintaining a mostly automated deployment workflow.

Overall, by combining these tools and workarounds, we were able to create a deployment system that’s efficient, easy to update, and stable enough to support ongoing development and testing.

---

5. Verification and Validation

Outcome Evaluation

We believe that our objectives in this project were met. We successfully implemented features that address both of the proposed problem statements for our project. We were able to deploy a platform that real users have tested which contains an AI-chatbot that students could interact with to complete assignments, alongside assignments which have multiple parts to them to address task-delineation/assignment organization.

Additionally, our project has a section for instructor accounts for them to review student logs for analysis of student performance, which is additional evidence of addressing and completing deliverables for instructor tools.

Extra features which went beyond the scope of the original requirements to further improve usability for students were worked on, and a dashboard was being worked on to further improve instructor tools. These extra features also demonstrate that we had positive outcomes for the project and were able to develop further than was required originally. This also enabled us to get real user feedback from real students.

User Feedback

User feedback from our project mentor has been incredibly valuable in shaping TD3A. Our mentor noted how focused and engaged students were during live sessions, which confirmed the tool's ability to support meaningful, independent work. The dashboard was well received, especially its potential to highlight struggling students through clear metrics.

We also got thoughtful suggestions, like adding file uploads for more assignment context and giving instructors better control over the AI. Chat logs helped surface issues like the AI jumping ahead too quickly, which led us to improve context handling for the future.

In the future, our project mentor has discussed with the team that more students may be trying out the application in extra credit assignments, so additional testing and user feedback will likely come and will give future teams which work on the project a better idea on how to move the project forward.

Figure 5.1 – User Analytics for users who have tested TD3A

Testing Strategies

Our project had a variety of insights on how we tested our product beyond deploying the project offline setting. Especially in the spring term, we were able to do comprehensive testing by running the tool in a live classroom, and it gave us authentic feedback and helped surface specific edge cases, like how we track time across multiple sessions.

We captured student interactions in usage logs too, which we used to guide improvements and build sample reports. On the instructor side of our project, we tested out dashboards and chat review tools. Feedback from our project mentor led to meaningful UI tweaks and even introduced new ideas like letting instructors upload extra context files for assignments. During our technical testing, our developers would run repeated modular tests on each service in the application which allowed us to track bugs and small inconsistencies along the way to help us develop as the term progressed.

---

6. Team and Communication

Team Dynamics

The team worked together easily and everyone was friendly with each other and was able to get along. We all became friends in the process. Our communication methods were simple: **Internal communications were conducted through a discord server,** where updates to our TA and internal meetings were conducted. Team members would ping each other for any important updates or anything we needed to be accountable or present for.

External communications with our TA were conducted on a semi-regular basis in Zoom recording sessions, as well as our team meetings with the project mentor.

Collaboration tools were also simple: We used GitHub for organizing all project tasks and information, and used Discord and Google Team Drive for the rest. We would pin important information or features in our Discord channel, and use the team drive to organize and facilitate our development sprint process.

Roles and Responsibilities

All of our role assignments were as clear and defined as possible. Everyone had a unique role that contributed significantly to the project individually in our project, and the project would not be able to proceed without any of the team members. We have always emphasized our unique team roles on the front page of all our documentation, and we believe that it helped refine our team members significantly.

Our project had 3 main components, and it was easy to nominate one member as the primary expert on one of them. Between the 2 remaining members on our team, they split all non-technical aspects of the project, with one of them taking on managerial responsibilities, and the other taking on writing and documentation tasks.

Please visit section 2 in this document for reference of our specific team roles.

Conflict Resolution

We do not believe that our team ever ran into any significant conflicts. This is possibly due to the fact that all of our roles were modular and each person focused on one significant portion of the project. This gave a lot of autonomy to the team for specific decisions.

However, there were still some disagreements in the design process, such as what kinds of technologies we used to solve our problems, and the AI models we used in the project. These were resolved quickly and with civility in Discord, where we conducted most of our communication. People remained level-headed in our discussions, and we enforced our expectation with a thorough team charter document we created for the class.

Many features were driven by the needs of our project mentor, so a lot of the time, the team would be following along his guide. Any suggestions we might make were usually accepted by our project mentor, because he gave us a lot of freedom to decide on the implementation, as he was not a technical person.

Internal Communication

Our internal communication was responsive, consistent, and effective. This is because **the entire team conducted our communications on Discord,** where we all became friends and were able to get a hold of each other every day. Because of the simplicity of Discord, and the fact that everyone in our project team was already familiar with the platform, meetings were easily conducted with a click of the call button, and features and other communications were done instantly through text messages.

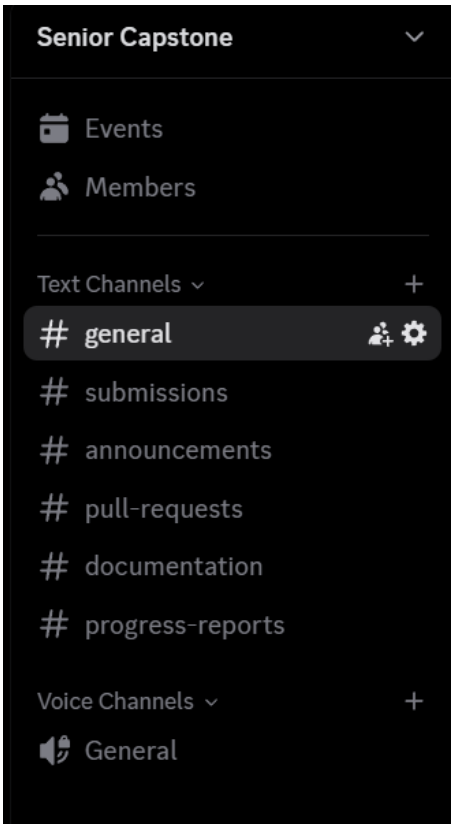

Figure 6.1 – Example of our Internal Discord Server Structure for Organizing Communication

External Communication

We did our best to communicate with the project mentor effectively throughout the entirety of the project, with varying degrees of success at different points. **The primary mode of communication with our project mentor was through email.**

We had Oliver conduct most of the project related external communications with our project mentor, whenever team meetings and class related requirements were needed from the project mentor. On the other hand, development focused sprint meetings were conducted regularly which were organized and then solo-conducted by Collin. The two modes of communication from both of our team members meant that we could communicate all aspects of the project with our mentor regularly.

Despite our communication system. There are some challenges we faced when communicating. Mainly, some information did not always get communicated, so then there would be delays before everyone could be on the same page again. Our project mentor is also busy with his own work, so sometimes communication could be slow. We could have improved our communication by giving him more information on our individual contributions throughout the term, possibly in email form, so that he could catch up with us at any time.

---

7. Summary

Major Challenges

Our biggest challenge during this project was learning how to develop as a team. All of us have worked in team settings before and have created web applications before, but never in such a formal setting. Also, we have never worked for a mentor who had specific requirements for us that we had to tailor our app too. For that reason, it took many months before the team got into the groove of communicating and developing effectively with each other and alongside our project mentor. Fitting the development timeline and structure when many of us did free-form coding in the past was a real

Problem-Solving

A

Key Successes

A

Best Practices

A

Recommendations

The team has come up with a few suggestions for future improvements which another team may find useful in the future.

In the future, we recommend adding the ability to include PDFs as part of the assignment context, which would give the AI more detailed information to work with. Another important improvement is to expand caching across more API endpoints, since right now only one endpoint is cached. This would help improve performance and reduce server load.

To improve our testing, we should broaden our classroom testing across different subjects to see how well the platform works in various learning environments and uncover any discipline-specific needs. Using guided user testing scripts will help us collect feedback more consistently and make it easier to compare insights. We also want to expand our sentiment analysis by including more diverse language patterns which might help us develop our testing methodology further. Lastly, starting some A/B testing (especially for the instructor dashboard), might help determine which layouts work best for monitoring student progress and managing tasks.

---

8. Acknowledgements

This section is dedicated to acknowledging the contributions made to the project by our project mentor, Sanjai Tripathi – who was the mastermind behind the project, the five team members who developed the application, and the student's of our project mentor who tested the project and gave feedback on the application. Everyone who contributed helped bring this project to life and we hope that this project can be further developed in the future and is able to give value to the education space.

This document's sections and text were written by the team. However, some team members consulted ChatGPT for summarizations of internal documents and communications, which

helped inform the writing process. These contributions from AI were minimal, but present and acknowledged by the team.

---

eod.