T3DA

CS 46X: Senior Software Engineering Project Design Document

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

Group 28

Project Deadline: End of Spring 2025

Project Mentor: Sanjai Tripathi

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

Contents

Document Review

Version History

v1.0.0 (November 17th, 2024) – Initial Draft for Canvas Submission.

Author(s): Oliver, Sai, Team
Reason for Change: First submission Canvas.

- Design Introduction
- Frontend
- Backend, Database
- AI-Integration
- Activities

v2.0.0 (March 2nd, 2025) – First resubmission for CS462 document review.

Author(s): Oliver, Sai, Team
Reason for Change: Fitting document review standards for CS462 submission

Overview:
The winter development term revealed some new insights on the planning of the design document. Mainly by the way we designed the user interface. So things were revised in the document to accommodate changes and considerations in our development process.

Document Review

The design document will change drastically after initial prototyping and testing. Also design decisions might be changed if the project mentor desires different requirements or a different implementation of features. This section is for showing the changes between the documents between submissions. The document is reviewed alongside the project mentor.

The initial submission was revisited to format it according to the suggestions made by the revision assignment. The revision assignment suggested to assess our document with the following criteria:

- **Clarity:** Are all requirements stated clearly and unambiguously?
- **Completeness:** Have you captured all necessary functional and non-functional requirements?

- **Consistency:** Are there any conflicting or duplicate requirements?
- **Feasibility:** Are the requirements realistic and achievable within the project's scope and timeline?
- **Traceability:** Can each requirement be traced back to a specific stakeholder need or project goal?

Much like the previous document review, Oliver and Sai conducted a thorough review of the design document to follow the suggestions made in the review assignment instructions. We made several reconsiderations in this document depending on how our design has changed over time.

Much of the design document morphed into the task list which a new section has been constructed for.

As for other changes, we also considered that the AI tasks in our project may be executed better if they were done using a different technology. One technology that stood out to us was the recently released Deepseek. While this is under consideration, it will only be properly replaced in the project if testing shows that it is more promising. For now, it only exists as a consideration in the document changes.

As for the styling of the document itself, we believe that it still follows good formatting practices much like the requirements document that came before it. For that reason, we did not feel the need to change too much of its formatting. Minor inclusions like heading fixes and organization were done, but common styling formalities like organized headings, numbered pages and sections, and labeled figures were all included.

Preliminary

Introduction

This document exhaustively lists out the core components of the project and describes how they will be implemented through design specifications. This document serves to give a better idea about how the software will be structured and implemented, linking between the requirements and implementation components of the project timeline.

In the following sections in the preliminary, we will mostly list what core components are in the project in concise format. This is for reference as we describe our frontend, backend, etc.

Key Features

1. **Task-Delineated Assignment Creator:**
    - Allows instructors to create and manage assignments.

2. **AI Interaction Bot:**
    - Facilitates interactions between students, instructors, and the AI.

Core Components

Frontend

Role: User-facing interface for instructors and students to interact with the system.

Functions:

- Collects inputs (e.g., assignment parameters, AI queries).
- Displays outputs (e.g., generated assignments, AI responses).

Backend

Role: Handles internal logic and facilitates communication between components.

Functions:

- Processes user inputs from the frontend.
- Interacts with external APIs and databases.
- Generates (and serves?) assignment objects and AI responses.
- Contains the database

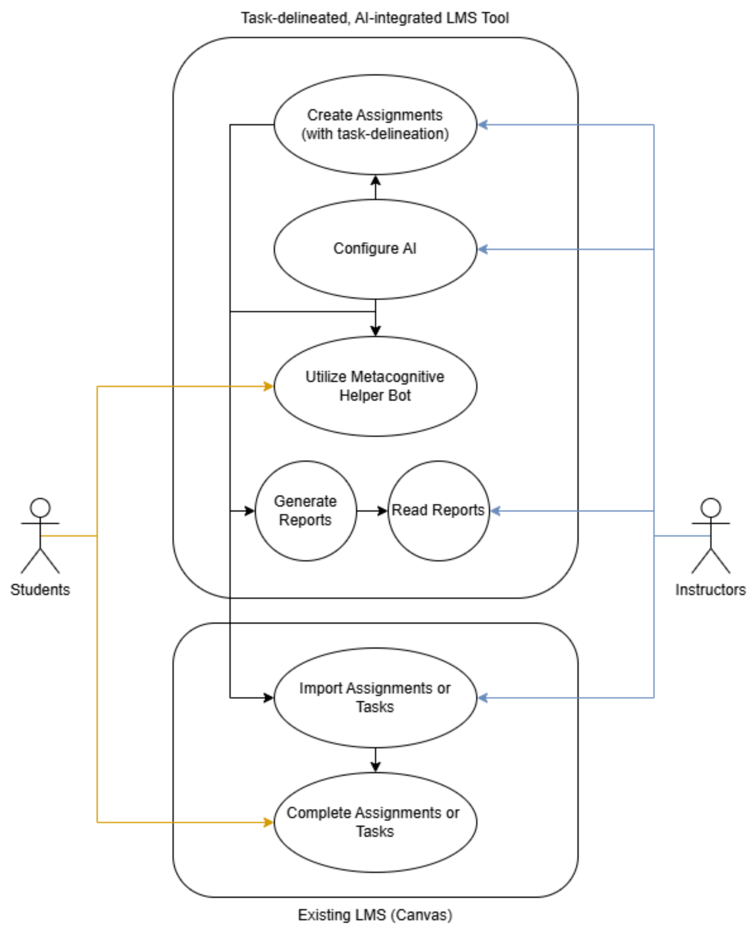
AI Components (External)

Role: Provides AI features for interaction and assignment creation.

Functions:

- AI APIs are queried to process instructor and student interactions.
- Response generation.

Figure 1 – Use Case Diagram


Task-delineated, AI-integrated LMS Tool
Create Assignments (with task-delineation)
Configure AI
Utilize Metacognitive Helper Bot
Generate Reports
Read Reports
Students
Instructors
Import Assignments or Tasks
Complete Assignments or Tasks
Existing LMS (Canvas)

The use case diagram shows how the two types of users of the webpage will interact with the program. This serves also as a basic layout for how each component interacts with one another.

1. Web Interface

1.1 Overall Structure - React, TailwindCSS, Vite

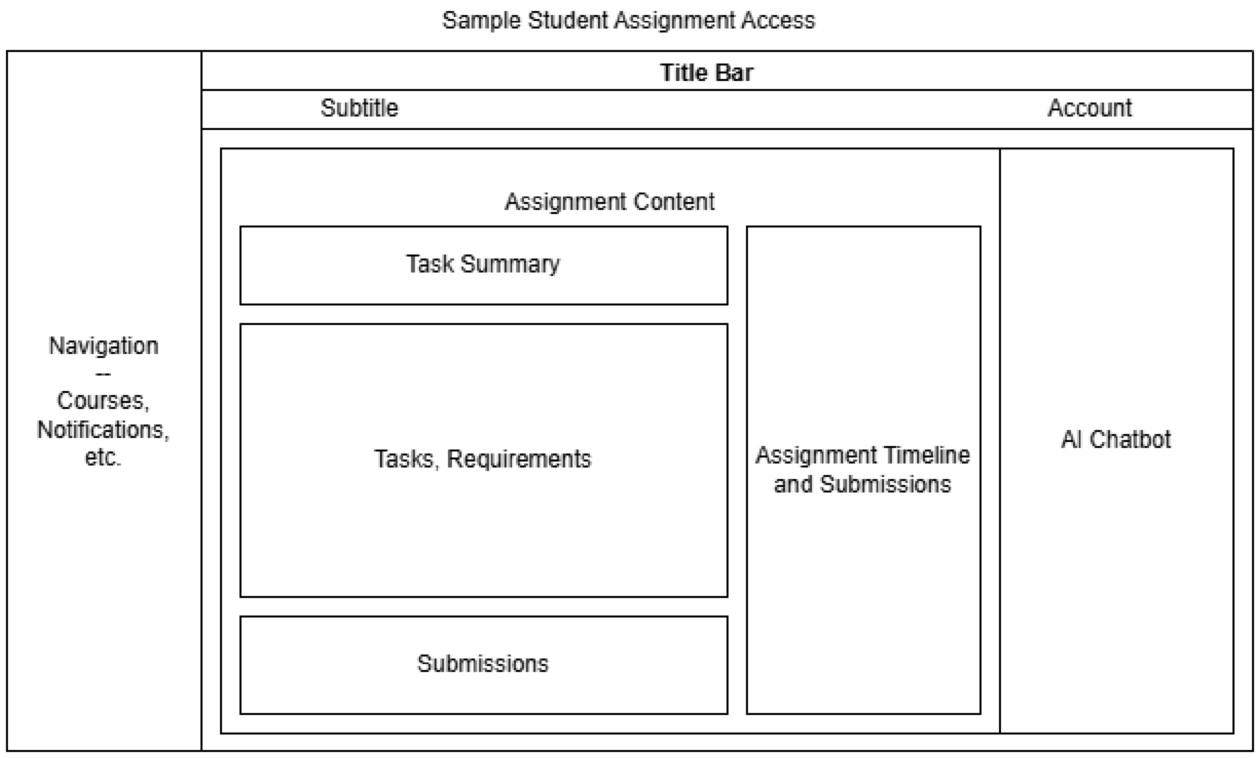
The web interfaces developed mainly using React and a combination of Vite. It features a layout with a header, sidebar, and content area. The header will display the navigation path, ensuring users always understand their location within the application. The sidebar will function as the primary navigation hub, offering collapsible menus for key sections, including “dashboard”, “courses”, “chatbots”, account management tools, and customizable settings. The content area will dynamically update to display the sections selected in the sidebar, ensuring seamless navigation. This interface will be created with the shadcn component library, which provides accessible and responsive React components.

The dashboard will display notifications, courses, and chatbots in a manner similar to Canvas. The “Courses” section will display all courses created by the user, with the ability to add new courses. Courses will show all assignments created for that course, along with options to add, group, and delete assignments. Additionally, users will have the ability to interact with student submissions for each assignment. Assignment creation will feature input fields designed to align with Canvas's format, enabling copy-and-paste functionality. In places where errors occur or where content isn’t found, generic 404 pages will be outputted to users. Similarly, where students are restricted access to content, they will also have similar pages which tell them that content is restricted. These pages will have minimal UI elements in them.

Instructors will need to configure AI tools on the frontend, so inputs and appropriate widgets need to be placed in course pages and assignment pages to enable interaction with it. Parameters will be listed and configured by dropdowns, boxes, and manual inputs. The "Chatbots" section in the sidebar will allow users to view all previously configured chatbots, with the ability to customize each one. This includes providing relevant context through text, PDFs, or other resources. Furthermore, users will be able to view generated reports from the AI about student submissions and performance. Performance optimization will be handled by utilizing Vite to remove unnecessary JavaScript and CSS code. Vite also additionally allows faster deployment times which makes it easier for loading during the development process.

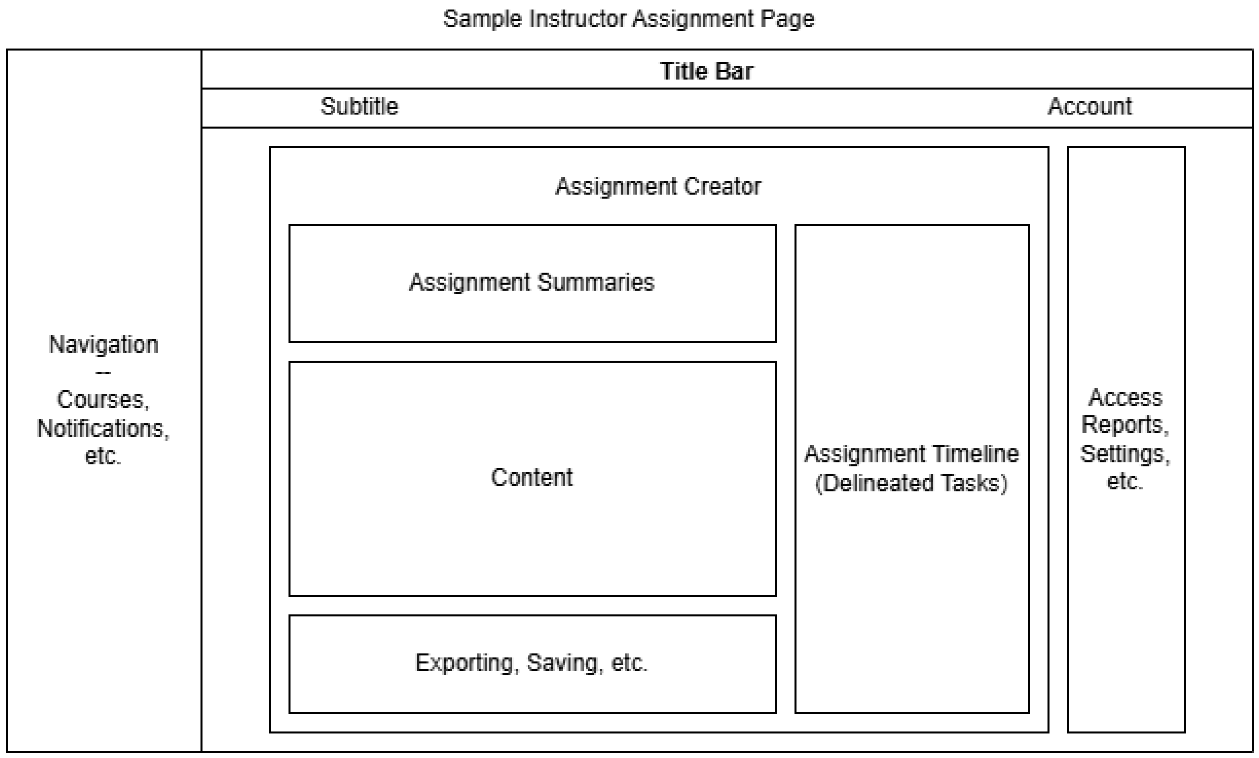
1.2 Assignment Layout Mock-ups

Figure 2 – Sample Student Assignment Access UI


Sample Student Assignment Access
Title Bar
Subtitle
Account
Navigation -- Courses, Notifications, etc.
Assignment Content
Task Summary
Tasks, Requirements
Submissions
Assignment Timeline and Submissions
AI Chatbot

The sample student page insures content from the assignment and a panel dedicated to AI interaction is present in Figure 3. In the next page, Figure 4 also shows a similar user interface, but demonstrates the assignment creation tool. The assignment creation tool allows the instructor to enter in assignment information, configure AI features, delineate tasks, and view submission reports.

Figure 3 – Sample Assignment Creation UI


Sample Instructor Assignment Page
Title Bar
Subtitle
Account
Navigation -- Courses, Notifications, etc.
Assignment Creator
Assignment Summaries
Content
Exporting, Saving, etc.
Assignment Timeline (Delineated Tasks)
Access Reports, Settings, etc.

1.3 State Management and Backend Communication - Redux Toolkit

Redux Toolkit will primarily manage the global state of the web interface, ensuring consistent data flow across the app. It will handle tasks such as storing user data, course information, assignment details, and chatbot configurations. Additionally, Redux Toolkit will facilitate communication with the backend through CRUD operations.

1.4 Bot Communication Interface

The interface for communication with a student helper bot (which extends to the same interface for student reports) will appear similar to chatbots that are dedicated to natural human responses. That is to say, it will try to serve conversations in a way that feels like users are speaking personably with a human. The layout will look similar to a chatting app on a phone, not dissimilar to the interface found in general purpose chatbots such as ChatGPT. Figure 5 shows a mockup of what that could look like.

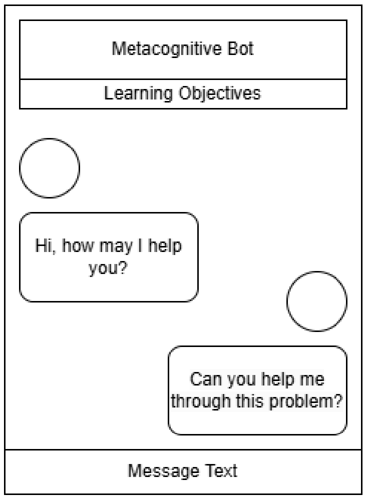
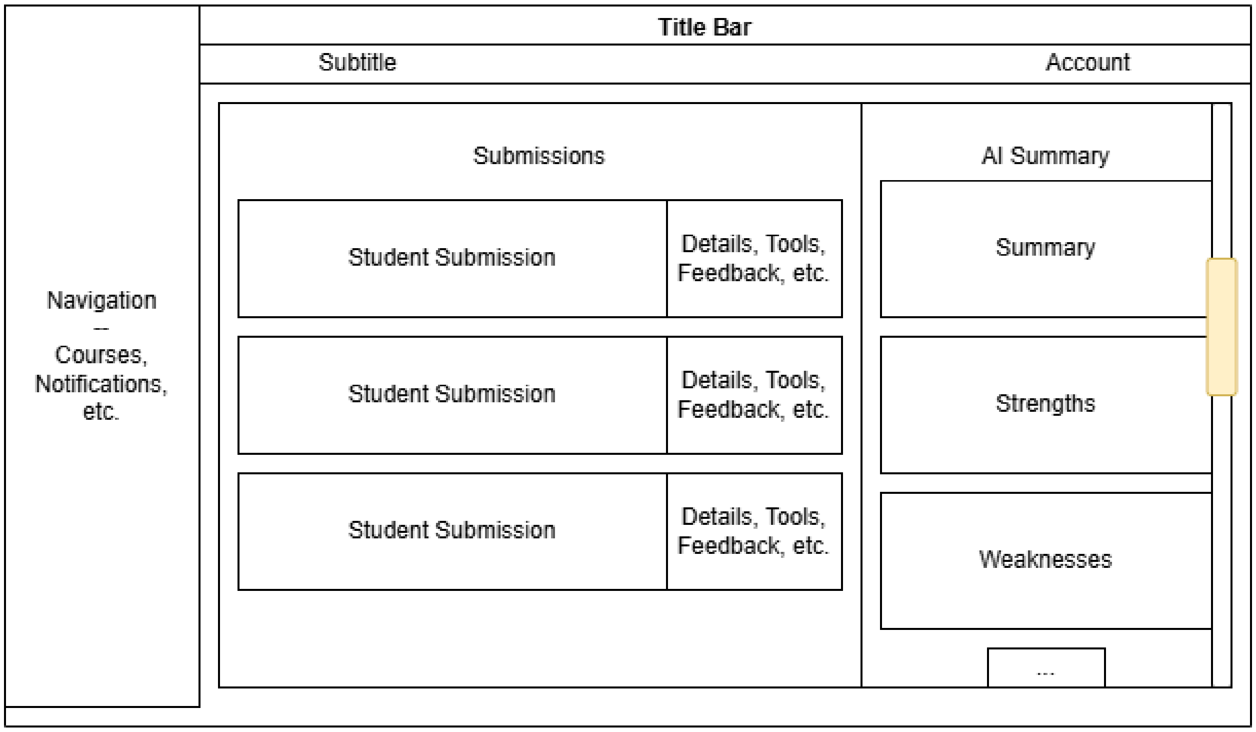


Figure 4 – Metacognitive Bot Chat

1.5 Additional Mockups

Figure 5 – Student Submissions and AI Reports



The left side of the submissions page can contain standard student submission features, but the right side will contain a panel which allows space for AI summaries and reports to be given. The specifics of this implementation likely lie on multiple pages and popup panes, but this serves to show a basic possible layout. This is more flexible to change as implantation occurs.

1.6 Mobile Support

Mobile support of web pages is crucial especially for education, which increasingly takes place on mobile devices in this age. However, due to limitations of the project timeline, and the scope of the project, mobile support will most likely only extend to at most a flexible implementation of the user interface that scales with resolution sizes to fit on any display. However, direct apps or dedicated mobile user interfaces will be implemented. Implementing a flexible interface will involve smart implementations of CSS – nothing unusual.

1.7 Task List

For the purpose of organizing the project into executable tasks based on project mentor requirements, we assembled a task list that outlines the design processes for each feature of the project. They assume the following table:

Figure 6 – Development Task List

Task	Purpose
1. Tracking and Analyzing Student Activity. Instructors can track students' progress through tasks and utilization of AI and receive analytic reports on both.	
Create a user account system with unique identifiers for students.	An instructor needs to be able to track individual students. Implementing an account system will enable professors to track their students.
Individually log AI API requests from student's assignment pages	Tracking student activity requires logs of student chat history to be reviewed by the instructor.
Feed a students log into a an API request configured for summarization in the individual student performance section	The instructor must be able to check if student's are using the AI appropriately. This must be done in a summarized format for quick confirmation.
Add a flag onto API responses which contain content deemed as non-compliance. Use a filter for catching these.	Non compliance is necessary to evaluate how the AI is being used so that changes and feedback can occur.
Have all student logs aggregate into a single request, generating a class summary	A single AI sumamrization of all student logs can identify overall class performance, so that the instructor can understand the usage of AI assistance.
2. Assignment and Task Creation: Instructors can create assignments within a web tool, breaking them down into multiple delineated tasks with their details, submission items, deadlines, etc., and assign them to students or groups.	
Create an exportable object format of assignments generated in a rich-text format	A pastable and exportable object will allow tasks from the website to integrate seamlessly with Canvas. The rich-text format is necessary for easy pasting into Canvas.
Prompt pre-set templates with fields for assignment content.	Efficient and organized assignment design requires strong outlines for assignment strucuture. The instructor can complete assignment creation tasks easily in this structure.
Flex assignment content	Assignment content should be widely configurable for different tasks which require more or less dense content.
Class Invite Codes	Instructors can send invite codes to students
Create an intuitive and organized assignment structure focusing on being intuitive to use and navigate. i.e. Fit important content (requirements) in visible and organized fields.	The UI design needs to be easily understood for students to engage with the material clearly and concisely.
3. Configure AI Tools: Instructors can configure AI assistance for specific tasks, tailoring chatbot responses to assist students and avoid enabling misuse. They can also designate when and where AI tools are to be used.	
Configure API responses. Return outputs from AI API.	API responses enable the usage of the third party AI tool for the website. This is the core necessary functionality for

	everything else.
Log responses from API, store them in database.	The database needs to store logs to later feed into summarization for the instructor.
Add default parameters to AI tools: Student summary, student helper, class sumarry.	Individual configurations of each AI tools need to be configured so they appropriate act upon their intended use case
Add parameters to helper bot: beneficial_assistance, allowed_assistance, inappropriate_assistance	These parameters will help the bot determine which uses are acceptable for analysis and feedback.
Student tracking configuration: Add two options for simple and advanced summaries -- include default paramters: Overviews, and detailed breakdowns	The instructor needs flexible ways to understand performance with minimal effort, for the ease of their workflow.
Remember output history	The output history needs to be rememebered so that the tool can understand context from previous outputs.
Add into UI control system that is interactable that tunes bot performance	This control system provides instructors with the ability to set advanced configurations for the AI bot.
4. Exporting Assignment Objects: Instructors can generate exportable or pasteable versions of their delineated assignments to import into LMS such as Canvas.	
Functional export buttons on assignment creation page	The functional buttons on this page will allow the users to find and execute export functions.
Format conversion according to Canvas	The format of the exported object needs to fit with the current Canvas format so that there is a seamless experience importing assignments from this tool
5. Metacognitive Bot: A bot that can understand the context of course material and specifically output prompts to students which promote metacognition and assist students with summarizable information.	
Contextual Understanding of Course Material	The bot should analyze and interpret the content within the course to generate relevant prompts that guide students toward deeper thinking.
Generation of Metacognitive Prompts	The bot should create tailored prompts that encourage students to reflect on their learning process, enhancing self-awareness and critical thinking.
See assignment, launch bot to help with this assignment button with course context to guide through it	
Summarization of Key Information	The bot should provide students with concise and meaningful summaries of course material, aiding in comprehension and retention.

2. Backend

2.1 Web Interface Communication - JavaScript

This will be a necessary point of this project as this will be the main method of how the user will communicate with the server using the web ui elements. This is also important for our implementation of the AI model as the JavaScript will link the inputs from the user and send them to the AI models so that they may generate responses. The user should be able to enter prompts within a given input box and this information will be sent to the AI models that will use the prompts to generate a response for the student. We will accomplish this through our JavaScript web interface, as we will implement a text window to gather user prompts which will be stored in MySQL database. This MySQL database is meant to store both the student prompts and AI responses. Furthermore, the prompt will be sent to our AI chatbot through an API request, who will also respond in its designated text window.

The communication between the server and the user will also be important for the entire interaction with the tool. Creating assignments, the helper bot, report generation, and importing the tasks or assignments will all require the usage of a proper communication channel with the server and the interface as all the information gathered from the interface will be either modified or used in a way and shown back to the user. This project will accomplish these tasks and functions through the use of Javascript and its subsequent features.

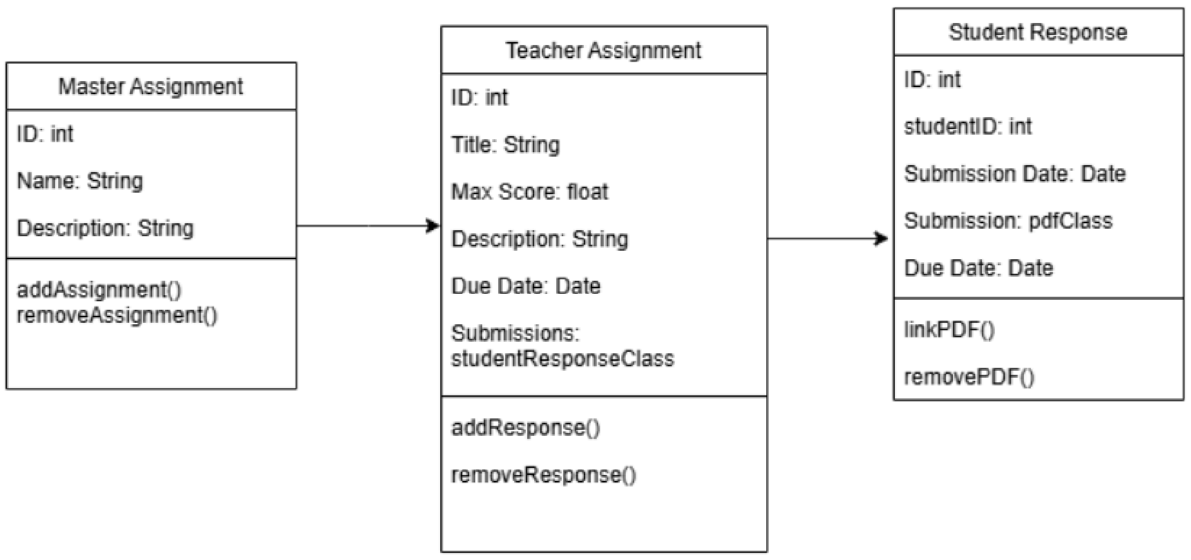
2.2 Assignment Creation Generation

To create and store our assignments, we will use Python scripts to create an assignment class. Our assignment class will store information such as text information from professors, but more importantly will contain classes for student assignment turnins. We will need to store student PDFs, Word documents, etc. in a MySQL database. In our Python script, we will create an array of an associated student assignment class which will contain data such as the assignment ID and upload date.

The assignment class will not be completely isolated, and assignments will be able to store an association with other assignments. This will allow professors to create assignments that are not encapsulated in strictly one file, but rather are task-delineated as they will be able to create an assignment in multiple linked steps, rather than isolated assignments. We may implement this linkage through a use of a master class for assignments which can contain multiple smaller assignments.

These assignments objects will then be fetched through by our JavaScript which will be able to display them on our web interface. Similarly, when students turn in assignments through the web interface, our script will make an API fetch to retrieve and store their document in our database, which professors will then be able to access through our web interface opposite of how students turn it in. Because our assignments will contain an array of student assignments, any individual student assignment that is turned in will be linked to that assignment object. Using these assignment classes, we can ensure professors can create task-delineated assignments which are linked through our backend to student responses.

Figure 7 – Assignment Classes



2.3 Databases - MySQL

Databases are another portion of the backend that would be useful for this project. This would mainly be linked to the AI to hold the student prompts and the responses that AI would generate back. We will be using SQL as the main language to implement this. The main tables that are included in the database would be used to store the assignment information, the user prompts, and the chat logs between the user and AI. Future usage will involve a more developed database as it continues and the requirements become more clear or grand.

2.4 Security - Python

For this portion of the backend work, it should be noted that it is unsure whether this will be a necessity for this project, but it should be noted that if this were to be implemented, python would be the main choice. The main tasks that would be performed would be encryption and decryption of information. Information should be safely handled by using hashing or digital signatures to ensure user safety and security. This is subject to future revisions as the project becomes more fleshed out.

2.5 Performance, Scalability

There are few performance requirements to design the software around because the project is small scale and likely does not make a significant amount of API requests to the AI model. So while performance requirements are more easily met at this scale, scalability will be a bigger concern. We will implement scalability by making core components microservices in the backend, so that they are modular and can be easily swapped as they develop. Performance of the website will be assisted in optimization tasks by Vite.

3. AI-Integration

3.1 Technology Selection

To achieve AI-integration, we will configure a preexisting LLM, starting with a model like GPT-Neo, and potentially exploring others such as OpenAI's GPT-4 or GPT-J, and possibly a more proprietary AI model like CoPilot, but that has not been decided on. We will configure the selected LLM, using course material, to ensure it can reasonably and consistently respond to prompts given by students.

Figure 8 – Technology Comparison

Feature	GPT-Neo	OpenAI
Security	Self-managed hosting; security is user-dependent	OpenAI manages hosting
Data Privacy	Full control over data; no third-party involvement	Data handled by OpenAI
Security Risks	Vulnerable to misconfigurations	OpenAI's infrastructure manages security, prone to a corporation
Community Support	Community support via EleutherAI	Developer support on OpenAI forums
Documentation	Comprehensive and community-driven.	API examples from OpenAI
Dependencies	None	Managed by OpenAI's proprietary infrastructure

We did a comparison of various technologies, but wanted to highlight two from specific categories of existing LLM: open source and large-scale. We chose the open source option of GPT-Neo rather than a model such as OpenAI because we figure there are less security concerns on a smaller project that is community driven and open source. We also have more control over our data security, which is necessary for the institutions using this website.

3.1.1 Other Technology

As the document changes, new technologies arise, with one example being Deepseek AI, which is known to be both powerful and efficient on runtime. This technology will be experimented on and it will replace our existing AI API if development decides that it performs better than the current option. This however, will not be used otherwise.

3.2 Configuration

The configuration of the different AI tools is important because each AI component serves a different purpose in the project. Configuration is done by interacting with parameters set by the backend. Instructors send inputs from their frontend, which is passed through as parameters set into API calls in the AI interface. It’s likely that these parameters need to be specified every time it's making a request because it needs to keep context about its responsibilities and the material it understands.

3.3 Student Chatbot

The primary use of AI in the project is to create a metacognitive bot that is usable to students. They will be able to prompt and ask the chatbot questions related to course material, and the chatbot itself will be able to give reasonable advice/guidance towards the user. Professors will also be able to configure where the chatbot is used in their assignments, and customize it so its responses will align with specific learning objectives.

Since it uses pre-existing technologies, training an AI-model is not necessary for the chatbot. Rather, configured parameters are geared towards being helpful to students is necessary here. This comes down to placing keywords that are effective in keeping the bot specified to the task. Parameters will indicate that the chatbot shouldn’t be giving homework answers, but focus on guiding students through understanding the material. Professors interact through the frontend, passing through the backend, and sending an API call which processes their options for the chatbot to align it with specific learning objectives or particular parts of an assignment that the AI assistant should guide students towards. The student chatbot will have a button next to the assignment that allows it to launch the helper bot. The helper bot assists using context fed specifically to it from course content.

3.3 API

Since the technology is not decided on (will be decided through testing during the development period), the API calls made are not decided yet. This section will contain API calls, URLs, types, and other API data necessary for functionality in this section.

3.4 Report Generation

Report generation will use the same technology as the student helper bot, but configured for this specific use case. For a professor or teacher to create a report, they will use our frontend web interface to make an API request to our integrated AI model. Our AI model will have a configuration set to generate a report which it will be set to upon a report request. The API fetch request will also fetch the assignment itself and any student prompts and AI responses generated during the completion of the assignment which will be stored in our MySQL database. This information will be sent through our API to our integrated AI model which will generate a report from it.

This report will then be stored in our database. Upon its completion, the frontend will make an API fetch request to the database or AI model itself for the full report, which will be returned to the frontend for a teacher to analyze.

Some of the work for these features will be implemented through backend or web UI developers, such as correctness of multiple choice/text answers to questions. Similarly, student prompts and AI responses will have to be stored through a backend database. However, our AI model will analyze a student's implementation of it so it can give feedback to a professor about how it was used or how a student progressed through an assignment. We will have to configure our model separately from other chatbot configurations meant to answer questions with this specific analytical configuration.

Activities

Oliver Zhou

Describe your Architecture

Individual Activity

Clearly define the overall system architecture of your project, focusing on key components and how they interact.

The overall architecture of the project consists of a few key components based around the 2 main features necessary to meet the project requirements. The first of which consists of the task-delineated assignment creator. The second key component is the AI bot that interacts with students and instructors. The AI components are embedded into the assignment creation/interaction components.

Identify Core Components: List and describe major components (e.g., frontend, backend, database, external APIs, functions, data sources, …).

The website consists of a frontend and backend. It's possible that we use a database to store assignment data but this possibility depends on the scope of the project because it's possible that we only export assignments. External APIs may consist of the AI APIs that we use for the project.

Define Interactions: Explain how components communicate (e.g., API calls, data flows).

For a basic flow of the components. The Frontend components will provide access for the user to use the webpage's features. This can be accomplished in a variety of ways, starting with the task-delineation assignment creator first. The assignment creator takes in user inputs from an instructor and passes it to the backend. Backend code creates the actual assignment object and outputs it to the instructor or student. This may be sent to a database for storage depending on the extent of the project. For interacting around the assignment, the student may input requests to the AI component that makes API calls to an existing AI interface. That response returns to the user through the frontend. A similar event occurs for professors when generating reports.

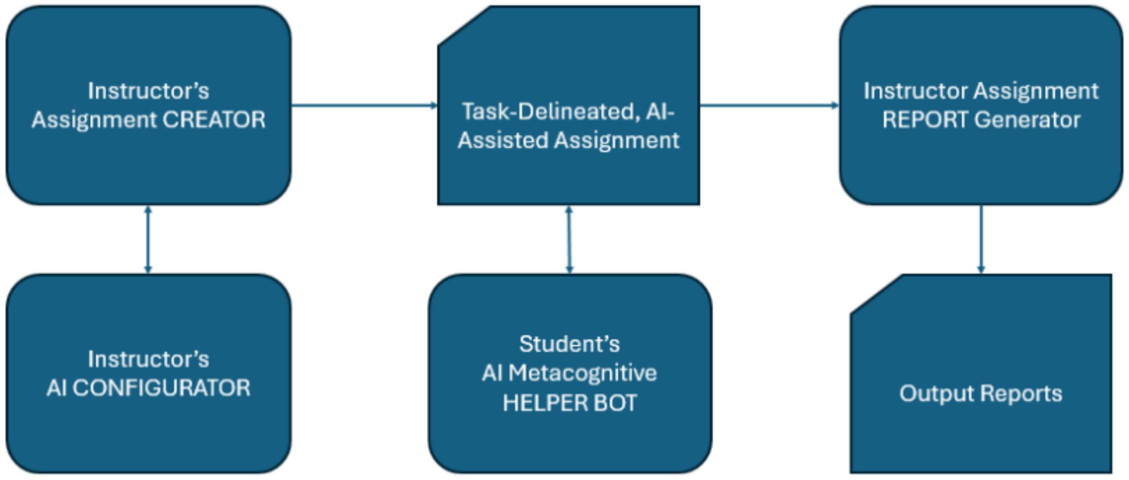
Consider Scalability: Discuss how the architecture handles future growth.

For future growth, the database is the main part that needs to be scaled. As for the frontend and backend, these might need features that allow larger class sizes such as group features etc. Also, API limits would need to be increased if the project gained an actual user base.

Security and Performance: Identify security measures and performance optimization techniques integrated into the architecture.

We may need backups in the database of the webpage so that assignments can be safe and secure and data won't be lost. Also, since class materials are generally sensitive and need to be restricted from students, we need to make sure any data doesn’t accidentally leak onto the frontend for students.

Submit a diagram of the architecture and a brief explanation of each component and its role in the system.


Instructor’s Assignment CREATOR
Task-Delineated, AI-Assisted Assignment
Instructor Assignment REPORT Generator
Instructor’s AI CONFIGURATOR
Student’s AI Metacognitive HELPER BOT
Output Reports

The assignment creator creates the assignment objects for the web page. The configurator interacts with the instructor and changes the requirements of the assignment objects. That together outputs the assignment portion of the task-delineated, ai-assisted assignment. That also takes inputs from the AI metacognitive helper bot, which embeds itself into the assignment page. Outside of the assignment, submissions are processed through the report generator which outputs reports to the instructor.

Evaluate Different Technologies

Individual Activity

Compare and assess multiple technologies (e.g., languages, frameworks, libraries, …) to choose the best fit for your project’s requirements.

This activity is conducted to decide which AI technology is best suited for the needs of our project.

Identify Key Criteria: Define what you need from the technology (e.g., performance, scalability, ease of use, …).

The primary feature we need from the tool is some level of configurability and flexibility. We should be able to set the parameters of the tool easily so that it can be repurposed for each feature that we use the AI in the project. If we don’t meet this requirement, then responses may either be unhelpful, unrelated, or easily misled by irrelevant queries.

Research Options: Explore 2-3 technology solutions and their strengths/weaknesses.

GPT-Neo:
- Good for open-source alternatives to GPT-3/4 with complete control over the model.
- Requires significant resources for hosting and tuning but offers flexibility.

OpenAI GPT-4:
- Industry-leading performance in NLP tasks with accuracy.
- Ideal for enterprise solutions or developers needing a powerful API.
- Possibly too large scale and general for our use case

GPT-J:
- Similar to GPT-Neo in being open-source, offering a balance of performance and cost-efficiency.
- Suitable for control over the model and have the infrastructure to support it.
- Simple responses, low-context(?)

Compare Performance and Compatibility: Evaluate how well each technology integrates with your existing project and meets performance goals.

I think that researching each technology has led me to believe that we should give GPTNeo a try. I think it offers the best flexibility for our use case while being able to output reasonable responses. It at least is more promising and seemingly easier to integrate than a model such as OpenAI’s GPT-4.

Security and Support: Consider security features and available community support for each option. Provide a comparison matrix and a justification for the chosen technology.

Feature	**GPT-Neo**	**OpenAI**	**GPT-J**
Security	Self-managed hosting; security is user-dependent	OpenAI manages hosting	Relies on self-managed hosting; security is user-dependent
Data Privacy	Full control over data; no third-party involvement	Data handled by OpenAI	Full control over data; no third-party involvement
Security Risks	Vulnerable to misconfigurations	OpenAI’s infrastructure manages security	Vulnerable to misconfigurations
Community Support	Community support via	Developer support on	Community support via

	EleutherAI	OpenAI forums	EleutherAI
Documentation	Comprehensive and community-driven.	API examples from OpenAI	Comprehensive and community-driven.
Dependencies	None	Managed by OpenAI's proprietary infrastructure	None

---

Collin Kimball

Optimizing System Performance

Performance metrics:

- Response Time
- Latency
- Scalability
- Availability

Caching API Responses:
For ChatGPT API integration, a primary metric is the API response time, which averages between 15-20 seconds. Minimizing this delay through caching and system optimizations is a top priority to ensure a seamless user experience. Frequently requested responses from ChatGPT can be cached using in-memory storage solutions like Redis. By storing responses for repeated queries, the system can serve cached data instantly, avoiding repeated API calls and saving the 15-20 seconds per request. Similarly, other frequently accessed data can also be cached, such as chatbot configurations, or course and assignment details.

Rate Limiting:
To manage API usage effectively and prevent overloading the backend, rate limiting can be implemented. By controlling the number of requests made per user within a specified timeframe, rate limiting ensures that API calls remain within allocated quotas and that the service operates reliably under large traffic levels.

Database Sharding:
Database sharding can be utilized to improve scalability and reduce latency by partitioning data across multiple servers. This approach ensures that large datasets, such as user chatbot logs, are distributed evenly, reducing query load on individual database instances. This may not be necessary depending on the number of users.

Complete a Tutorial

The tutorial that I selected was focused around React hooks. I learned how custom hooks can encapsulate reusable logic, making components cleaner and more maintainable, which will help with fetching data from the backend in a modular way. The tutorial also covered hooks such as the useMemo hook, highlighting its role in optimizing performance by memoizing computationally expensive functions, preventing unnecessary re-renders. This will help to improve the web interfaces efficiency and code maintainability.

Ethan Lu

Select a Database or Storage Technology:

Oracle

Pros:

- High scalability. Designed to handle high-volume workloads.
- Has many built in security features.
- Cross platform across Windows, Linux, etc.
- Good at storing predefined data types.

Cons:

- High licensing fees.
- Lack of open-source contributions/support.
- Used for large databases and has less application for smaller developers.
- Higher learning curve - None of our developers have much experience with Oracle.

MySQL

Pros:

- Open source, free software, good for small developers.
- Cross platform across Windows, Linux, etc.
- Good scalability for both small and large applications.
- Multiple open source plugins and solutions. Commonly used, so issues are easier to solve through community.
- Lower learning curve - All of our developers have experience with MySQL.

Cons:

- Certain features are locked behind MySQL's paid versions.
- Reports of concurrency issues that Oracle typically does not have.
- Can be very slow to alter large tables.
- In general can run slowly and can be difficult to manually modify tables.

Overall, I believe that our group will choose to use MySQL over the alternatives. The first and primary reason is simply that it is free and open source, meaning that we can use it without any issue for developing our software. The other primary factor to take into consideration is that all of our developers have had experience using MySQL previously due to taking courses previously during our OSU curriculum. This means that we will have a level of familiarity with MySQL which can reduce the workload of understanding it and help development in other areas.

Furthermore, the primary alternative that I looked at was Oracle, which is commonly used by larger business models. However, as we are merely making a smaller-developed software, we are unlikely to need to store significantly large amounts of data for the time being. Furthermore, MySQL has decent scalability for both small and large projects, meaning that if our database is larger than expected, it is dynamic enough to handle our information.

Apply the Privacy by Design (PbD) Guidelines:

1. Proactive not reactive; preventive not remedial.
2. Privacy as the default setting.
3. Privacy embedded into Design.
4. Full functionality — positive-sum, not zero-sum.
5. End-to-end security — full lifecycle protection.
6. Visibility and transparency — keep it open.
7. Respect for user privacy — keep it user-centric.

1. During our implementation we will need to consistently conduct risk assessments. Our software should not gather any personal information or have any method to obtain sensitive information from our users that they themselves do not choose to share. Furthermore, we need to ensure any gathered data, such as student prompts and our responses are not accessible by third parties. We do not want to have to retroactively implement security measures, our software should be tested thoroughly beforehand.

2. During our implementation, we will gather as little user information as possible and ensure that user profiles cannot be publicly displayed or shown without permission. Professors will have access to student accounts but only be able to see information from them regarding their classes. Students only have to verify through name/email, and otherwise we will gather as little data as possible.

3. Our implementation will include encrypted communication channels. Fair information practices will be applied, and we will conduct rigorous security/risk assessments that will also be published to allow users/shareholders to understand possible security risks.

4. To ensure full functionality, our implementation will embed privacy in such a way that full-functionality is not impaired. For instance, our AI models will grade student submissions without gathering student information besides necessary associated accounts which will strictly and only be sent to the associated

professor. Both students and professors will be able to delete or modify any information associated with their accounts at any time.

5. During our implementation, we will need to implement end-to-end security. We must ensure confidentiality of any gathered data to our users, such as through implementing data-deletion features after specific time periods, where such retained data is no longer necessary. In this way, we can ensure end-to-end security.

6. In our implementation, we must ensure users fully understand what data is being collected, and how and where it is being used through a privacy-policy. We must also ensure users can see what data has been collected from them, such as students being able to access and see what prompts are currently stored and shared with professors.

7. We must enable consent settings for our users so as for them to understand what of their data is being stored. For instance, we may allow students to hide their prompts from even professors (though professors will likely have knowledge that the specified student has requested their prompts be hidden from them). We want to ensure student privacy and safety at all costs, and we wish to ensure users have full control over all data that is being shared/stored.

Trent Matsumura

Complete a Tutorial

The tutorial I chose to follow was a class assignment for databases. This describes how to use MySQL to set up a basic database that contains arbitrary information about planets. What I learned from performing this activity was basic knowledge of SQL commands and how to structure a database. It walked me through each step of creating a database including instructing various ways to access the database and how to edit it. This will be very useful in creating a database for the storage of data throughout the process of training the AI model and logging its data as well.

Test Plan

1. Identify Test Objectives: Define what you need to test (e.g., functionality, usability).
2. Create Test Cases: Develop detailed test scenarios for each feature.
3. Determine Test Methods: Choose manual or automated testing.
4. Set Success Criteria: Define what constitutes a pass/fail result.

1. Some test objectives that will be useful for the project would be to test the functionality of the AI chatbot as well as the ability to create assignments. Another big portion that should be tested is the report generation. Overall the test objectives can be summarized by saying, the test

objectives are to ensure the functionality of the ability to create assignments, allowing the AI chatbot to work based on the configured limitations, and having a useful report generated.

2. The test cases for the three is as follows:

- Create Assignments:
    - Give the assignment prompt by using the interfaces given
    - The assignment should be split up into tasks that are readable by the user
- Configuration:
    - Configure the chatbot to not allow giving answers out, instead only helping tips out, test whether it is still giving out direct answers
    - Configure the chatbot to only allow step by step responses to assist users. Test if the responses are following this prompt
- Report Generation:
    - Test the report generation by filling out the assignment in a sloppy and incorrect way to test if it will accurately describe the faults and errors.

3. Manual testing for all of these test scenarios as they are very human oriented and the functionality needs to be verified by a human as it will be exclusively used as a tool for humans. This means that each outcome and scenario needs to be set up and evaluated by a human.
4. Passes for the test cases are as follows:

- Create Assignments:
    a. Is the assignment is delineated and split up into tasks
    b. Is the assignment is related to the given prompt
- Configuration:
    c. Does the chatbot suggest banned topics or prompts
    d. Does the chatbot follow the prompts given to it
- Report Generation:
    e. Is the report readable and related to the assignment
    f. Can the report accurately describe the assignment

Sai Anand

Performance Optimization Plan

1. Define Performance Metrics

- Response Time: Measure the time taken for each system response (e.g., loading assignments, accessing task details).
- Throughput: Track the number of requests handled per second, especially during peak usage times.
- Latency: Ensure minimal delay in system processing, especially for AI-based feedback responses and assignment tracking.
- Error Rate: Monitor the frequency of errors under various loads to assess system

resilience.
- Uptime: Ensure high system availability, aiming for 99.9% uptime.

2. Plan for Load Handling
- Stress Testing: Simulate high traffic to identify the breaking point and establish thresholds for maximum system load.
- Load Balancing: Distribute requests across multiple servers to handle high traffic and prevent server overload.
- Implement a reverse proxy like Nginx for load balancing.
- Enable auto-scaling based on real-time demand.
- Concurrency Limits: Set a cap on the number of simultaneous users to maintain response quality.
- Consider queue mechanisms to handle peak traffic without compromising performance.

3. Optimize Code and Queries
- Database Optimization:
- Indexing: Apply indexing on frequently queried fields (e.g., student IDs, assignment IDs) to reduce query time.
- Caching: Cache common queries, such as frequently accessed assignments and user data, using tools like Redis.
- Code Efficiency:
- Minimize database queries within loops and refactor code for streamlined performance.
- Use asynchronous processing for non-critical tasks to free up system resources.
- Algorithm Tuning:
- Fine-tune AI model responses to quickly generate results without excessive computational load.

4. Monitoring and Scaling
- Performance Monitoring Tools:
- Implement real-time monitoring using tools like Prometheus and Grafana to track server metrics (CPU, memory usage, response time).
- Set alerts to notify the team when response times exceed defined thresholds or when unusual traffic patterns emerge.
- Scalability Strategy:
- Use containerization (e.g., Docker) and deploy on cloud infrastructure (e.g., AWS or Google Cloud) to enable easy scaling.
- Prepare for horizontal scaling by adding more instances when needed, with a load balancer to direct traffic.

5. Identify Bottlenecks and Optimization Strategies

- Bottleneck Identification:
- Use profiling tools like New Relic to identify code bottlenecks, query lag, and excessive memory consumption.
- Run load and stress tests periodically to reassess any performance bottlenecks as new features are added.
- Optimization:
- Optimize data retrieval times by revisiting the caching strategy regularly.
- Use data partitioning for large datasets to improve access speeds for high-demand data, like student progress and task tracking.

System Security Plan

1. Identify Vulnerabilities
- Data Breaches: Risk of sensitive student and instructor data exposure.
- Unauthorized Access: Potential for users to access restricted data or functionalities.
- Data Tampering: Possibility of malicious actors altering data, especially in assignment or progress records.
- System Misuse: Risk of malicious bots or unauthorized scripts exploiting the AI Canvas system, potentially overloading servers or manipulating content.

2. Implement Best Practices
- Encryption: All data, especially personally identifiable information (PII), will be encrypted both in transit (via TLS) and at rest (using AES-256).
- Authentication: Implement multi-factor authentication (MFA) for instructors and administrators, reducing risk in case of compromised passwords.
- Role-Based Access Control (RBAC): Access to features will be defined by roles (e.g., instructor, student, admin) with permission settings to restrict unauthorized data viewing or editing.
- Regular Security Audits: Routine internal audits and vulnerability assessments will be conducted to identify and close potential security gaps.

3. Secure Data
- Data in Transit: Secure through end-to-end encryption protocols (e.g., HTTPS/TLS) to prevent interception.
- Data at Rest: Use server-side encryption for all stored data, and ensure encryption keys are managed securely.
- Regular Backups: Automated daily backups with encryption will be stored offsite to maintain data integrity and recovery options.
- Anonymization for Student Data: Personally identifiable information will be separated and anonymized wherever possible, especially in data analytics processes.

4. Compliance and Monitoring
- Compliance with Regulations:

- GDPR: Implement strict data handling policies, including data access logs, explicit user consent, and user data deletion options.
- FERPA/HIPAA (if applicable): Ensure that student data meets federal privacy standards with restricted access and audit capabilities.
- Continuous Monitoring:
- Intrusion Detection System (IDS): Deploy IDS to monitor for suspicious activity and identify potential breaches early.
- Automated Threat Scanning: Real-time scanning for malware and anomaly detection to catch unusual access patterns.
- Alerts and Logs: Maintain detailed logs for access and changes, with alerts for unusual patterns, helping catch security issues early.

---

eod.