Using Generative AI to Assess Learning

FINAL REPORT DOCUMENT

sdmay24-36

Client/Adivsor: Henry Duwe
Adviser: Mathew Wymore

Team Members/Roles:
Mac Whitney: Team Lead
Kyle Geerts: Canvas Integration/API
Alec Frey: Prompt Engineering
Reese Jamison: Backend and Scribe
William Nash: Full Stack Design
Johnny Tran: Research and Testing

sdmay24-36@iastate.edu
https://sdmay24-36.sd.ece.iastate.edu

Revised: April 25, 2024

Executive Summary

Development Standards & Practices Used

- Programming language
    - Our program will be written in JavaScript. While there is no official standard for JavaScript, we will do our best to adhere to the JavaScript Reference. Python was used for the first design, but this was updated.
- Standard of Software Engineering Technology (IEEE 610.12)
    - Providing a glossary of software engineering terminology, establishing a common language, and understanding in the field.
- Standard for Security and Trustworthiness Requirements in Generative Pretrained Artificial Intelligence (AI) Models (P7018)
    - This standard establishes a comprehensive framework for mitigating security risks, privacy leaking in the development, deployment, and use of generative pretrained AI models.
- FERPA (not engineering, but said we should include it)
    - Establishes rules for student data and how it should be handled (need to know basis)
- JSON(ECMA-404)
    - Follow correct syntax for data transfer with JSON requests
- HTTP
    - RFC 2616
- IEEE Code of Ethics
    - Uphold the highest standards of integrity, responsible behavior, and ethical conduct throughout the development
- Prompt Engineering
    - Prompt Patterns
        - Describes the approach taken to craft prompts to elicit a specific response.

New Skills/Knowledge acquired that was not taught in courses

Tools

- OpenAI API
- Developer Canvas
- React.js/Node.js

Skills

- Integrating application in canvas
- React.js / Node.js
- Agile Workflow
- Prompt Engineering

Knowledge Gained

- Creating Effective Prompts for Generative AI
- Testing Generative AI

Table of Contents

Problem Statement

General statement: Utilize generative AI (ChatGPT) to develop and assess students' learning via a personalized conversational format to develop a deeper understanding of the reasoning behind students' answers.

Problems being solved:

- How to create more time for ambitious professors to complete more tasks/research?
    - Be able to automatically generate an exam by only providing the topic and lecture material used to cover that topic in their class.
    - Be able to have a thorough automatic grading system to grade the automated exams based on initial response to the question with support generated from the AI. This cuts down time spent on grading, as well as being able to generate better analytics on why students understood or misunderstood the topic being quizzed.

- How do we give students a better opportunity to show an in-depth understanding of a topic when being tested?

    - This project will allow students an adequate ability to show their understanding of a topic in a conversational format with an AI chat bot to probe farther into their understanding.

    - The system will probe the student to gain a better understanding of how they are thinking, leading to more in-depth grading. This allows for a better representation of what the student knows.

- How are we able to solve these problems today?

    - We are now able to solve these issues because AI's learning models have gotten to a large enough scale and can accurately hold topic focused conversations that allow for in-depth question generation and response understanding.

    - OpenAI allows for an extensive modeling program that can solve difficult problems with greater accuracy, thanks to its broader general knowledge and problem-solving abilities.

    - Utilizing prompt engineering helps to train the AI to act as expected for the project guidelines, and ensures it is viable for conducting a class quiz.

- Pros:
    - Automated test generation and grading allows for faster collection of students' understanding of the topics covered in class. It also creates more time for instructors to focus on other tasks.

    - Allows the ability for students to show a deep understanding of a topic.

        - If students do not have a deep understanding of the topic, the system will be able to get a superficial idea of what the student is thinking and grade them accordingly.
- Cons:
    - The lack of human input to grading for academic success.
    - AI regulations and ethical practices are in their infant state.
    - Exams could be biased towards some students depending on the conversational responses given by AI, as the questions will be unique per student.

Intended Users and Uses

- Professors and Students
    - These are the main people who will use and benefit from this system.
    - Professors will benefit through increased efficiency brought with the AI tools for grading purposes. This will not only benefit professors by saving time making quizzes, but the goal is for a better quality of exams for both the students and professors.
    - Students will benefit in their studying from a personalized learning tool built specifically for the specific course. The conversation will be tailored to each individual student, making a better learning experience for all.
    - Professors will input class material into the AI, in which the AI will analyze and generate a series of questions to test the understanding of the students. Students will have an interactive exam, in which they will receive feedback and further questions to examine their understanding of the given class material.
- Teaching assistants
    - This is another group of users that will also benefit from the system. The generative AI model will save them time (re)grading quizzes, and hopefully save time explaining basic concepts and/or topics to students (possible secondary use).

Similar Products on the Market

The most similar product to the generative AI model that we are using is called PrairieLearn. PrairieLearn is an online assessment and learning system that empowers instructors to create robust educational resources for students. It is an open-source software for creating and delivering learning experiences and assessments for students. Instructors easily write questions as code, which automatically generate and grade infinite variants of themselves. Students are encouraged to keep trying new variants of the same question until they achieve mastery. Some pros of PrairieLearn is there is real-time feedback for students, ability to train, and repetitions so students can repeat until mastery is achieved.

Gradescope is another software used in an equivalent way. Gradescope helps you seamlessly administer and grade all of your assessments, whether online or in-class. Save time grading and get a clear picture of how your students are doing. Some of the pros of Gradescope is that it can automatically grade students based on a rubric and provide accurate scoring, just like a TA would. Some cons include no back-and-forth interactions between students and the software.

We will be using prompt engineering to form specialized input prompts to elicit a specific response from ChatGPT. This will be done using prompt patterns, which will aim to provide context, and constraint for ChatGPT Output

References:

us.prairielearn.com
www.gradescope.com

Below are references to papers about prompt engineering and prompt patterns

White, Jules, et al. "A Prompt Pattern Catalog to Enhance Prompt Engineering with Chatgpt." *arXiv.Org*, 21 Feb. 2023, arxiv.org/abs/2302.11382.

Ekin, Sabit (2023). Prompt Engineering For ChatGPT: A Quick Guide To Techniques, Tips, And Best Practices. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.22683919.v2

KorzynskiP., MazurekG., KrzypkowskaP., & KurasinskiA. (2023). Artificial intelligence prompt engineering as a new digital competence: Analysis of generative AI technologies such as ChatGPT. *Entrepreneurial Business and Economics Review*, *11*(3), 25-37. https://doi.org/10.15678/EBER.2023.110302

Revised Design

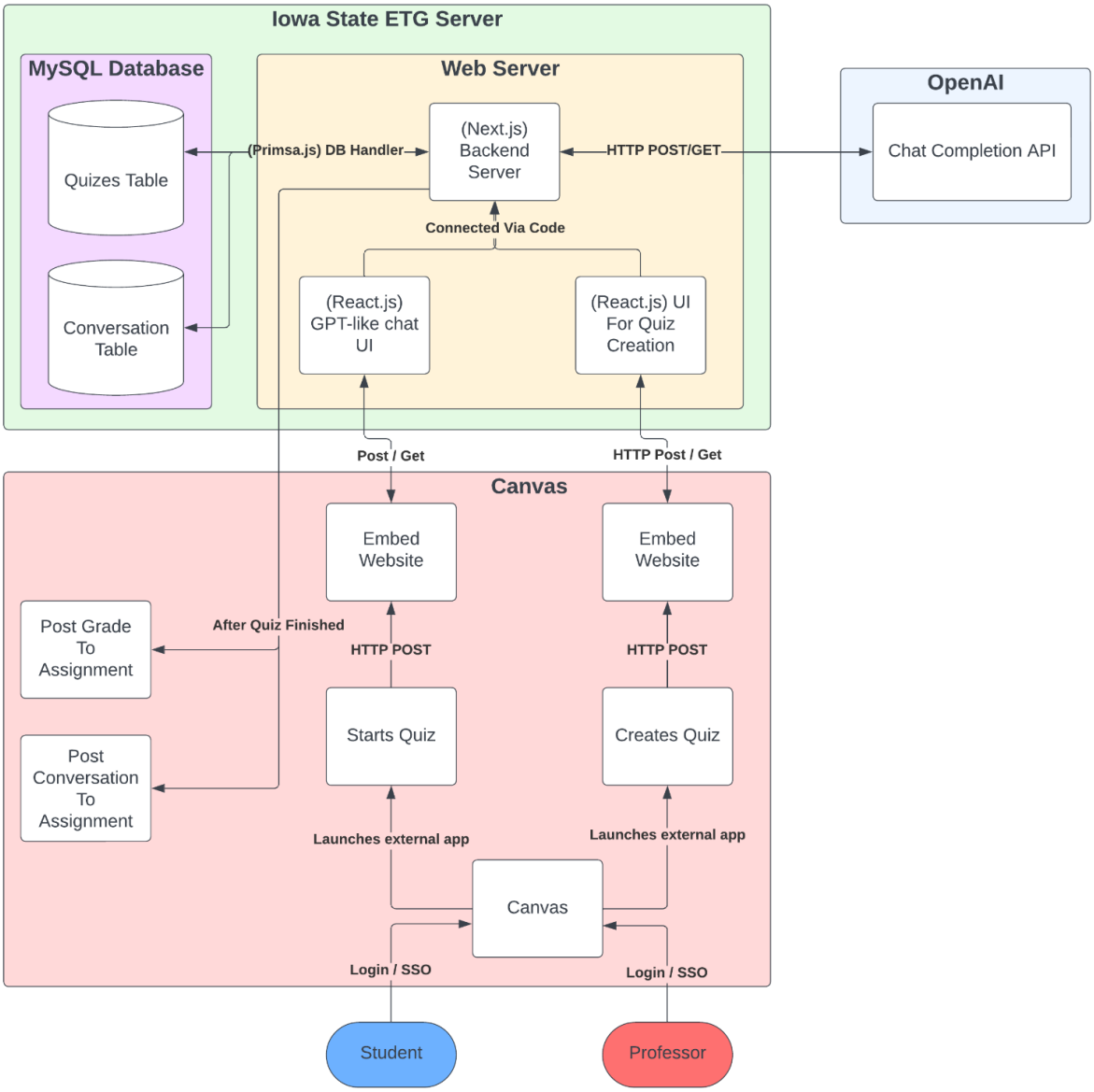

Figure 1

Requirements

Requirements:

- Based on the given information on a topic, be able to hold a back-and-forth conversation between the student and the AI chatbot. Throughout this conversation, the AI chatbot can understand the way the student is thinking and lead them in a direction to gain a better idea of the student's thought process. While this conversation is happening, the AI chatbot will be able to grade the student on their answers based on the input given to the generative AI system. The grading feedback will be available after the quiz is finished.

- Implement the generative AI into Iowa State's Canvas so the system can quiz students directly from Canvas. This also allows grades to automatically be updated in Canvas and includes a link to the grading feedback as a comment.

- The professor would provide the generative AI with notes, lecture slides, research, general knowledge, or whatever is needed from them, so the system knows the scope of the class and what is expected to be talked about in the conversation and/or quiz. They will also provide general constraints and wants for the exam, these include primary topic, number of questions, type of questions practical or theoretical.

- Follow Iowa State's FERPA and netiquette standards so the generative AI stays in scope with the requirements that Iowa State has put forward. This would include proper language from the chatbot, protecting students' and teachers' personal information, and staying within the law.

- Utilize a large-language model to administer the quizzes.

- Should be able to hold around a 15-minute conversation (+/- one minute) with a student or follow the question total provided as context. Responses should than 30 seconds.

- Must meet ethical standards so there is fairness between students, limited bias, and quality control so different students do not get vastly different conversations, leading to a "harder" quiz for one student vs. another.

- Must have high availability. There should be limited downtime with the system so professors can use it whenever they need it, and so students do not run into issues during a timed quiz. This should not be a big problem as the server will be hosted by ETG services at Iowa State.

Constraints:

- The conversation and questions the AI asks a student must be within the scope of the class and/or the data provided to the student. This way, the student is not quizzed on something not discussed in lectures or notes provided by the professor.

- Responses from the AI must be appropriate and meet the netiquette standards of Iowa State. This includes language, respectfulness, honesty, and engagement in the conversation.

- The tone of the AI should be encouraging and not disrespectful towards the professor or student.

- Must have high availability so a student can finish their quiz without interruptions.

- The server will be hosted on Iowa State's internal network, so if the server would have interruptions, we would have to rely on Iowa State to fix the issues.

Engineering Standards

- Programming language
    - Our program will be written in JavaScript. While there is no official standard for JavaScript, we will do our best to adhere to the JavaScript Reference. Python was used for the first design, but this was updated.
- Standard of Software Engineering Technology (IEEE 610.12)
    - Providing a glossary of software engineering terminology, establishing a common language, and understanding in the field.
- Standard for Security and Trustworthiness Requirements in Generative Pretrained Artificial Intelligence (AI) Models (P7018)
    - This standard establishes a comprehensive framework for mitigating security risks, privacy leaking in the development, deployment, and use of generative pretrained AI models.
- FERPA (not engineering, but said we should include it)
    - Establishes rules for student data and how it should be handled (need to know basis)
- JSON(ECMA-404)
    - Follow correct syntax for data transfer with JSON requests
- HTTP
    - RFC 2616
- IEEE Code of Ethics
    - Uphold the highest standards of integrity, responsible behavior, and ethical conduct throughout the development
- Prompt Engineering
    - Prompt Patterns
    - Describes the approach taken to craft prompts to elicit a specific response, based on the current research published on prompt engineering

It is important to note generative AI topics are rapidly growing, so some regulations are developing. If these standards get developed as our project moves through the semester, we will add them to our design specifications. We acknowledge that most generative AI standards are not available but will be added appropriately when they arise.

From IEEE Standards Association:
"We are engaged in forward-looking measures to establish necessary standards and guidelines for ethically aligned and age-appropriate design, and work to address issues that require an informed public dialogue and remediate action. AI generative models leverage both established and cutting-edge computational techniques, offering immense potential across various sectors, including industry, education, and humanitarian initiatives, and can improve accessibility, as well as inclusivity in content creation. Despite their promise, generative AI models raise serious ethical concerns and display profound limitations. AI systems integrate data, algorithms of varying complexity, sensors, and actuators – each with inherent values, biases, and unanticipated impacts when introduced into ever-changing socio-technical environments."

Security Concerns and Countermeasures

- Given that the tools being used involve interactions with students' educational data, a robust security and privacy system will be implemented. We did this by having a deep linking connection via Canvas's API. This way, all student information stayed connected to ISU's environment protecting their privacy.
- We need to ensure that the app functions properly at runtime. This has been achieved by via prompt engineering and fine tuning of our model to remove unwanted behavior.

Design Evolution

Initial Design

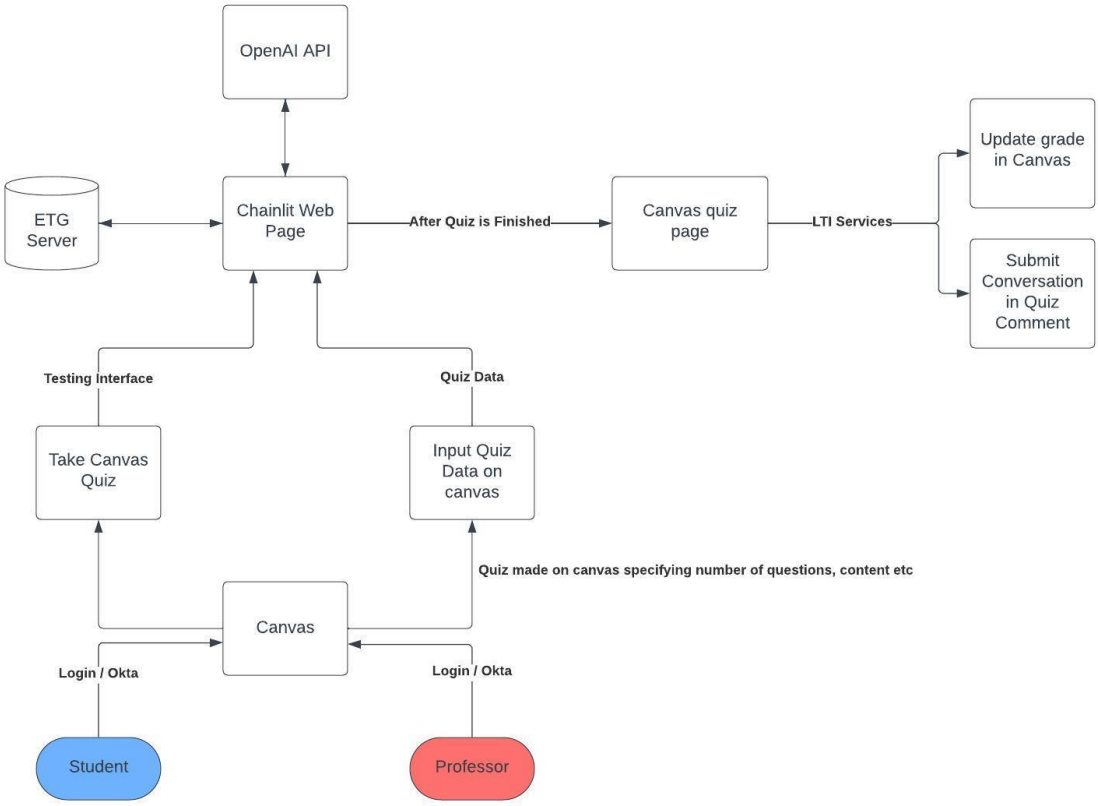

Figure 2

The diagram has two paths that can be taken, one from the student and the other from the professor of the class. We will start with the professor's side of the diagram.

Professor

The professor will login to the Canvas interface through SSO, just as they would normally for any class. Once logging in, the professor inputs the quiz data within Canvas, specifying content, number of questions, etc. After which, the data is transferred to a third-party platform called Chainlit. Here the quiz data enters a pipeline where it is processed by the ETG server and the OpenAI API, which analyzes and enhances the data. The data is then returned to the Chainlit web page. This is when the students can start their personal quiz. Once the student finishes the interactive quiz with ChatGPT, the conversation data is sent through LTI services to update the grade on Canvas and submit the conversation in the quiz comment.

Student

The student will login to Canvas through SSO just as they normally do. They will then navigate to the quiz section of the specific course and click on the quiz that is due. Once the student opens the quiz, they will be taken to our custom interface as an application setup through the Canvas API. As shown by the diagram, the custom interface will be connected straight to a server and the OpenAI API for ChatGPT purposes. The student will take the quiz through our interface. The quiz will conclude for the quiz as specified by the professor, and the data from the quiz will be graded by ChatGPT. The results will be stored in the database for each student. Once the grade is calculated, the interface will send the data back to Canvas and enter it in the specified quiz section, so the student can see.

FUNCTIONALITY

- It will operate similarly to a chatbot, where the user will see the latest response from ChatGPT, type out their response, and be able to hold a conversation with the ChatGPT program.
    - the conversation will consist of the program asking follow-up questions to the student to better understand the reasoning behind the answer given
- Professors will be able to tweak the program to allow more or less "outside the box thinking" and to control how much output the program provides.
- At the end of the conversation, now we are shooting for around 15 minutes, the program will be able to give the student a grade based on the rubric, or guidelines given by the instructor.
- Then once graded, the program should enter the grade into Canvas for the specified quiz or exam.

    - Upon completion, the program will provide a text file into the comments of the Canvas assignment that outlines all the questions and responses from the program and student, respectively.

Design 1 (Design Iteration)

Figure 3

Design Visual and Description

Professor

Even though the underlying infrastructure has changed, the professor view should remain the same from what is was previously. The professor will login to the Canvas interface through SSO, just as they would normally for any class. Once logging in, the professor inputs the quiz data within Canvas, specifying content, number of questions, etc. After which, the data is now transferred to our next.js program. Here the quiz data enters a pipeline where it is processed by the ETG server and the OpenAI API, which formats the data and sends it back to the Canvas in a post request. is when the students can start their personal quiz. Once the student finishes the interactive quiz with ChatGPT, the conversation data is sent through LTI services to update the grade on Canvas and submit the conversation in the quiz comment.

Student

With the addition of next.js, the student route remains remarkably similar to the first design. The student will still login to Canvas through SSO just as they normally do. They will then navigate to the quiz section of the specific course and click on the quiz that is due. Once the student opens the quiz, they will be taken to our next.js program. As shown by the diagram, the next.js program will be connected to OpenAI for ChatGPT purposes. The student will take the quiz through our interface. The quiz will conclude as specified by the professor, and the data from the quiz will be graded by ChatGPT. The results will be stored in the database for each student. Once the grade is calculated, the interface will send the data back to Canvas with a submission POST request and enter it in the specified quiz section, so the student can see it.

Implementation

1. Prompt engineering
    a. Continuous refinement of our prompt to assess students the way we want it to.
2. Fine-Tuning a ChatGPT model
    a. Fine-tuning was done by manually collecting back and forths with ChatGPT, modifying them to the type of questions/responses that were desirable and training ChatGPT off those datasets
3. React/next.js integration
    a. Finish implementation of the user interface.
4. Implementation with Canvas
    a. App is integrated with Canvas and runs effectively (shows up in canvas tab).
5. Refinement the automatic grading process
    a. Follows rubric
    b. Ensure grading accuracy
    c. ChatGPT offers accurate grades based on students answers with feedback to help the student better understand their mistakes and learn from them.
6. Testing and validation.
    a. Tests are successful (Trial runs, etc. students can take quizzes start to finish with no errors).
    b. Opportunity to test with students

Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, power system, or software.

The testing plan should connect the requirements and the design to the adopting test strategy and instruments. In this overarching introduction, an overview of the testing strategy is given. Emphasize any unique challenges to testing your system/design.

The following is our testing plan and process we will be implementing to address acceptance testing, usability testing, security, and performance testing.

Unit Testing

In this section, we wanted to create a comprehensive testing plan that would be able to test not only our individual design components, but our final integrated product. The goal of our testing was to create a system that would ensure each module's desired functionality is achieved individually.

Units

- Web application
    - Quiz generation
        - For interactions with OpenAI's API and our JavaScript environment used for communication with the chat bot we will be able to use PyTest to make sure that the javascript driving the applications connections are robust and accurate for the exam execution.
    - Quiz taking
    - Results and reporting
        - When approaching testing for our web application that is developed with Next.js. We will use Cypress testing and its ability to be able to run both End-to-End and Component Testing to be able to cover each component as well as full End-to-End testing to show full usability testing.
            - Components:
                - Conversation Thread – Both the UI as well as the processing to the API calls.
                - Professor File Upload and Topic/Context Generation.
                - Canvas Embedded Frame integration View from multiple display sizes and device formats.
            - End-To-End:
                - Full testing from professor context and quiz topic proposal to completion of the exam with results being posted on canvas.
- Canvas integration
    - Student integration and view / states of application when opened
- API Calls
    - Open AI API calls
        - Basic testing is needed to make sure that the endpoints are active and reachable when opening a quiz.
            - Chat.create()

- Canvas API calls
  - Basic testing to make sure communication can be made from our application into Canvas's environment.
    - GET, POST requests with needed parameters to create quizzes, and submissions (grades)
    - It is possible to extract text data from these specific parameters
      - Parameters can be assigned to a variable

INTERFACE TESTING

For our interface testing, we have 3 primary interfaces: one for quiz execution, Canvases embedded architecture's view, and the test creating/generation of constraints professor interface.

Interfaces:

- Canvas quiz Screen
- Conversation screen

Overall Interface Testing:

- We will be using a User-Centered Design for creating our interfaces. This will allow us to be focused on making an easy-to-follow self-guidable experience when taking an exam using our software.
  - We will be using Human-Computer Interaction and Web Content Accessibility Guidelines as benchmarks for testing.
  - We will also be using Nielsen Norman Groups 5 Usability Heuristics for User Interface Design and manual testing to make sure each of the following categories are satisfactorily met.
    - Visibility of System status
      - Showing the quiz is live and started via prompts and text box interactions
    - Consistency and standards
      - Use proper conventions that are common in UI development and on other web applications to keep uniformity and a self-navigable environment.
    - Recognition rather than recall
      - Having recognizable components that are easy to navigate without instruction.
    - Aesthetic and minimalistic design
      - Creating a conversational interface as well as a professor quiz creator page that is easy to follow and is devoid of distractions for a proper testing environment.
    - Documentation
      - Having clear and readable documentation for all groups of users who will be interacting with our software

The Interface testing will be in partnership with the component testing to make sure the frames and states that are displayed to the user are accurate and follow the guidelines listed above by using both automated unit testing and manual UI accessibility and usability guidelines.

System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

Check for response time, overall accuracy of responses, as well as things like token generation/overall costs.

Ie. Student asks program to break the rules 100 times, does the program say no every time?

Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure they do not break? Is it driven by requirements? Tools?

Due to the nature of AI development and integration for our application, most of the regression testing will come from the need to be able to increase the accuracy and effectiveness of our AI model used for test generation and automatic grading.

This will come through the constant testing for benchmarking our system on the following key principles:

- Performance goals tied to acceptance testing
- Load testing
    - How many users can we have on our server instance running the site at the same time.
- Endurance testing
    - How does the AI respond to large time gaps or marginally longer and shorter exams.
- Response testing
    - How long does it take each iteration of the model to complete the tasks required of it and how does this change depend on the question type, either practical or theoretical.
- Thruput testing
    - How does it handle quick rapid-fire responses from students.
- Repeat testing
    - Continued testing to achieve the most accurate model for our application.

Acceptance Testing

Our approach to validating the fulfillment of our design requirements involves a comprehensive verification process. To make sure our design meets the desired specifications, we will continuously verify that our results match our tests (listed above). In addition to passing tests, we will get continuous feedback from our clients to make sure their vision corresponds to our vision in the design.

We will also make sure to be able to show statistics on the functional requirements for our application which include:

- Time to complete the next question generation after users' response has been submitted.
- Average cost total per exam dependent upon question length parameter

We will also make sure to be able to document examples of an accurate system by showing examples of completed quizzes and provide feedback as to design issues when it comes to expected vs. actual output instances.

The last requirement for acceptance testing will be in the form of ISU Netiquette compliance. This will be achieved with model management and fine-tuning, which will require manual testing to ensure that the guidelines are being met correctly. www.celt.iastate.edu/wp-content/uploads/2015/09/netiquetteatISU.pdf

Testing Results

Fine-tuning has been shown to improve follow-up question generation. This helped lead us to achieving the goal of having a more adaptable exam environment.

Canvas integration was used in testing the endpoints of the APIs and demonstrated proper interactions via users when taking exams and communicating grades back to Canvas.

Broader Context

Area	Description	Examples
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities) Users (Students) our project deals with AI chatbot interactions between a user and the bot using conversational prompts to be able to figure the next step in the conversation. We must be considerate of mental health and ensure that our conversation follows the correct ethical and ISU netiquette protocols, to ensure students are always treated fairly.	Cursing, negative responses, condescending responses and ethical responses.
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures. Students- this project hopes to assign grades more accurately to students based more on their knowledge of a subject than just a right or wrong answer, and subsequently make life for professors easier by automating the grading and quiz administering process	Development or operation of the solution would violate a profession's code of ethics, implementation of the solution would require an undesired change in community practices By focusing on understanding rather than correctness, the project seeks to support students who may have different ways of approaching and expressing their knowledge.
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement. Our project would most likely increase energy usage on campus as	Increasing/decreasing energy usage from nonrenewable sources, increasing/decreasing usage/production of non-recyclable materials Shifting from paper-based exams to online quizzes reduces the demand for paper, potentially lowering

	every student would be taking quizzes online through OpenAI instead of possibly on paper in a classroom. Additionally, professors would need to access the same system to input quiz information, further increasing energy usage.	deforestation rates. Carbon foot print of Chat GPT is Cited at 8.4 tons of carbon dioxide per year which is about the same amount as 2 people contribute in a total year. Water consumption a conversation of 20-50 questions consumes 500 ml of water and training GPT-3 required a staggering 700,301 liters of water.
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups. Quizzes should cost less than $2 per quiz per person, based on preliminary estimation using token costs.	Product needs to remain affordable for target users, product creates or diminishes opportunities for economic advancement, high development cost creates risk for organization Consider a university with thousands of students taking exams regularly. The shift to online quizzes not only eliminates the need for paper and ink but also reduces administrative costs associated with manual grading. The institution could realize substantial cost savings in the long run.

Closing Material

Conclusion

Our goal is to create an interface, useable to both students and professors. This will be achieved by using prompt engineering, JavaScript, react, node.js etc. So far, we have done research on prompt engineering, and identified several prompt patterns that will be helpful in eliciting the response we want from ChatGPT. We have been able to communicate with ChatGPT using the OpenAI API from JavaScript and have full round-trip communication from canvas to ChatGPT. The best plan of action going forward is to further test and refine our current prototype, and to find a more convenient way to access the application on canvas. Future designs may look to further optimize token usage – for example, we are looking into using a hybrid fine-tuning / prompt engineering approach. The reason for this will be to try and reduce the sheer size of the prompt that we currently must pass the language model to get the desired results.

Final Progress

Currently, our project is set up in a way that a student could take an exam through Canvas. Once a student clicks on their link for the quiz, they will be redirected to that quiz. The student will type "Ready" to begin the quiz. The back-and-forth conversation will be held, and once the quiz ends the grade will be posted to Canvas. The backend is all tested and set up. The conversation with the student will be posted to the database. The integration with Canvas is also working as expected, where a student can get to our user interface through Canvas.

Value

This project represents an emerging field of educational technology at Iowa State University. It makes the traditional exam-taking experience into a dynamic and interactive process. Finally, it streamlines the administrative workload for professors while generating mass personalized feedback. The project holds value for the student, teacher, and the university.

Appendix 1 – Operation Manual

The navigation from the student view is straight forward. Students will login to Canvas, and head to their class. In our case, the the student heads to their assignment called “SDMAY24-36”. Within this assignment, our created user interface will show up. You can see below; the quizzes will appear for the student to click on.

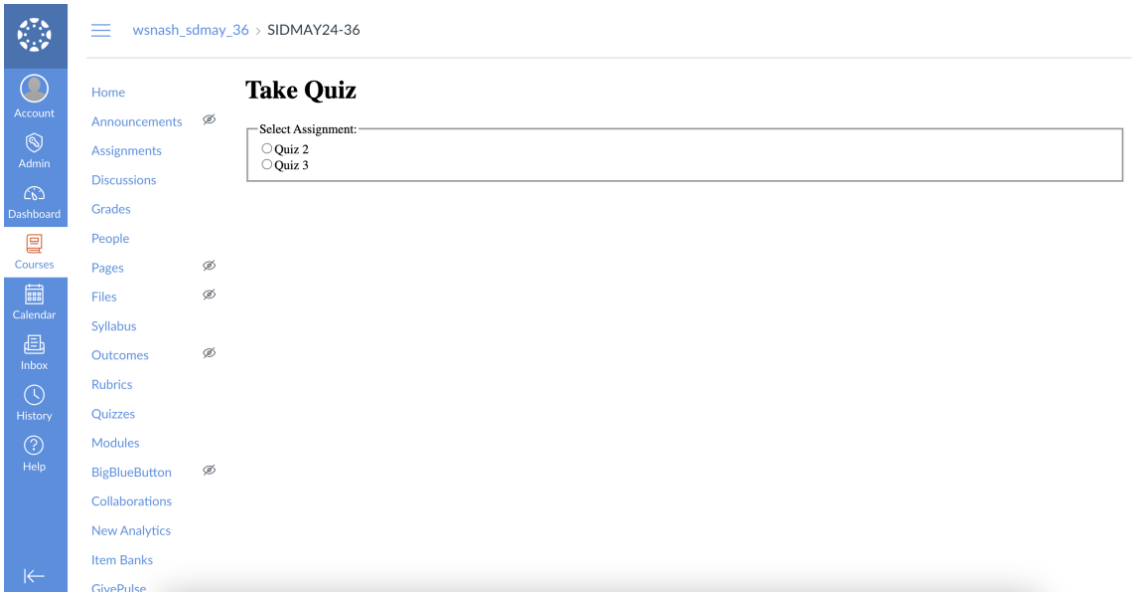

Figure 4

Once the student navigates to their quiz, they will be taken straight to the quiz interface. This is where the student can type “Ready” to start their quiz.

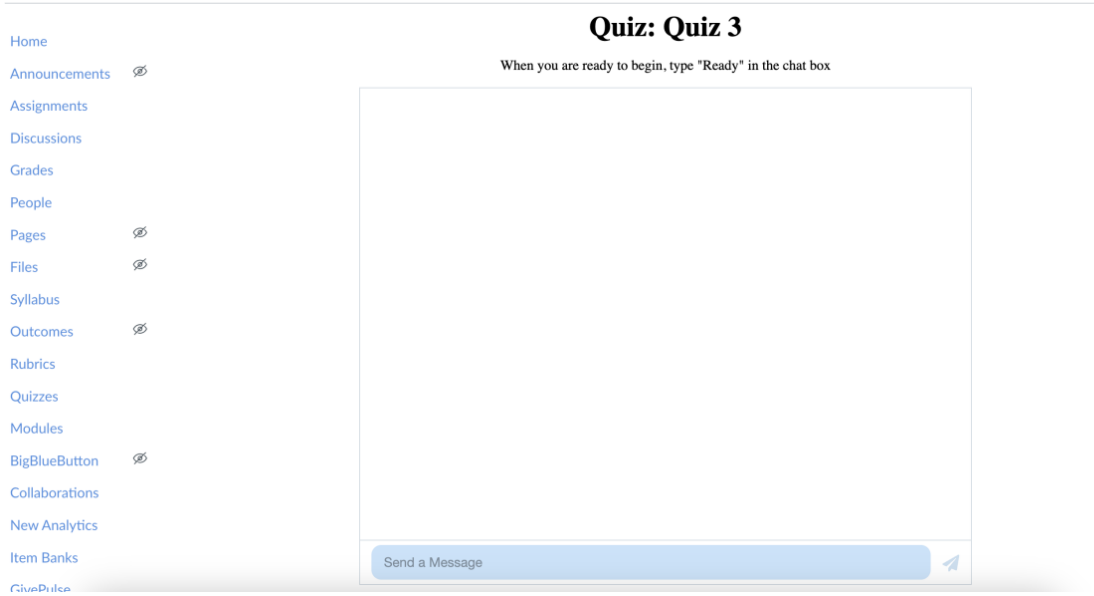

Figure 5

Once the quiz is done, the grade and conversation will be posted to the connected Canvas quiz. All the student work is done, and the backend handles the grade posting and comment submission.

Appendix 2 – Initial Version of Design

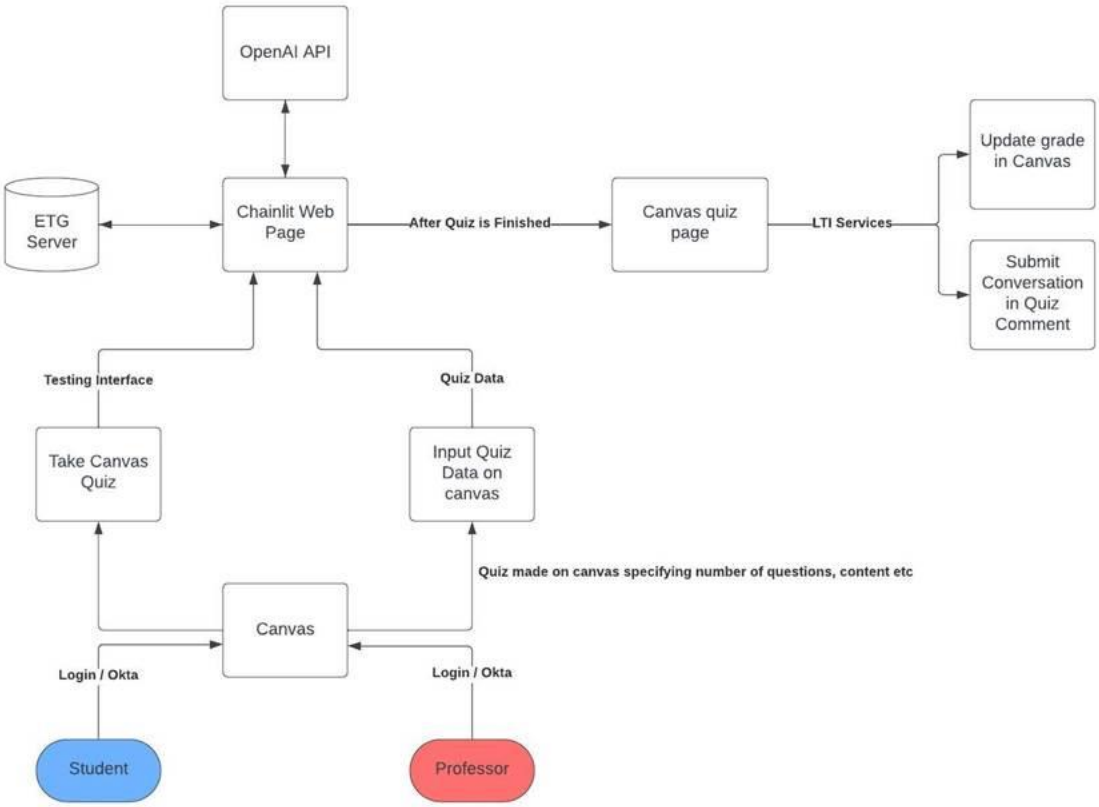

Figure 6

The diagram has two paths that can be taken, one from the student and the other from the professor of the class. We will start with the professor's side of the diagram.

Professor

The professor will login to the Canvas interface through SSO, just as they would normally for any class. Once logging in, the professor inputs the quiz data within Canvas, specifying content, number of questions, etc. After which, the data is transferred to a third-party platform called Chainlit. Here the quiz data enters a pipeline where it is processed by the ETG server and the OpenAI API, which analyzes and enhances the data. The data is then returned to the Chainlit web

page. This is when the students can start their personal quiz. Once the student finishes the interactive quiz with ChatGPT, the conversation data is sent through LTI services to update the grade on Canvas and submit the conversation in the quiz comment.

Student

The student will login to Canvas through SSO just as they normally do. They will then navigate to the quiz section of the specific course and click on the quiz that is due. Once the student opens the quiz, they will be taken to our custom interface as an application setup through the Canvas API. As shown by the diagram, the custom interface will be connected straight to a server and the OpenAI API for ChatGPT purposes. The student will take the quiz through our interface. The quiz will conclude for the quiz as specified by the professor, and the data from the quiz will be graded by ChatGPT. The results will be stored in the database for each student. Once the grade is calculated, the interface will send the data back to Canvas and enter it in the specified quiz section, so the student is able to see.

Appendix 3 – Code

All code can be found in our Iowa Sate managed Gitlab instance: https://git.ece.iastate.edu/sd/sdmay24-36. Our server is found at: https://sdmay24-36.sd.ece.iastate.edu:3000/ (with Canvas authorization only).