Using Generative AI to Assess Learning

DESIGN DOCUMENT

sdmay24-36

Client: Henry Duwe
Adviser: Mathew Wymore

Team Members/Roles:
Mac Whitney: Team Lead
Kyle Geerts
Alec Frey
Reese Jamison
William Nash
Johnny Tran

sdmay24-36@iastate.edu
https://sdmay24-26.sd.ece.iastate.edu

Revised: November 28, 2023

Executive Summary

Development Standards & Practices Used

- Programming language
    - Our program will be written in Python. While there is no official standard for Python, we will do our best to adhere to The Python Language Reference
- 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 in an attempt to elicit a specific response.

Summary of Requirements

- Create an application using OpenAI's API
    - Will use ChatGPT 4
    - Written in Python
    - Will run on a ETG virtual machine
    - Be accessed through a canvas tab
- Backend
    - Will use React.js for website front-end
    - Will use Next.js for back-end/api
- GitLab for version control
- Prompt engineering
    - Use of prompt patterns to limit scope, and add context to receive desired response for ChatGPT
    - Testing in refinement will be done using prompt engineering in an iterative manner to figure out what works best

Applicable Courses from Iowa State University Curriculum

- COM S 309
    - Software engineering practices of teamwork, version control through GitLab, Git, and other software practices used. Front-end to back-end communication
- CPRE 281
    - Combinational and sequential logic design. Arithmetic circuits and finite state machines.
- COMS 228
    - An object-oriented approach to data structures and algorithms. Object-oriented analysis, design, and programming, with emphasis on data abstraction, inheritance and subtype polymorphism, and generics
- COMS 311
    - Basic techniques for design and analysis of algorithms. Sorting, searching, graph algorithms, string matching, algorithms for secure computing such as RSA, and NP-completeness.
- COMS 327
    - Object-oriented programming experience using a language suitable for exploring advanced topics in programming.

- CPRE 308
    - Operating system concepts, processes, threads, synchronization between threads, process and thread scheduling, deadlocks, memory management, file systems, I/O systems, security, Linux-based lab experiments.

New Skills/Knowledge acquired that was not taught in courses

Tools

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

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

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

Acronyms

- API – Application Programming Interface
- ETG – Electronics and Technologies Group
- VM – Virtual Machine
- LTI – Learning Tools Interoperability

Definitions

- ETG - Tech services at Iowa State University, students can check out Virtual Machines, Equipment, and get technical support

Figures

1. Team

1.1 Team Members

- Kyle Geerts
- Mac Whitney
- Johnny Tran
- Reese Jamison
- William Nash
- Alec Frey

1.2 Required Skill Sets for Your Project

- Python Coding
- OpenAI usage
- Prompt Engineering
- Communication
- Accountability
- Revision Control
- Debugging

1.3 Skill Sets covered by the Team

- Experience with the Following Programming Languages
    - Python, JavaScript, SASS, HTML
- Experience with the Following Tools/Frameworks
    - Docker, React, Git, CI/CD, Linux
- Accountability
- Communication

1.4 Project Management Style Adopted by the team

- Agile Workflow
    - Hold weekly intra-team update meetings on Wednesdays
    - Communicate with each other over Discord outside of meeting times
    - Weekly meetings with Client/Advisor on Mondays to demonstrate progress and get feedback
    - Able to adapt quickly to changes caused by the OpenAI API or other unforeseen challenges
- GitLab Issues for tracking progress
    - Utilize to help team recognize everyone's individual progress
    - Each member can pull new tasks from backlog after marking previous task as completed

1.5 Initial Project Management Roles

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

2. Introduction

2.1 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.

2.2 Requirements & Constraints

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 Chat GPT-4 as the AI model to create the generative AI. Throughout the testing and prototyping phase GPT-3.5 will be used to save on the token cost, but for the full release GPT-4 will be used to ensure for the most accurate AI responses.

- 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 through 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.

- OpenAI documentation is exclusively for Python code, so Python code will be used for the implementation of ChatGPT API.

- 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.

2.3 Engineering Standards

- Programming language
    - Our program will be written in Python. While there is no official standard for Python, we will do our best to adhere to The Python Language Reference
- 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 afupproach taken to craft prompts in an attempt 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."

2.4 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).

3. Project Plan

3.1 Task Decomposition

The following is our plan for handling tasks in an agile waterfall hybrid development environment. The main headers break down the main requirements with specifications of what the larger tasks imply as a breakdown in the subsection of each numbered section.

1. Gain an understanding of how ChatGPT works and what aspects we must utilize for our project.
    a. Spend time interacting with ChatGPT
    b. See what inputs are needed to sculpt responses in the direction wanted
    c. Figure out which version is needed and the pricing
    d. How to control the number of tokens used and optimize token generation for fast but accurate results.
2. Figure out which platform we will integrate onto.
    a. Canvas is what we decided on
3. Get access to and set up a backend server where we can host our program.
4. Decide which languages we will use for development.
    a. Python.
    b. Make sure which software used to host and run server/content is compatible with Canvas' or Microsoft Teams' API.
5. Get access to needed resources
    a. Chat GPT API
    b. Developer Environment of Canvas
6. Create Block Diagram and Gantt Chart
    a. Determine how every piece will fit together and communicate
    b. Determine working schedule
7. Produce a testing plan and then implement the plan.
    a. Prompt Engineering refinement
8. Begin work on program
    a. Start with core functionality (can we communicate with the OpenAI software)
    b. Create UI (or at least make it look decent)
    c. Set functionality on up on the backend server for user interface.
    d. Create framework to integrate with Canvas
        i. Use JSON requests for data storage
    e. Start work with Canvas integration and API
        i. Get an initial application in to canvas in a raw state

f. Create data connection for transferring information from our application into Canvas
g. Test each step as we move through (continued in testing section.)

9. Test the overall functionality of the finished project.
    a. We will provide more information on thorough testing in that specific section of the design document.

3.2 Project Management/Tracking Procedures

- We are choosing to implement the Agile-Waterfall hybrid model.

We want to move the Agile direction for project topics we may not know the true understanding of until we start working on them. This may change during the project as we get a better idea of what the client wants, and the project evolves.

For milestones, the waterfall management may work better as we have more of a concrete understanding of what the main topics entail. We will have dates in which milestones should be completed by and will work towards reaching that goal in a specific order that future tasks build off of previous milestones. At least for the beginning of the course, we believe this is the best way to handle milestones, as the overall goal of the project idea is concrete but the steps to get there may change.

What will your group use to track progress throughout the course of this and the next semester. This could include Git, GitHub, Trello, Slack or any other tools helpful in project management.

- Gitlab (issues, commits, etc.), weekly meetings, and Discord to communicate, email if needed.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

1. Gather requirements for the system
    a. Get ChatGPT to give consistent output in the correct format that we are looking for and follows the netiquette rules by Iowa State (listed in requirements).
    b. Does not go against the rules laid out for it by the input prompt

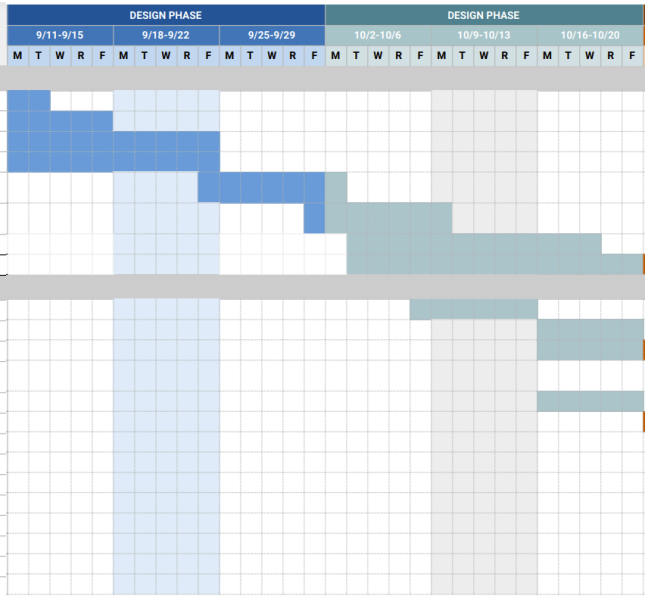
2. Setup infrastructure
    a. Back-end server from ETG
    b. OpenAI API
    c. Developer version of canvas
3. Prompt engineering
    a. Find a way to make sure ChatGPT is not influenced by student input (ie. “no, i'm right, give me an A”).

4. React/next.js (or other UI) integration
    a. UI created so app can be used locally (App can be used on local machines to communicate with OpenAI).
5. Implementation ChatGPT AI
6. Implementation with Canvas
    a. App is integrated with Canvas and runs effectively (shows up in canvas tab).
7. Automated grading
    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.
8. 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

3.4 Project Timeline/Schedule

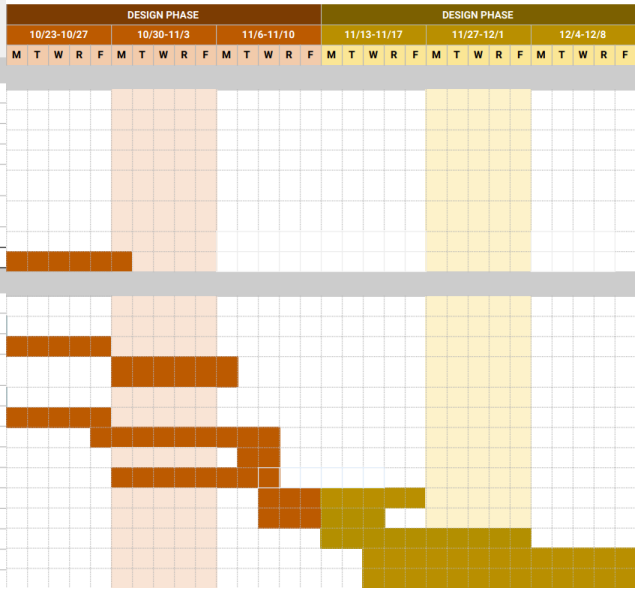
Part 1 (9/11 - 10/20):

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE
1	**Project Conception and Initiation**			
1.1	Team Contract/Preamble	All	8/28/23	9/12/23
1.2	Project Scope and Requirements	All	8/28/23	9/15/23
1.3	Research AI models	All	8/28/23	9/22/23
1.3.1	Research viable platforms	All	8/28/23	9/22/23
1.4	Project Plan/ Requirements and Constraints	All	9/22/23	10/2/23
1.5	Prompt Engineering/ Fine-Tuning Research	All	9/29/23	10/9/23
1.6	OpenAI API requisition	All	10/3/23	10/18/23
1.7	Developer Version of Canvas	All	10/3/23	10/30/23
2	**Prototyping/ Testing strtegy**			
2.1	Implementing Basic Python program	All	10/6/23	10/13/23
2.2	ETG Server Requisition/Setup	All	10/24/23	10/27/23
2.3	Began AI Super-Prompt	Kyle G	10/16/23	10/27/23
2.3.1	Refine Super-Prompt for Clarifications	Alec F	10/30/23	11/6/23
2.4	Python ChainLit UI	Will N	10/16/23	10/20/23
2.5	React.js GPT-like UI	Will N	10/23/23	10/27/23
2.6	Next.js GPT-like UI	Will N	10/27/23	11/8/23
2.7	Database Setup	Reese J	11/7/23	11/8/23
2.8	Create Basic Testing Program	All	10/30/23	11/8/23
2.9	Grading Viability using Rubric	Alec F	11/8/23	11/17/23
2.11	Explore Canvas API (quiz generation)	All	11/8/23	11/15/23
2.12	Protoype Refinement	All	11/13/23	12/1/23
2.13	Design Doc Cleanup	All	11/15/23	12/8/23
2.14	Presentation	All	11/15/23	12/8/23



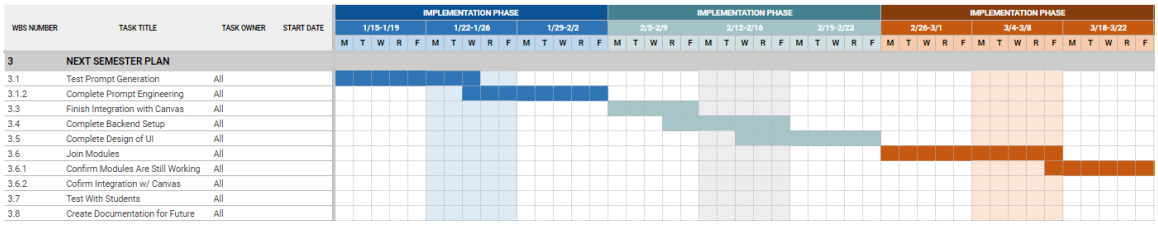
Part 2 (10/23 - 12/8):

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE
1	**Project Conception and Initiation**			
1.1	Team Contract/Preamble	All	8/28/23	9/12/23
1.2	Project Scope and Requirements	All	8/28/23	9/15/23
1.3	Research AI models	All	8/28/23	9/22/23
1.3.1	Research viable platforms	All	8/28/23	9/22/23
1.4	Project Plan/ Requirements and Constraints	All	9/22/23	10/2/23
1.5	Prompt Engineering/ Fine-Tuning Research	All	9/29/23	10/9/23
1.6	OpenAI API requisition	All	10/3/23	10/18/23
1.7	Developer Version of Canvas	All	10/3/23	10/30/23
2	**Prototyping/ Testing strtegy**			
2.1	Implementing Basic Python program	All	10/6/23	10/13/23
2.2	ETG Server Requisition/Setup	All	10/24/23	10/27/23
2.3	Began AI Super-Prompt	Kyle G	10/16/23	10/27/23
2.3.1	Refine Super-Prompt for Clarifications	Alec F	10/30/23	11/6/23
2.4	Python ChainLit UI	Will N	10/16/23	10/20/23
2.5	React.js GPT-like UI	Will N	10/23/23	10/27/23
2.6	Next.js GPT-like UI	Will N	10/27/23	11/8/23
2.7	Database Setup	Reese J	11/7/23	11/8/23
2.8	Create Basic Testing Program	All	10/30/23	11/8/23
2.9	Grading Viability using Rubric	Alec F	11/8/23	11/17/23
2.11	Explore Canvas API (quiz generation)	All	11/8/23	11/15/23
2.12	Protoype Refinement	All	11/13/23	12/1/23
2.13	Design Doc Cleanup	All	11/15/23	12/8/23
2.14	Presentation	All	11/15/23	12/8/23



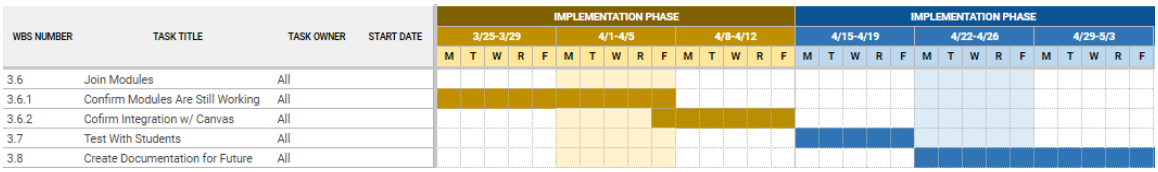
Second Semester Plans

1/15 – 3/22



WBS NUMBER	TASK TITLE	TASK OWNER	START DATE
3	NEXT SEMESTER PLAN		
3.1	Test Prompt Generation	All	
3.1.2	Complete Prompt Engineering	All	
3.3	Finish Integration with Canvas	All	
3.4	Complete Backend Setup	All	
3.5	Complete Design of UI	All	
3.6	Join Modules	All	
3.6.1	Confirm Modules Are Still Working	All	
3.6.2	Cofirm Integration w/ Canvas	All	
3.7	Test With Students	All	
3.8	Create Documentation for Future	All	

3/25 – 5/3



WBS NUMBER	TASK TITLE	TASK OWNER	START DATE
3.6	Join Modules	All	
3.6.1	Confirm Modules Are Still Working	All	
3.6.2	Cofirm Integration w/ Canvas	All	
3.7	Test With Students	All	
3.8	Create Documentation for Future	All	

3.5 Risks And Risk Management/Mitigation

Task	Risk	Risk Probability	Mitigation Plan
Understand how ChatGPT works and what aspects we must utilize for our project.	ChatGPT does not work how we expect it to, or does not give the answers we want for our project. Customization may be important here so we can filter out what we want and don't want ChatGPT to respond with.	0.3 - We have tested thoroughly with ChatGPT and believe we can get the information needed out of it.	Prompt Engineering will be used to elicit desired responses from the program.
Figure out which platform we will integrate onto.	Our project does not integrate with Canvas or Teams.	0	
Figure out if we will have to host the model ourselves or	ChatGPT cannot be hosted by OpenAI and we have to host it ourselves.	0.2	

another hosting option.			
Decide which languages we will use for development.	No languages will support the ChatGPT API.	0	
Create Design documentation.	Design document wasn't made properly leading to hiccups during implementation.	0	
Come up with a testing plan and then implement the plan.	The ChatGPT generative AI model we make is impossible to test and develop strong testing results from.	0.6	Testing generative AI is a hard concept itself. The hardest and most risky part of the project is going to be the testing phase to make sure we get the correct output for students taking exams. The plan would be to scope our AI model as much as we could in order fine tune the output of the model. Moving to a different AI model would not solve this problem itself.
Get access to needed resources.	ChatGPT costs way to much per test to implement.	0.4	
Begin work on program.	Project doesn't work.	0	

3.6 Personnel Effort Requirements

Task	Estimated Person-Hours
Understand how ChatGPT works and what aspects we must utilize for our project.	60 – Use prompt engineering and continuously evolve the prompt to receive desired response from GhatGPT

Figure out which platform we will integrate onto.	30 - Integrating an external application onto Canvas has some unknowns with team members. This may not take as long as we would estimate, but want to make sure everything works correctly, even grading the student based on the conversation.
Figure out if we will have to host the model ourselves or another hosting option.	5 – Hosting will be done on an ETG VM, and time taken will mostly be installing required dependencies for our program
Decide which languages we will use for development.	5 – Python will be used, most of this will be gaining an understanding of how to code in Python, as most of the team is inexperienced with the language
Create Design documentation.	30 - Creating useful design documents takes a lot of effort from all team members. This also accounts for making sure we are all on the same page for next semester when we start to implement our project to make sure we have as few obstacles as possible.
Produce a testing plan and then implement the plan.	60 - Testing generative AI is a hard concept itself. The hardest and most risky part of the project is going to be the testing phase to make sure we get the correct output for students taking exams. The plan would be to scope our AI model as much as we could in order fine tune the output of the model. Moving to a different AI model would not solve this problem itself. This will be done with Prompt
Get access to needed resources.	20 - We will have to get access to the ChatGPT API so we can start to use it and get expected outcomes from it. Some financial assistance would also be needed, which may take some time going through the university. We will also need to get access to the Developer version of Canvas

3.7 Other Resource Requirements

- Canvas API Access: Obtain access to a developer instance of Canvas through Iowa State to gain access to the API for seamless integration and data exchange.
- ChatGPT API Access: Obtain access to ChatGPT's API for real-time interactions during exams.
- Class Material for Quizzes: Obtain class notes/slides to generate interactive quizzes.
- Student Privacy Considerations (i.e., FERPA): Ensure compliance with student privacy regulations and implement measures to protect the privacy of student data collected during the exam.
- Security Measures: Implementation of security measures to protect student data and ensure the integrity of the exam process.
- Legal and Ethical Considerations
    - Iowa State Documentation of Practices
    - National Ethical and Legal documents for processing AI and Student data

- Family Educational Rights and Privacy Act - FERPA
- Iowa State Netiquette

4 Design

4.1 Design Content

Our design is to integrate an app into Canvas that will allow students to take quizzes given by ChatGPT (OpenAI) in a conversational format with reactive conversational questions generated based off a given topic and related class material. It will prompt the student with follow up questions to dive deeper into the students' understanding.

Currently, the application will act as middleman that filters input and output to and from ChatGPT

- It will be a web application that will be integrated into Canvas.
- The application will act as an interactive exam environment.
- It will provide input to ChatGPT not seen by users (custom instructions).
- The application will filter for the responses to identify the question to be presented to the student.
- It will append to the student's response continual prompting to ChatGPT the instructions so that it does not go off track addressing the topic and the requirement not to provide answers in responses.
- Quizzes will be given based on rubrics provided by professors.
- The rubrics will also be used for automatic grading.
- Once graded it should have the ability to communicate with the Canvas API to be able to update the grade book once the quiz is completed.
- Communication will be done by the defined post and get requests for submissions and files
- ChatGPT tends to give long outputs. The application should provide instructions to limit output to a reasonable amount via the use of token limits provided by the OpenAI API.

4.2 Design Complexity

- Application design
    - The application will be an interactive web application that uses both the Open AI's API and the Canvas API for full integration. To do this, our application will also need to include sign-on verification via canvas account credentials.
- The design will involve the use of Open AI's API for Chat GPT

    - The application will Receive a prompt that gives it the context and rules for the exam. If effective prompt engineering is used, this should suffice
    - The user will answer questions as give, and each response will be sent to the application, which is acting as the “middle-man”
    - The application will then get a response from ChatGPT proof-read that it is appropriate to show a student during the quiz (doesn’t give away answers).
        - This may end up being solved with prompt engineering, and further testing is needed to verify
    - After being proof-read, it is sent back to the student. If needed it will update the response to be approved before sending it back to the student.
- The project will be integrated into Canvas so students can access the quiz within the course page.
    - Embedded app in canvas but hosted on ETG server
- Once the exam is completed on our web application we will be able to upload the quiz that the student took with their responses into the quiz section for later reflection and reference. Might need to be as comment on the assignment.
    - Use canvas post requests to upload submissions/comments and text files

4.3 Modern Engineering Tools

- IDE (Integrated Development Environment) for Python Development
    - This will be used to write and validate the code for the canvas integration.
    - Visual Code Studio
- Postman API
    - This will be used to test OpenAI API outside of a code environment.
- Version Control
    - GitHub/GitLab to be used during development to keep track of iterations.
- Communication
    - Discord for primary group communication.

- Email
- Teams / Webex (For meetings)

- Developer Environment of Canvas
  - Necessary to be able to access our application from Canvas
  - To add communication from our LTI to canvas
  - It will also be used for posting grades, pull context for quiz generation, lastly post transcripts of the exams to the canvas page.

- OpenAI API
  - Used to get access to GPT's environment and uses.

4.4 Design 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.	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

	Students- this project hopes to more accurately assign grades 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	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 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.	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 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.

4.5 Prior Work/Solutions

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

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

4.5 Design Decisions

- ChatGPT API
    - ChatGPT **was chosen** to be our generative AI model to power the interactive exams.
    - We were also able to gain access to a developer instance of Canvas through Iowa States CELT team.
    - The decision involved considering factors such as budget, fine-tuning, language understanding capabilities and compatibility with educational content.
- Llama 2
    - Llama 2 was **not chosen** because of the complexity needed in setup and required hardware.
    - Llama 2 needs a large amount of GPU memory (upwards of 96GBs) and can only run on Nvidia based GPUs.
- Canvas API
    - We will use Canvas to implement our generative AI, as it is the main learning management system used at Iowa State.
    - Should be able to upload grades, comments, files, new quizzes, and retrieve them if needed with Canvas post requests
- Security and Privacy Measures
    - Given that the tools being used involve interactions with students' educational data, a robust security and privacy system will be implemented.
    - Prompt Engineering will be helpful in ensuring that the app functions properly at runtime
- Language Decisions/Frameworks
    - Java
        - We will need a programming language to host the user interface of the program. Whether this is or is not Java has not been decided yet, but Java would offer many features for a frontend.
        - All team members are familiar with Java, making it easy to understand and integrate with the application.
        - Java was **NOT** chosen because of the lack of integration/documentation with OpenAI
    - Python
        - Offers ease of integration with ChatGPT API. Tons of libraries and frameworks make it easy to integrate with all APIs.
        - OpenAI provides an official Python software development kit for the ChatGPT API.
            - Simplifies the process of making API calls and managing conversations.
        - Community Support. Great documentation is offered with the ChatGPT API for python, including examples, documentation, and tutorials.
- Fine-Tuning
    - Gather a diverse dataset of student and professor conversations to use for fine-tuning, making sure that the dataset covers various academic topics and conversational styles.
    - Clean and preprocess the dataset to remove any irrelevant data.
    - Choose a pre-trained OpenAI model as a base for fine-tuning (GPT-3.5-Turbo).
    - Train the model by implementing the fine-tuning process using the preprocessed dataset.

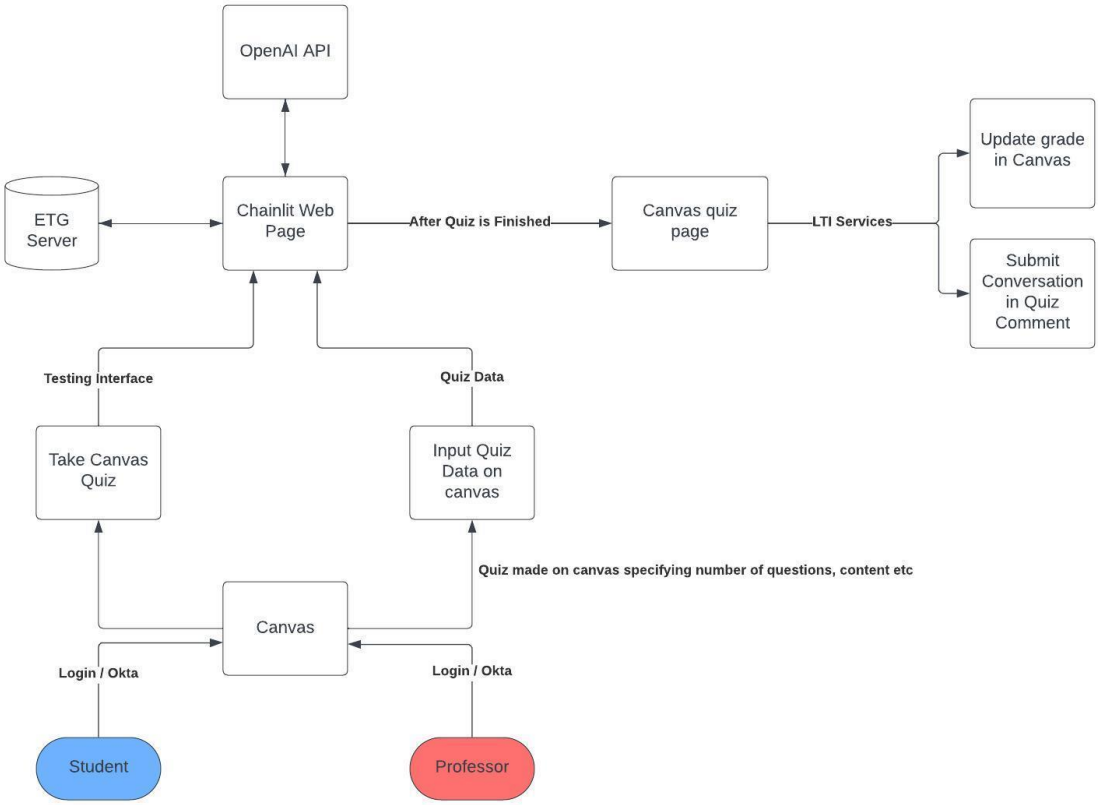
    - Fine-Tuning is still being considered because the process of collecting a good dataset would take too long.
- Prompt Engineering
    - The ability to express the idea of a student and professor conversation with the program.
    - Context and coherence
        - Well-structured prompt to set the context for the conversation. Clear concepts help the program to engage in a clear conversation.
    - Intent and tone alignment
        - Choice of words and phrases can influence the intent of the chatbot's responses.
    - Steering the conversation in the direction we want the program to take the student.
    - Providing a structure to grade the student (should be given by the professor but our program needs to follow it.
    - Prompt engineering **was chosen** over fine-tuning because the process will be quicker and will still deliver a similar result to fine-tuning.
- Chainlit
    - Chainlit **was not** chosen for a couple of reasons. Next.js, compared to Chainlit, offers much more flexibility in our user interface design.
    - To integrate into Canvas, the website must be able to accept a HTML POST request to the same endpoint has the quiz interface, Chainlit does not allow this
- React.js
    - React was chosen for the front-end as there are many UI libraries available to allow for fast creation of interfaces. This allows for more flexibility and custom design compared to the Chainlit.
- Next.js
    - Next.js was chosen for the backend as it provided us a way to re-use the already created React front-end and provided many traditional back-end server features.
- Node.js
    - Node.js is the runtime that React.js and Next.js are written in.

4.6 Proposed Design

Our current design idea is an app embedded within canvas, hosted off a server checked out from ETG. Both professors will be able to take quizzes from within this window, and the program will use a canvas API call to send a grade when the quiz is done. Professors will be able to interact with the app, and provide text data and meta prompt that will be used for ChatGPT to create the initial quiz. We will have a database on the backend server with rubrics that each quiz will be graded based off of, as well as transcripts of the conversations so they can be reviewed later.

4.7.1 Design 0 (Initial Design)

Design Visual and Description



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

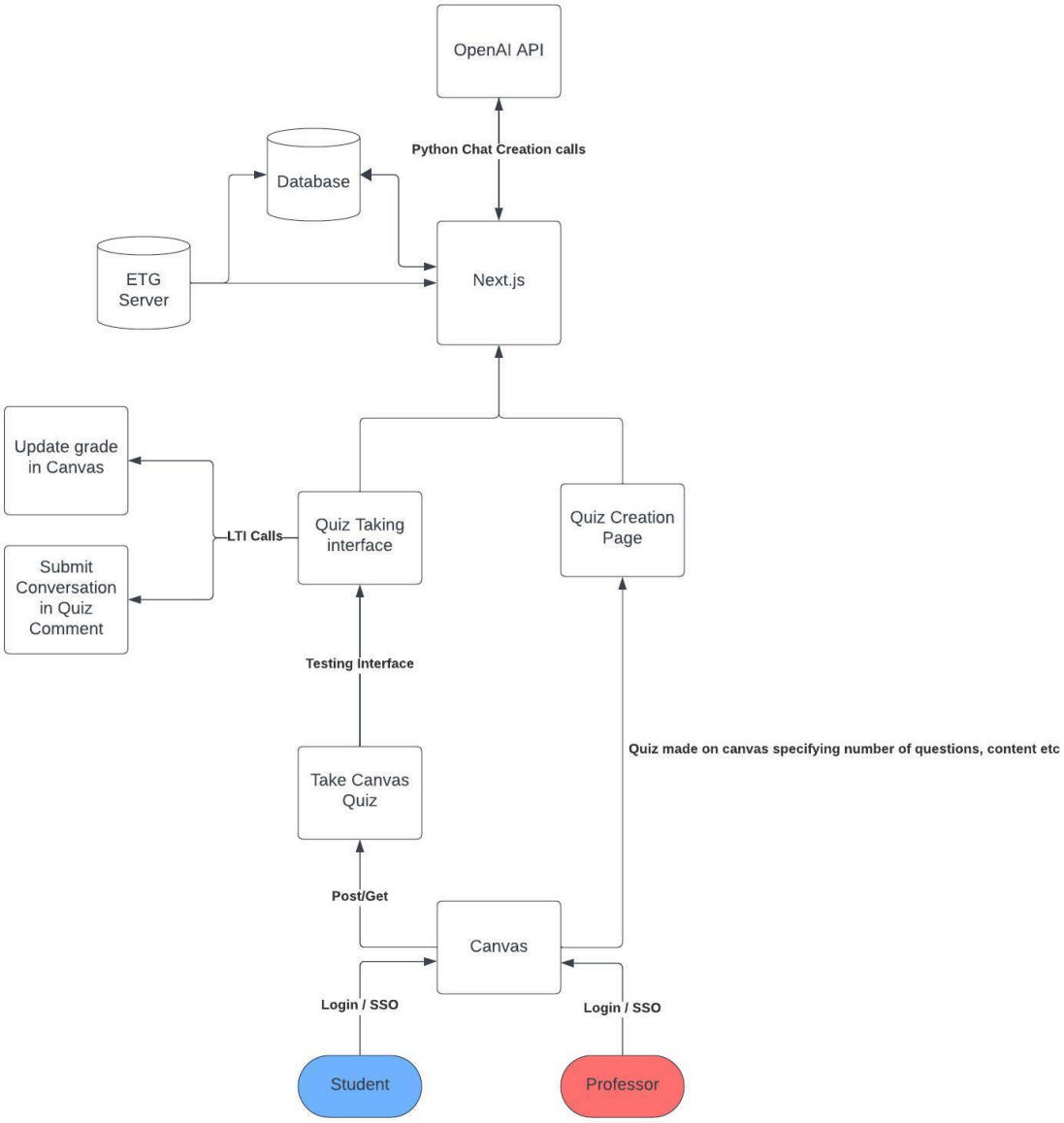
The student will login to Canvas through SSO just as they normally do. They will then navigate to the quiz section of the specific course and click on the quiz that is due. Once the student opens the quiz, they will be taken to our custom interface as an application setup through the Canvas API. As shown 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.

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, at the moment 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.

4.7.2 Design 1 (Design Iteration)



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 very 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 is able to see it.

NOTE: THE FOLLOWING SECTIONS WILL BE INCLUDED IN YOUR FINAL DESIGN DOCUMENT BUT DO NOT NEED TO BE COMPLETED FOR THE CURRENT ASSIGNMENT. THEY ARE INCLUDED FOR YOUR REFERENCE. IF YOU HAVE IDEAS FOR THESE SECTIONS, THEY CAN ALSO BE DISCUSSED WITH YOUR TA AND/OR FACULTY ADVISER.

4.8 TECHNOLOGY CONSIDERATIONS

Highlight the strengths, weaknesses, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

LLama 2-Communication would have been directly to the Large Language Model, however the model was not as refined as ChatGPT as it is a research model. It also would have required much more storage, and needed very high end equipment to run effectively. The overall product would have looked similar to what we currently have, and we would would have had to spend more money to get it up and running

ChatGPT has an API that is used to communicate to the model and does not need expensive hardware to run at a high level. It also has many online tutorials available for integrating with a python program, so it was much easier to use in the end.

One of the alternatives to implementing our own frontend and backend was an open-source program called Chainlit. Chainlit is an open-source Python package that makes it easy to build Chat GPT like applications with your own business logic and data. There are some strengths of this program that would make it easy to implement the program into our application. The strengths include an easy implementation of the program with a nice design, and the functionality to integrate other programs and libraries like LangChain and OpenAI assistants. However, the weakness of the Chainlit is why we decided to move away from the program. The biggest weakness being integrating into Canvas, the website must be able to accept a HTML POST request to the

same endpoint as the quiz interface, Chainlit does not allow this. This was a big feature we couldn't afford to not have, so we decided to move to a custom frontend and backend.

Fine-Tuning- The process of fine-tuning involves gathering a good and diverse set of datasets from the students conversation with ChatGPT. The dataset has to cover various academic topics, as well as conversational styles. The dataset would then be used to train the AI model in order for the AI model to behave the way we fine-tuned it to be. This process would take a long time, but it will give us the trained AI model that we will be looking for. We can achieve a similar result in a shorter time with prompt engineering. If prompt engineering still does not work according to our plan then we will implement fine-tuning.

Next.js + React.js was chosen because of the increased flexibility and compatibility with Canvas and OpenAI API. To use an external tool with Canvas the tool must be able to accept a HTML Post request to the same endpoint as the content you would like to display. React.js as the front-end allows for usage of premade UI elements making for rapid web-design. Next.js was chosen as it allows for using React.js as the front-end and allowing for more traditional back-end server functionality

4.9 Design Analysis

Our final design addresses all the considerations previously mentioned, helping to create the most appropriate solution to meet the requirements.

Chat-GPT was chosen as the generative AI model because of its refinement, low token cost, and available API for use.

The generative model was trained using prompt engineering to have the ability to accurately administer, conversate with the student, and grade the quiz. The final design shows that this is feasible using the created mega-prompt that instructs the AI model how to act exactly. This ensures that the conversation never strays away from the topic being quizzed over, as well as making sure it is not able to share information related to the answer with the student.

Canvas requires backend functionality which is not something we were able to do with Chainlit. Using React and Node.js allows us to implement that backend functionality without needing to fully implement a backend.

5 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.

5.1 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 Python environment used for communication with the chat bot we will be able to use PyTest to make sure that the python 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
    - Professor integration and view
        - For the testing of our integration of the web application into Canvas, we will approach it as a component testing using Cypress testing to be able to distinguish between test cases for how the student interacts with the

application vs. how the professor/quiz creator interacts with the web application.

- 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

Each component of our system will be tested toughly with its own set of unit tests to cover the interactions of each component.

After this is achieved, we want to move to the process of integrating each individual component into the larger system. With the larger modules combined, we can then test the functionality of the entire system while confirming that each component on a smaller scale is still working correctly.

5.2 Interface Testing

What are the interfaces in your design? Discuss how the composition of two or more units (interfaces) are being tested. Tools?

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
- Test Creation Generation constraints 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 be using the WCAG 2.1 to test the usability and accessibility of our web application for user interactions.
        - Tools: There are multiple UI accessibility checks such as accessible for automatic auditing for compliance and the WCAG compliance guide provided by the Bureau of internet accessibility for reference.
        - https://www.boia.org/blog/how-to-check-wcag-compliance-a-quick-guide
        - https://accessibe.com/accessscan?utm_feeditemid=&utm_device=c&utm_term=%2Bwcag%20%2Btesting&utm_source=google&utm_medium=ppc&utm_campaign=GSN_%7C_US-

CA_%7C_Accessibility_and_Compliance_Checkers_(accessScan)&hsa_cam=9492882453&hsa_grp=97916663993&hsa_mt=b&hsa_src=g&hsa_ad=589939198502&hsa_acc=%7B5473750088%7D&hsa_net=adwords&hsa_kw=%2Bwcag%20%2Btesting&hsa_tgt=kwd-301781234372&hsa_ver=3&gad_source=1&gclid=Cj0KCQiAuqKqBhDxARIsAFZELmKw7J-fQQqlFobDalX287Fwx01erCh2iF5ag2tfkIot2uwTLG8Y_CAaAgjMEALw_wcB

- We will also be using Nielsen Norman Groups 10 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
    - Match between system and the real world
        - Using a quiz design that feels normal to a quiz format given in Canvas with the integration of software to add unique but familiar process to taking the exam
    - User control and freedom
        - Allowing the user to be able to resize the screen and be able to accurately be able to interact with the environment in a way they feel comfortable.
    - 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.
    - Error prevention
        - Have proper error messages to let the user know if an error has occurred such as loss of connection or question timeouts.
    - Recognition rather than recall
        - Having recognizable components that are easy to navigate without instruction.
    - Flexibility and efficiency of use
        - Allowing for accessibility such as screen readers or language preferences for the user.
    - 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.
    - Correction guidance for errors
        - Having the proper explanations of what to do in the case of an error during exam creation or while taking an exam.
    - Documentation
        - Having clear and readable documentation for all groups of users who will be interacting with our software.

    https://www.nngroup.com/articles/ten-usability-heuristics/

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. This automated unit testing can be implemented into our CI/CD for Gitlab and the server.

5.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

The crucial systems that come into play for integration testing come from the integration between our web application and OpenAI's API, as well as the integration of our application into Canvas using Canvas's API. This, again, will be primarily handled by using unit testing with Cypress tests as well as selenium testing, if need be, for multiple instance testing for load capacity or multi-window functionality.

5.4 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?

This will be done by using basic Pytests on the meta prompt to ensure that our constraints and directions are being followed

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?

These tests will focus on the effectiveness of prompt engineering

Unique testing situation.
We will be using active environment testing with our prototype next semester. Our active prototype will get in classroom testing via professor Duwe's class as quick one question conversational assignment to be able to see working version and be able to gather large amount of information on user experience and potential downfalls of the system.

In addition to system testing, we are going to have another instance of a ChatGPT model check the first instance's output to verify the validity of the original instance. This will add another layer of testing to filter the responses from ChatGPT in order to provide the best responses to each student.

5.5 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.

A few tools that will be used for this are:

- Py Test for unit testing and benchmarking model improvements.
- Cypress to make sure the UI stays consistent with changes made to the overall system and model changes.

5.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional, are being met? How would you involve your client in the 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.

Additionally, we have talked to our client about testing with current students in a computer engineering class. This way we can have real students in our client's course take a quiz on our system to get real time feedback if our program works correctly and asks the correct questions. Doing this, we can change our data into system to fit what the system may have missed.

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

5.7 Security Testing (if applicable)

Our security testing will be tied in with our system level testing, essentially, does ChatGPT follow our directions and adhere to its given constraints

5.8 Results

What are the results of your testing? How do they ensure compliance with the requirements? Include figures and tables to explain your testing process better. A summary narrative concluding that your design is as intended is useful.