Clarify: A Modernized Educational Forum

Tyler Cady
tcady3@gatech.edu
Georgia Institute of Technology

Abstract

This report discusses Clarify, a modernized approach to an educational forum aimed to combat knowledge obsolescence and support knowledge rejuvenation. First we will introduce Clarify, its proposed implementation, and timeline, followed by our executed solution and its results, as well as present a conclusion primarily focused on self-evaluation.

I. Introduction

Clarify challenges the preconceptions of how a traditional educational forum operates and what it should provide. We take advantage of the reality of knowledge obsolescence and rejuvenation to stray from current leaders like Piazza and Ed Discussion, who abandon older, yet important information while obfuscating current up-to-date resources. Throughout this section, we will discuss an overview of this project, any motivations pertaining to the ideation of Clarify, and finally, the skills that are bound to be learned.

i. Overview

As an application, Clarify will begin similar to existing forums, containing many of the core features necessary for an educational forum to be designated as so (i.e. courses, threads, tags, etc.), however, the application will quickly take on many unique features to differentiate itself from archaic models. The following speak, in order of tentative completion, of the anticipated features of this system:

Streamlined Course Setup and Expansion: Using Canvas's roster feature, an instructor can simply download the associated CSV file and upload it to set up a new course or expand a current one. This allows courses on a forum level to not be strictly class/semester dependent, but rather, become instructor/course dependent, unlike previously seen before. By uploading a file, students will be automatically given privileges via their associated email, further eliminating the need to join a course from the students' end, as it will be automatically handled throughout this process.

Anonymized Users: The sign up process will solely require an email and password from the user (apart from 2FA sent to their email), so we will be utilizing purely anonymous handles when setting up forums. The usage of names in forums is irrelevant to the knowledge being discussed, thus, while the nomenclature will be consistent by user, it will be meaninglessly so. However meaningless, this consistency will allow administrators to continue to take necessary action against abusers of the forum like before.

Intelligent Search and Summaries: Thread-specific AI-generated summaries will save considerable time when searching for information, especially in threads that contain excessive information, some of which is not quintessential to the question or topic at hand. Beyond this, intelligent search will ensure that users can easily access the information that pertains best to their query. At first, search will be approached from a full-text point of view, but with time, it will most likely evolve to a fully semantic search.

Topical Knowledge Graphs: Knowledge graphs, which will originally be developed as knowledge buckets, will be created to help visualize relationships between specific threads and generalized concepts. Grouping threads automatically with predetermined, and likely later, dynamic concepts will help students utilize Clarify as more than just a forum, and instead, as a study resource to refresh commonly asked questions based on topics that they may struggle the most with.

ii. Motivation

The motivation behind Clarify comes from an unpleasant experience as a teaching assistant at the Georgia Institute of Technology. Knowledge, being the core of any course, is only particularly useful if it is readily available. The goals of teaching assistants should be to provide knowledge to students as easy as possible, however, the current education forum model used by Piazza and Ed Discussion lack the necessary features and flow to allow for this. A huge issue currently propagating across higher education is the disorganization surrounding large class sizes. When courses reach a large number of students, the current usage of tags, filters, pinned posts, and megathreads in educational forums are simply not enough for effective organization. Further, older information that is lost between semesters due to our current conceptions of forums in education by isolating information on a semester basis is often re-asked, but may not be answered as satisfactorily, while new information may not be as relevant, but is more readily available to the student. With

help of the aforementioned features, Clarify aims to rework the foundations of how educational forums interact with those longing for knowledge, while providing an application to streamline this foundation.

iii. Skill Growth

Continually throughout the development of Clarify, multiple levels of skills will be grown and practiced on a daily occurrence. In the realm of basic and general skills (L1), the constant programming and use of various platforms, tools, and libraries, both uncommon and not, will allow for a substantial progression of reading and writing skills at a high level. Narrowing in on meta-skills through technical practice (L2), the necessity to integrate multiple platforms, which will all be discussed in the following section, and to learn the skills necessary to do so will help us take our proposed implementation which currently resides in theory, and see how it behaves in practice, while checking assumptions constantly, challenging our fact versus opinion L1 skills once again. Finally, with regards to advanced skills in trend recognition (L3), this project will immediately help begin the exposure to application pull and technology push, through the evolving technologies and advances that have recently been created/made, both in AI and not, to allow for a specialization/optimization of previously generalized educational forum functionalities that students are exposed to, which will hopefully evolve the state of forums in a cyclical nature once again.

II. Proposed Solution

How we will go about rectifying these plentiful issues in educational forums will stem from a rapid implementation and testing cycle with regards to the completion of the aforementioned desired features. Beginning similar to other educational forums, like Piazza and Ed Discussion, which will be discussed in the section below, we will develop from the ground up to allow for a complete customization of the system at hand. By utilizing technological advancements, the setup should be quite simple with help of Next.js and Supabase, and after a baseline implementation of the core features that make an application a forum (e.g. the posting, reading, and commenting of/on threads), we will begin implementing as many features as permissible with the time allotted. In order of priority of the previously discussed features, we will aim to complete (1) a streamlined course setup and expansion process, (2) anonymization of users, (3) intelligent search and summaries, and finally, (4) topical knowledge graphs. In the end, we hope to produce an application, or rather, a system, that effectively combats the preconceptions of educational forums through the implementation of several, if not all, of these features. Throughout this section, we will discuss related work that helps to build the foundations of our starting point, the technologies that will be tentatively utilized, a plan for our design and implementation of such a system, and lastly, how we will strive to evaluate our resulting application.

i. Related Work

To serve as a foundation, we have researched several aforementioned leaders in the educational forum space. Of these leaders, two stand out in specific: Ed Discussion and Piazza. Both of these education forums provide important features that are required of an application in order to be designated as a forum. Some of those features/aspects include a basic forum structure, CRUD threads, an intuitive UI, and the pagination of threads [ResetEra2021]. While these features will be included within the application, they are far from novel, thus, we delved deep into what the two existing competitors have non-trivially implemented to differentiate themselves. Ed Discussion prides themselves on their ability to incorporate many different modes of collaboration into their threads, such as code, math equations, images, and videos [EdDiscussion2025]. However, despite these many modes of collaboration, increasing the number of modes doesn't improve the knowledge obsolescence issue significantly nor does it ensure knowledge rejuvenation. With Piazza, they pride themselves on their many diverse features, such as link sharing to threads, immense user statistics, and their unresolved/resolved thread question/follow-up structure [Piazza2025]. While the latter feature is quite intelligent I will admit for ensuring that no thread goes unanswered, I believe the focus on external-facing features based on URLs and user statistics defeats the point of an educational forum. An education forum shouldn't be for competition nor to necessarily welcome outsiders, but rather, to encourage information growth, maintenance, and understanding within the existing forum and its users. With an understanding of these related works in the education forum space, we have a good idea of what current star players are lacking, yet also what they do well, to help us begin the development of our system.

ii. Technologies

Clarify will be utilizing several technologies throughout its implementation, all of which are tentative due to the potential discovery of new technologies that better meet the specific requirements of certain tasks as we progress with its development. At the time of this proposed solution, we are most definitely using Next.js as our main

front-end framework, Supabase for our back-end services and authentication, Tailwind CSS for interface styling, and TypeScript as our central programming language to ensure type safety throughout our code base. Beyond this stack, we will also likely be utilizing Postgres vector extensions for full-text and semantic search, JavaScript visualization libraries such as D3.js for visualizing our knowledge graphs, Vercel for hosting a finalized production-grade deployment, Resend to handle emailing services, CloudFlare as our core DNS to ensure heightened security, and OpenAI's API for AI-related queries. Throughout Clarify's development, any new technology that will be incorporated into the code base will be recorded in detail to ensure that it is properly mentioned throughout this report.

iii. Design and Implementation

Designated as a core portion of this report, much of our discussion with regards to the design and implementation of our system was previously mentioned in the previous sections up until now. However, to ensure a concise and solid understanding, we will reiterate our plans for the design and implementation of our project at a more holistic level. At a design level, our forum will borrow many similar stylistic features that are necessary from the average educational forum like Piazza and Ed Discussion, however, with a more modernistic UI similar to that of YC companies and startups like Stripe and Brex. The ability of our unique features to be successful will be dependent on the quality of our UI, so it is important that the design of our application is quite solid. The design process of certain pages will either be grown implicitly from its baseline UI when setting up a feature, or if necessary, will be drafted in detail on Figma prior to its implementation. Apart from the design of our system, we must always consider how it will be implemented. Our Supabase back-end will allow for a simplified and safe authentication process, and our use of Next.js for our front-end backed by TypeScript will ensure the availability of countless up-to-date and secure libraries and packages to aid in our core forum development, as well as our unique features of AI summarization, smart search, topical knowledge graphs, and whatever else is deemed a critical feature throughout the development of Clarify. Further, its compatibility and seamless integration with Vercel will allow for a streamlined process for deployment to production when such a time arises. Finally, the implementation of the aforementioned core and unique features will be tracked/version controlled by Git via GitHub, to aid in the flow of development. With implementation details specific to each core and distinct feature having previously been discussed, it should now be apparent of our general design and implementation plan at this point in time for our system.

iv. Evaluation

A valid evaluation of our system in its final form at the end of this semester is imperative to gauge if the previously mentioned skills were truly grown and if such a project was truly feasible. More so, the completion of this project with regards to its evaluation is a strong reflection of my planning and implementation skills as well, whether that be successful or not. Thus, a "successful" implementation has to meet the following criteria: (1) a fully functioning autonomous production deployment that (2) showcases at least three out of four of the proposed features, while (3) containing already populated forged/exemplar data to eliminate the possibility of an empty demo. If these three criteria can be met, I believe the status of the system is a success with the allotted time for its development, but by no means is its development finalized. If the three criteria cannot be met, it will be important to reflect on what went wrong, and if we reflect properly, I believe success can be drawn from a different lens, but lesser at that.

v. Datasets (Addendum)

In light of the recent failure modes being addressed in class and via means of Canvas by Dr. Calton Pu, I want to clarify how my project will not fail any of these criteria, nor is it my intent to by any means. I believe that I have strongly provided related work to serve as a strong foundation for my system, and further, I do not require approval for a group size of three, since I am in an individual group, nor am I doing a structured project, so all of these core failure modes are satisfied. However, with regards to incorporating pre-existing datasets, I believe that I could have been more specific throughout this report thus far to explain why an empty demo will not be possible. Throughout the definition of my success criteria, I state the following: "[my system will contain] already populated forged/exemplar data to eliminate the possibility of an empty demo." My original plan was to create this data from scratch throughout my development of Clarify, to prevent an empty demo from occurring like discussed in class. But after listening to Dr. Calton Pu's worries with regards to the proposed solutions, I think it will be incredibly more beneficial to utilize pre-existing data to ensure an enhanced quantity of data and efficiency in incorporating it. Therefore, to supplement my plans to create my own dataset to showcase my application, I will also be tentatively utilizing data from at least one of the following resources to serve as a starting point for my database schema, as well as for populating example

courses and interactions within my forum: (1) Coursera Forums Dataset [CourseraForumsDataset], (2) Education Dialogue Dataset [EducationDialogueDataset], and (3) Stack Exchange Data Dump [StackExchangeDataDump]. These three datasets correspond greatly with my project's goals, and I will explain their significance and relevance with regards to my project below. I have admittedly come to understand that without datasets, my project had a stronger chance of becoming an empty demo, however, I hope that it is quite clear now that I by no means intend to do an empty demo, and that I agree that I am much better off thanks to the inclusion of the aforementioned datasets.

Significance and Relevance of Datasets

As previously discussed, I will be tentatively utilizing at least one of the previous three datasets throughout my development to ensure that an empty demo realistically cannot occur. The first dataset is as follows: Coursera Forums Dataset [CourseraForumsDataset]. This dataset provides anonymized discussion threads from the forums of 60 MOOCs in CSV format, which could be incredibly useful in populating course specific threads and data. This dataset contains about 100,000 threads, and since they are all from a course perspective already, they will fit in perfectly with the educational-style forum that we are building. The second dataset is as follows: Education Dialogue Dataset [EducationDialogueDataset]. This dataset contains conversations between a teacher and student on specific topics (of which can be parsed) generated by an LLM, which could work to greatly supplement the previously mentioned dataset to provide more realistic discussions in an academic setting. With over 40,000 training examples, I believe it will provide us with meaningful data to experiment our features with, and even better, it provides the data in a JSON format to provide us with more options and ways to populate our database. A key use that I see this dataset being potentially useful for is with a course based on historical figures, as there are many interesting conversations regarding them. Finally, the third dataset is as follows: Stack Exchange Data Dump [StackExchangeDataDump]. This dataset is known as one of the largest publicly available question and answer datasets online, which contains posts on many broader topics such as Game Development, English, and Physics. While broader topics may be out of scope for the course-specific design that is in mind for Clarify, I believe that the sheer volume of data will be useful in providing us with valuable styles of information sharing that the other two datasets may lack. Apart from filling in the gaps of the other datasets, this dataset simulating a forum through a question and answer format will be very useful in testing core thread features like replies and comments, if the other datasets aren't productive or comprehensive enough. With the discovery of these three tentative datasets, of which more may be later utilized, I believe this dataset report addendum is officially concluded due to its explanation of preventative measures with respect to an empty demo.

III. Timeline

Clarify will be an individual semester-long project with the aims of creating a system that solves many critical issues that arise in traditional educational forums. Throughout this section, we will quantify everything discussed in the previous two sections to provide an organized and concise plan for the timeline regarding the implementation of this system.

i. Task Breakdown and Schedule

Below, you will find a tentative breakdown and schedule with regards to the implementation of our system:

Dates	Milestone	Description
Weeks 5-7	Authentication and Enrollment	Integrate Canvas's roster export schema for seamless course setup and user authentication.
Weeks 7-9	Core Forum Development	Implement fundamental anonymous forum features such as threads, comments, and tags.
Weeks 9-11	AI Summarization and Search	Implement AI-driven summarization for threads and enhance search capabilities.
Weeks 11-13	Knowledge Graphs	Develop and integrate topical knowledge graphs for improved information visualization.
Weeks 13-15	Testing and Deployment	Conduct extensive testing, deploy the application, and finalize documentation.

Table 1: Task Breakdown and Schedule

Note: If any task takes longer (or shorter) than predicted to implement, or if it is important to reorder the chronology of implemented tasks, the tasks around itself will become flexible, that is, the actual final resulting order of task completion may look different from the proposed order seen above.

ii. Work Distribution

As this is an individual project, the work will be solely completed through the means of myself.

IV. Executed Solution and Results

Throughout this semester-long project, we have successfully followed our proposed timeline with near perfection, and as a result, we have implemented all of our previously desired core functionalities and features at a high-quality level. Throughout this section, we will walk through each milestone in detail, in sequential order of completion, while providing useful figures, schemas, and snippets to help visualize the success and flow of our implementations. It is recommended to zoom into the report to enhance the detail of the figures if the quality appears low on the reader's end, as they are detail-packed, yet readable, if enlarged properly.

i. Authentication and Enrollment

Regarding the authentication and user enrollment flow for Clarify which incorporates Canvas's roster export schema data to continuously prevent empty demos, this section will be utilized to demonstrate the associated results. If you are not logged in, you will originally see a landing page designed to explain the goals of Clarify [Figure 1]; If you are signing up for the first time [Figure 2], once you type in your credentials, you will receive a 2FA email thanks to our integration with Resend to confirm your account [Figure 3]. Once you log in through the help of our Supabase backend [Figure 4][Figure 5], you will see a central dashboard page where all of your courses will appear (with a star next to them if you created them, without one if you didn't) [Figure 6][Figure 7]. From here, you can navigate to the settings page, where you can view account information, request admin access, and create a course if you are an admin [Figure 8]. From my dashboard, I can approve/reject pending admin requests to ensure a singular point of distributing privileges [Figure 9]. After a course is created from here, you can see it in your dashboard, and if you navigate to the course's page, you will either see a student view where forums will be implemented [Figure 10], or if you are the creator, a settings page to facilitate course enrollment [Figure 11]; By uploading a file to this page of the same format as Canvas's student roster export data [Schema 1], you will automatically set up privileges for users based on their course role, such that if they log in from their end, they will be automatically included in the course. Thus, we have successfully implemented our first milestone at a high caliber and in its entirety.

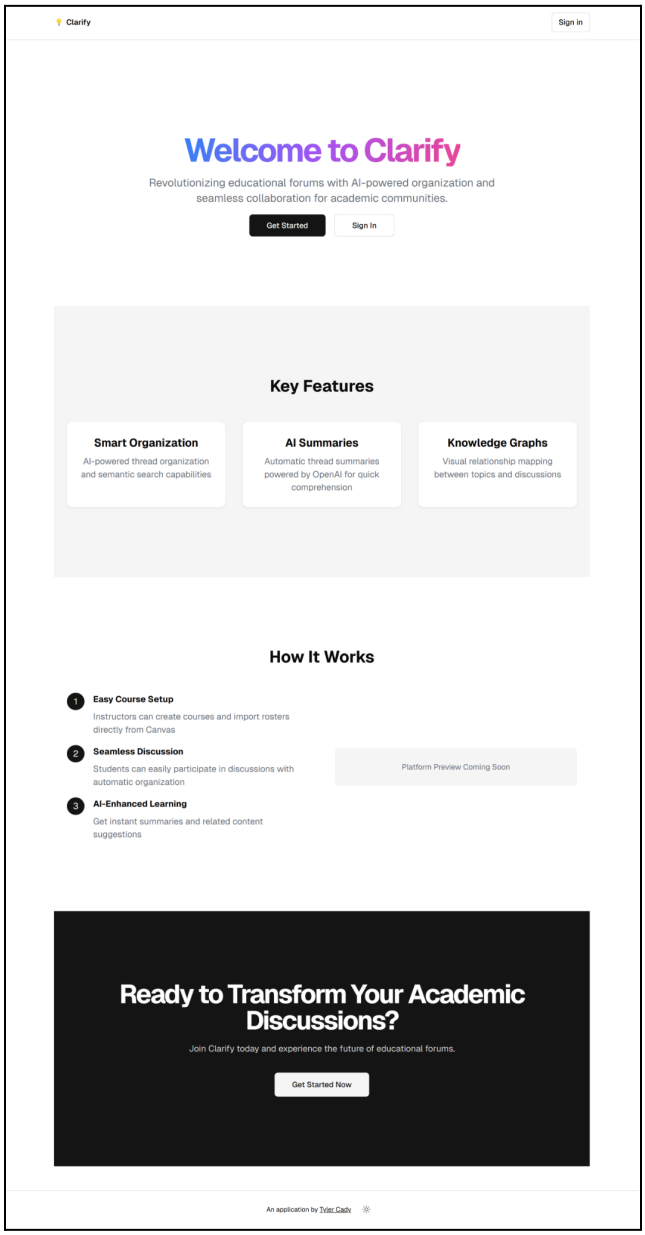


Figure 1: Landing Page

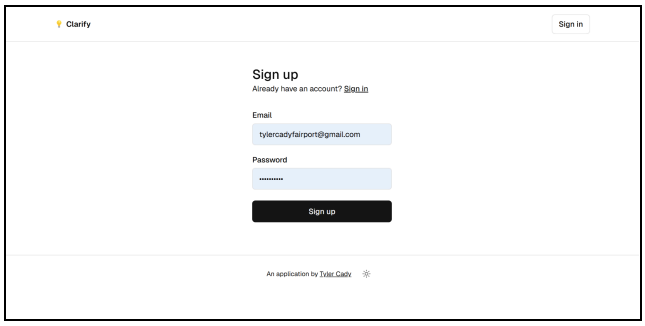


Figure 2: Sign Up

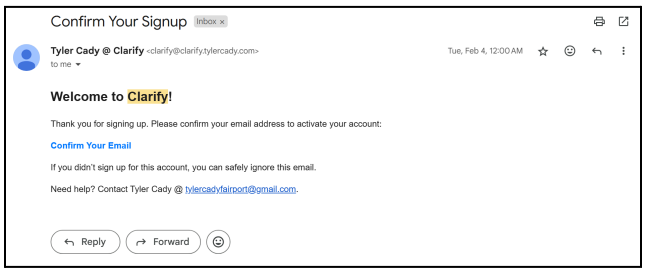


Figure 3: 2FA Email

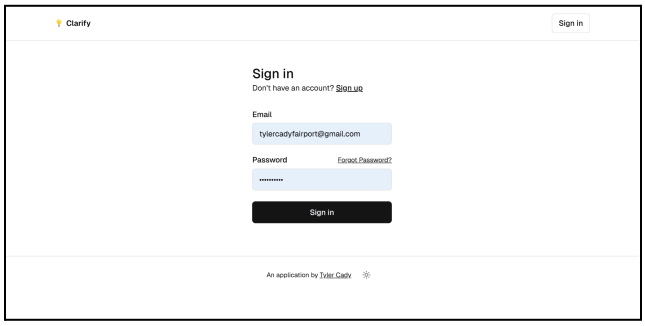


Figure 4: Sign In

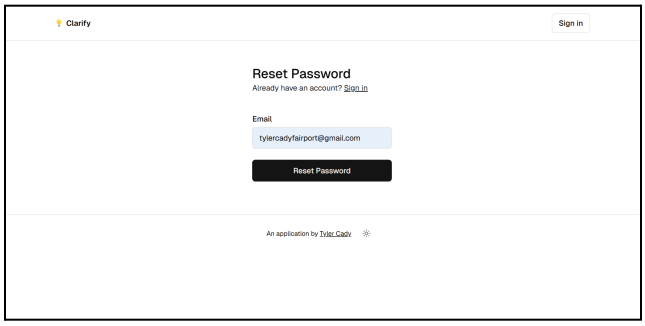


Figure 5: Reset Password

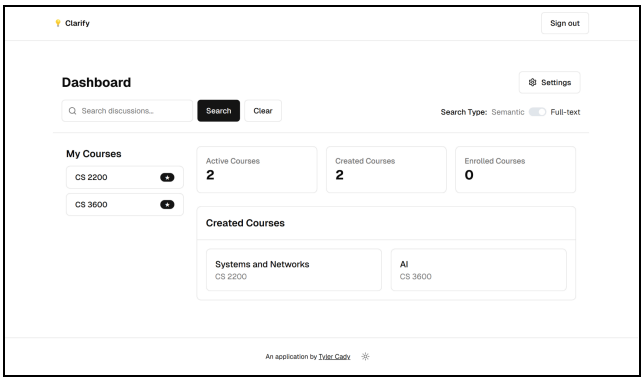


Figure 6: Admin Dashboard

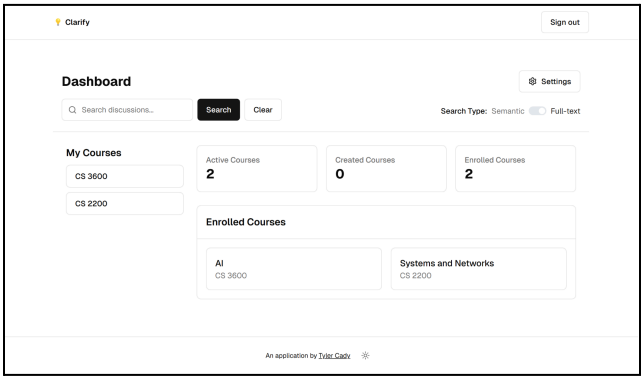


Figure 7: User Dashboard

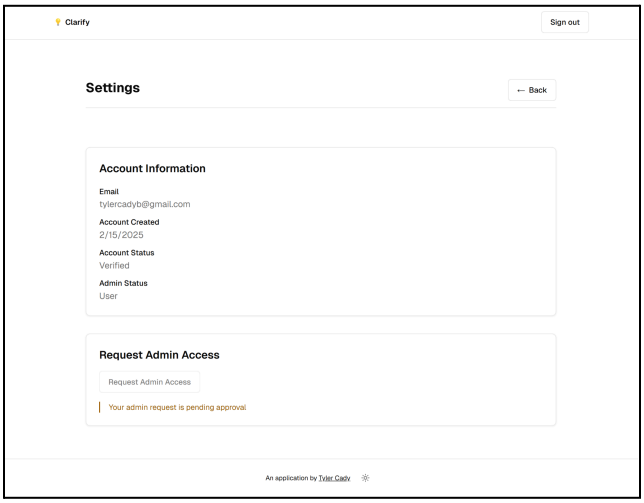


Figure 8: User Settings

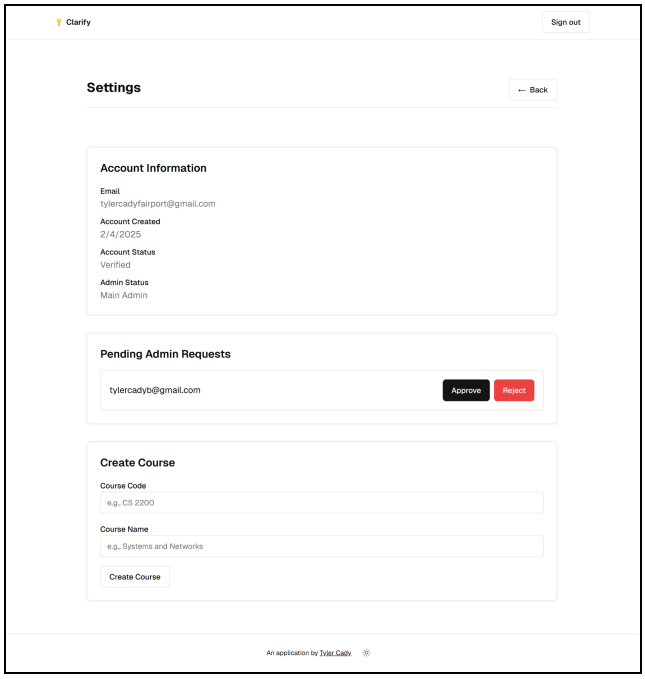


Figure 9: Admin Settings

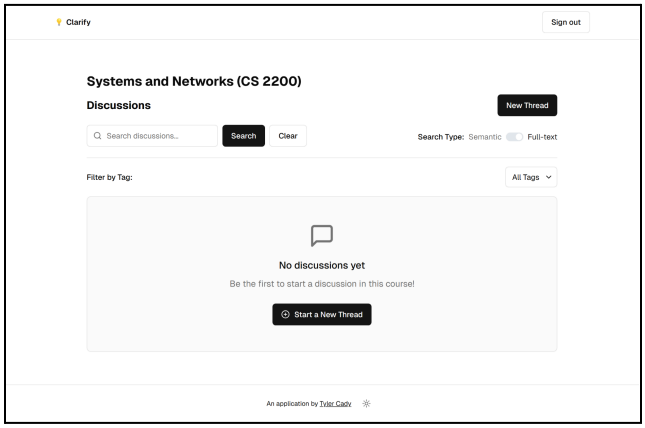


Figure 10: User Course Page

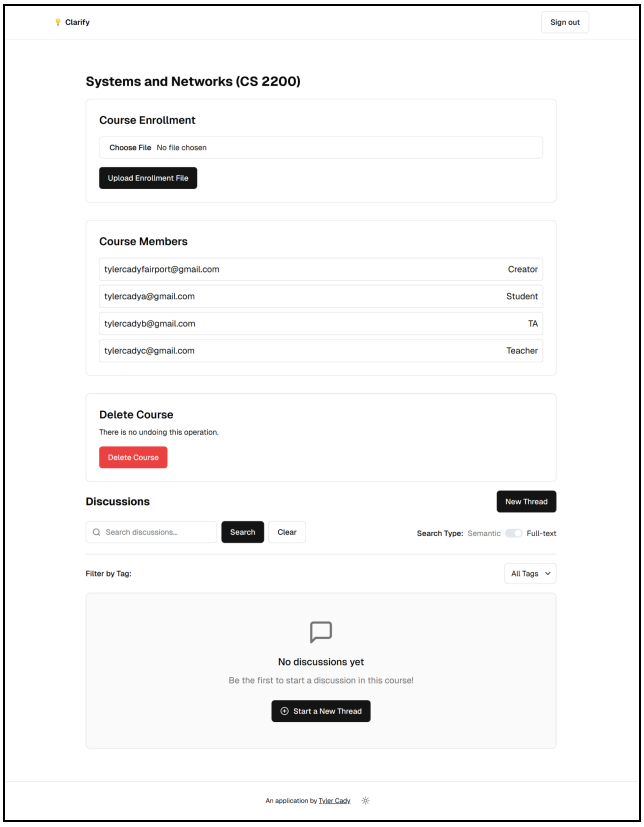


Figure 11: Creator Course Page

	Student #1	**Student #2**	**Student #3**
Name	Doe, John A	Smith, Jane B	Brown, Charlie C
Email	tylercadya@gmail.com	tylercadyb@gmail.com	tylercadyc@gmail.com
GTID	903111111	903222222	903333333
GT Account	jdoe3	jsmith9	cbrown7
Major(s)	us/c/coc/bscs/a/cs/cs08/info/internetwork-infrastructure	us/c/coc/bscs/a/cs/cs30/info/internetwork-systems	us/m/mgt/bsba/a/ba/mg04/information technology
Role	Student	TA	Teacher
Section(s)	202408/CS/2200/A/80169, 202408/CS/2200/A03/88677	202408/CS/2200/A/80169, 202408/CS/2200/A04/88678	202408/CS/2200/A/80169, 202408/CS/2200/A05/88679
Confidential?	N/A	N/A	N/A
Grade Mode	Letter Grade	Letter Grade	Letter Grade

Last Course Activity	2024-12-10 10:00 EST	2024-12-12 14:30 EST	2024-12-15 09:45 EST
Total Course Activity	100:30:15	120:45:30	95:20:10

Schema 1: Canvas Student Roster Export Data

ii. Core Forum Development

Regarding the core forum development features for Clarify which ensures an anonymized and persistent approach, this section will be utilized to demonstrate the associated results. The features that were implemented in an anonymized fashion are as follows: course-specific discussion boards, threads (with editing and deletion capabilities), comments (with editing and deletion capabilities), tags, replies, and upvoting. If you navigate to a course as a student, you will see the threads associated with the course, or an invitation to create the first thread of the course [Figure 10][Figure 12]. After creating a thread yourself [Figure 13], you will be able to upvote, edit, delete, or navigate to the thread [Figure 14]. Once you navigate to a thread, you can see or contribute to all related comments of the thread and utilize key features like upvoting and replying [Figure 15]. All of the thread and forum features take an anonymized and persistent approach, that is, if someone comments on a thread, they will either be given the name of OP (meaning "original poster") if they are the creator of the thread, or, a random AdjectiveAnimal name from a permutation pool of over 2500 names. This name is persistent and unique to each user, in that for the entirety of a user commenting on a thread (note that anonymous names are thread specific), the name will remain the same for the user and be unique to the user only. Through extensive testing and creation of new administrative features like course deletion (that deletes all data associated with a course) [Snippet 1], I am confident to say that all of these core forum features are sound and solid. Thus, we have successfully implemented our second milestone at a high caliber and in its entirety.

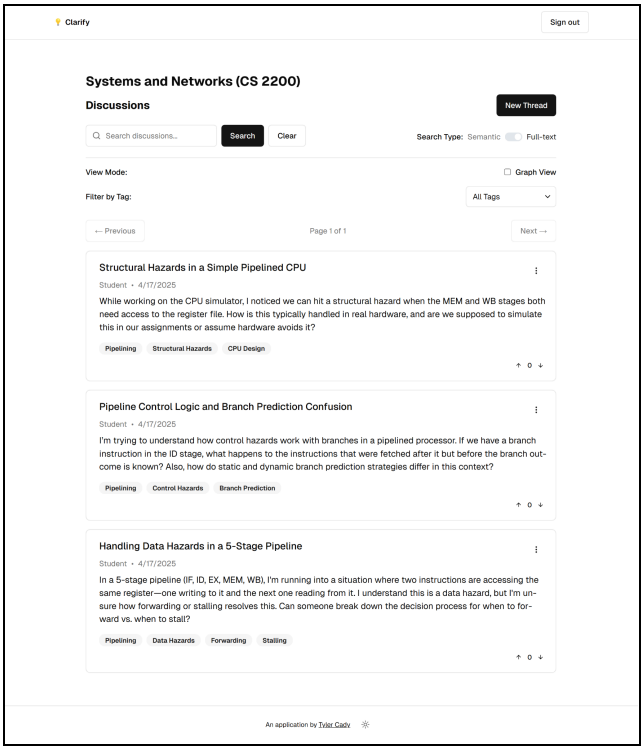


Figure 12: Populated User Course Page

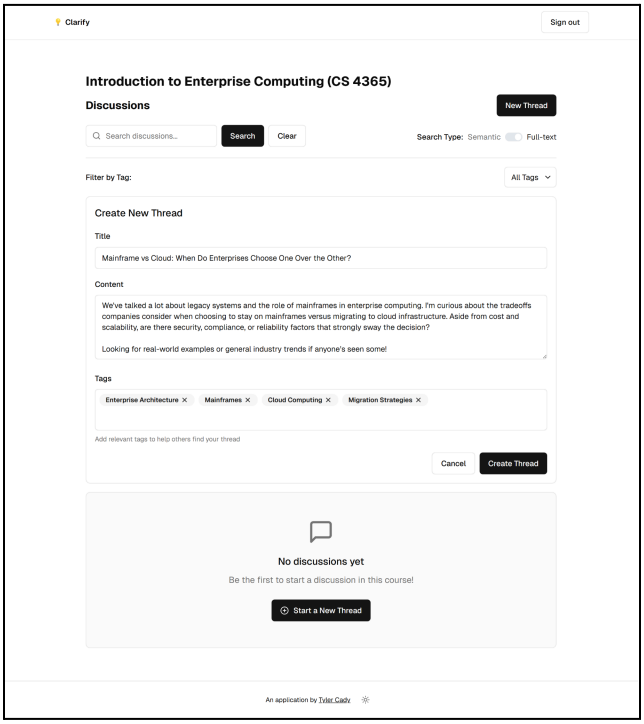


Figure 13: Thread Creation (can set title, content, & tags)

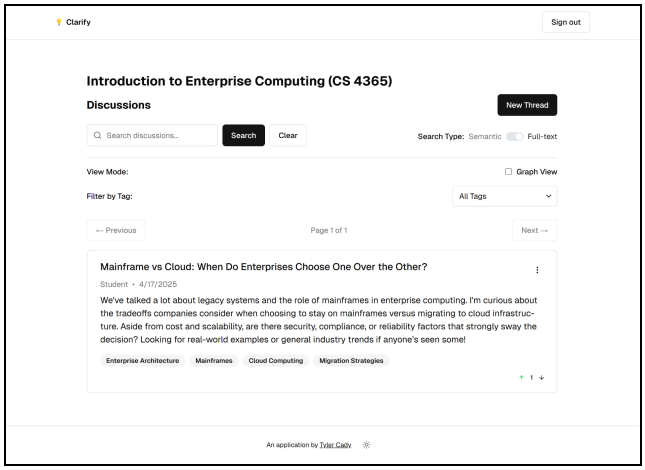


Figure 14: Created Thread (vertical ellipsis allows editing & deletion of the thread, arrows allow upvoting)

Figure 15: Thread Page (fully isolated thread-specific view; see: comments, comment replies, anonymized handles, & continued editing/upvoting permissions)

SQL

```sql
create or replace function delete_course(course_id_param uuid)
returns void
language plpgsql
security definer
as $$
begin
    -- Delete all thread votes for threads in this course
    delete from thread_votes
    where thread_id in (
        select id from threads where course_id = course_id_param
    );

    -- Delete all threads in the course
    delete from threads
    where course_id = course_id_param;

    -- Update course_enrollments to remove this course
    update course_enrollments
    set courses = coalesce(
        (
            select array_agg(c)::jsonb[]
            from (
                select c
                from unnest(courses) c
                where (c->>'courseId')::uuid != course_id_param
            ) sub
        ),
        '{}'::jsonb[]
    )
    where exists (
        select 1
        from unnest(courses) c
        where (c->>'courseId')::uuid = course_id_param
    );

    -- Finally delete the course itself
    delete from courses
    where id = course_id_param;
end;
$$;
```

Snippet 1: Delete Course Procedure/RPC

iii. AI Summarization and Search

Regarding the AI summarization and search functionalities for Clarify which correspondingly update existing features and database schemas to support artificial intelligence, this section will be utilized to demonstrate the associated results. The functionalities that were implemented were thread-specific AI summarizations generated with help of OpenAI's gpt-3.5-turbo model and API given the context of a thread and its related comments [Figure 16][Snippet 2], full-text search for both comments and threads [Figure 17], semantic search backed by OpenAI's text-embedding-3-small model and API for both comments and threads [Figure 18], and filtering of threads by tag [Figure 19]. When a user navigates to a thread, they will either see a previously generated AI-summary stored within our Supabase database given no new updates (via comments or edits) were made to the thread and its related contents, or otherwise, a new AI-summary will be generated, stored, and displayed. This will be displayed below the original content of the thread in a clearly labeled module. If a user then navigates back to their main dashboard (containing all of their courses), they will see a search bar that allows them to search across all of the

threads within all of their courses. While searching, the user will also have the choice of enabling our full-text search, or our semantic search, based on what best fits their need [Figure 17]. After searching, they will see all relevant results (sorted by match score), as well as a match score (calculated via trigram matching for full-text search or cosine vector similarity for semantic search) for each thread and an ability to clear the search results or navigate to any result [Figure 17]. If a user navigates to a specific course, they will be able to utilize the same search functionalities but given solely the context of the course, to better narrow their queries (note: the provided figures for this section demonstrate this kind of intra-course searching; the exact same UI is utilized for inter-course searching, so it is omitted from being an additional figure in this section). Finally, we've added intra-course filtering by tag, to ensure an additional previously unmentioned enhancement to searching. Thus, we have successfully implemented our third milestone and in its entirety.

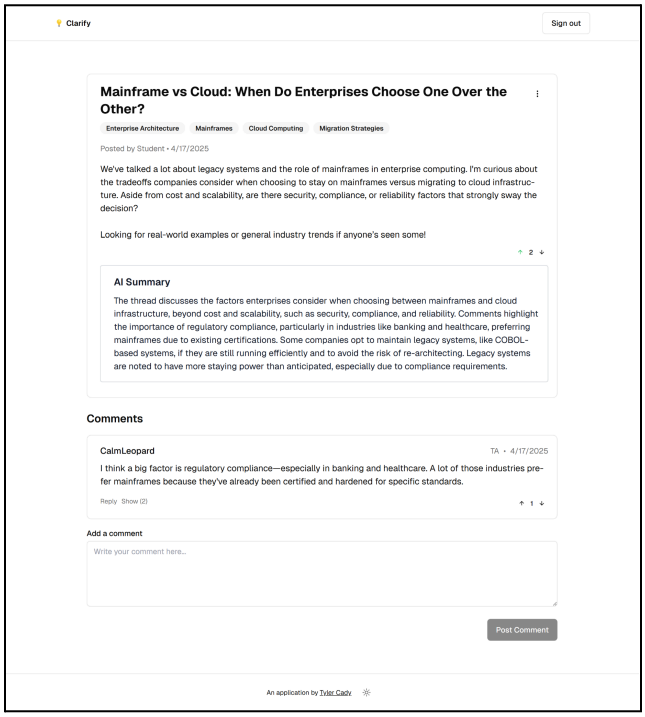


Figure 16: AI-Summarization (generated from thread and comment context)

```typescript
const threadContent = `Title: ${thread.title}\n\nContent:
${thread.content}`;
const commentsContent = comments
    .map((comment) => `Comment: ${comment.content}`)
    .join("\n\n");
const fullContent = `${threadContent}\n\n${commentsContent}`;

const response = await openai.chat.completions.create({
    model: "gpt-3.5-turbo",
    messages: [
        {
            role: "system",
            content:
                "You are an educational assistant that summarizes
academic discussions. Provide a concise summary in a few sentences,
focusing on the main content of the thread and key points from the
comments.",
        },
        {
            role: "user",
            content: `Please summarize the following educational
thread and its comments in a few sentences:\n\n${fullContent}`,
        },
    ],
    max_tokens: 150,
});

const summary = response.choices[0].message.content;
```

Snippet 2: Thread Summarization Code

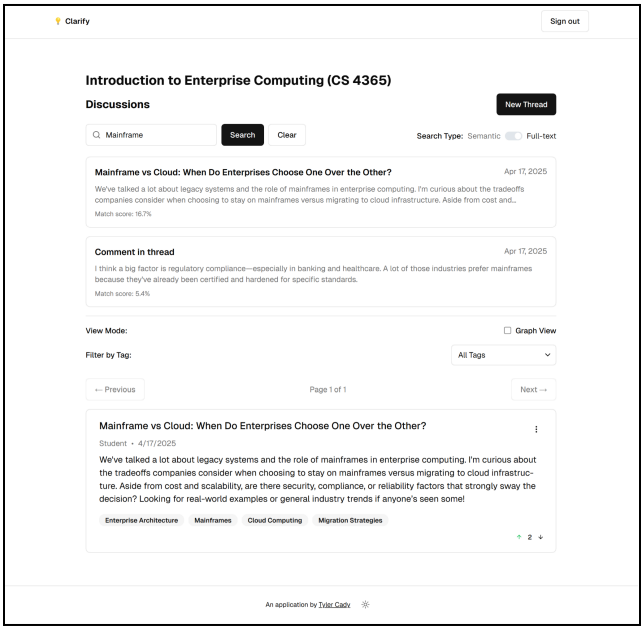


Figure 17: Full-text Search (notice the search type toggle, the match score ranking of returned results, and the ability to clear search results)

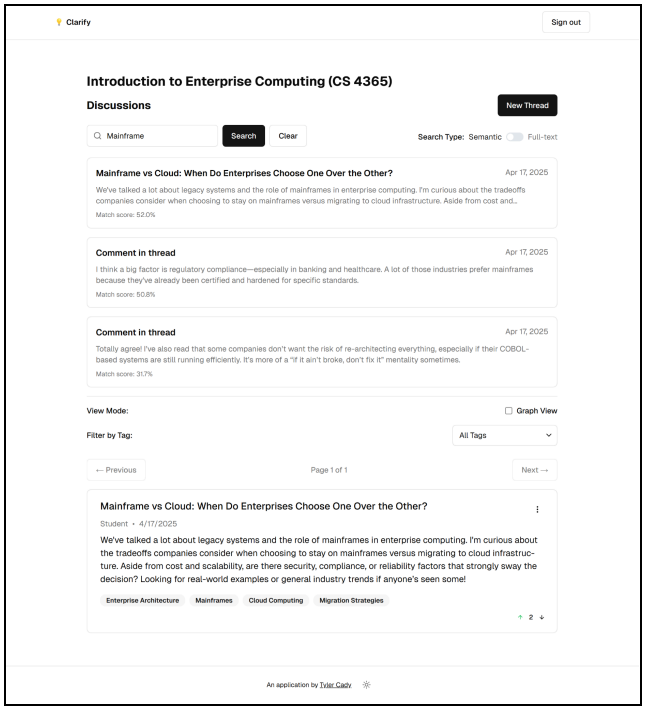


Figure 18: Semantic Search (notice how a thread appears that doesn't contain the search word)

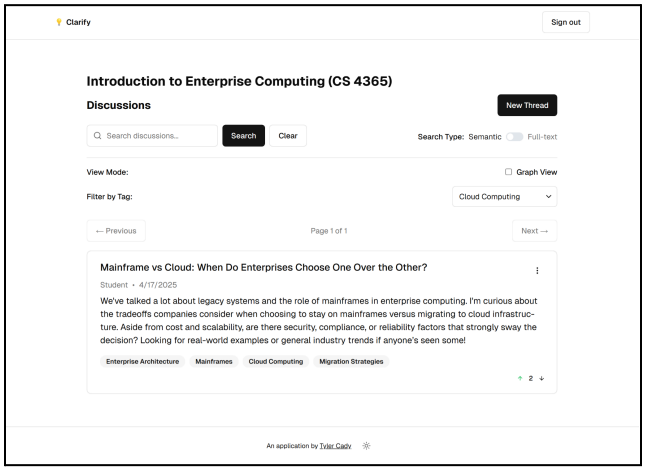


Figure 19: Tag Filters

iv. Knowledge Graphs

Regarding the knowledge graphs for Clarify which ensures optimized information visualization regardless of device size, this section will be utilized to demonstrate the associated results. When a user navigates to a course, they will now have the option to toggle their view mode to graph mode, upon which the listed threads will disappear and a knowledge graph will instead render [Figure 20]. The knowledge graph creation API accesses and utilizes the previously stored embeddings of each thread, along with two threshold values (*SIMILARITY_THRESHOLD* and *LINK_THRESHOLD*) to determine if threads cluster into the same node or if nodes should be linked together. After determining the "nodes" of the graph based on the similarity threshold and the "edges" based on the link threshold, our application utilizes D3.js to render two different kinds of graphs: bubble graphs and force graphs. While the force graph [Figure 20] is more interactive and malleable with respect to the user, the bubble graph [Figure 21] is more rigid, allowing for user preference to optimize one's viewing and navigation of information. Apart from the choice of which graph to view, the user may also pinch or button zoom and move around the graph to find the nodes that they are searching for, as well as resetting the graph if the user accidentally navigates too far [Figure 20][Figure 21]. In terms of optimizing information visualization, we have designed our knowledge graphs such that nodes with more threads are visibly larger, and upon clicking a node, a modal will pop up allowing the user to view semantically related threads, and thus, navigate to threads of interest [Figure 22]. This will be incredibly useful to users in discovering core concepts of a course and having access to related threads to study from upon finding the aforementioned concepts (note: hovering over nodes also provides information pertaining to its contained threads via tooltip prior to opening up its associated modal). To test our knowledge graphs, meaningless data was no longer of use, as a large scale of data and real-world semantics were necessary to prevent not only an empty demo from occurring, but for the true tuning of parameters in ensuring our knowledge graphs genuinely work. Thus, Python scripts were also created to populate data from Stack Exchange Data Dump [StackExchangeDataDump] (one of our previously proposed datasets) into our database via a mock course, allowing us to prematurely complete one of our few core evaluation goals remaining [Snippet 3]. Thus, we have successfully implemented our fourth milestone and in its entirety.

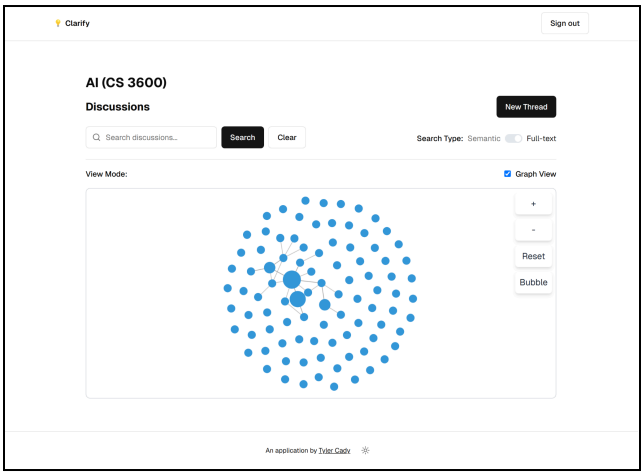


Figure 20: Force Knowledge Graph (notice the view mode toggle and the knowledge graph controls [i.e. zoom, reset, & toggle graph type])

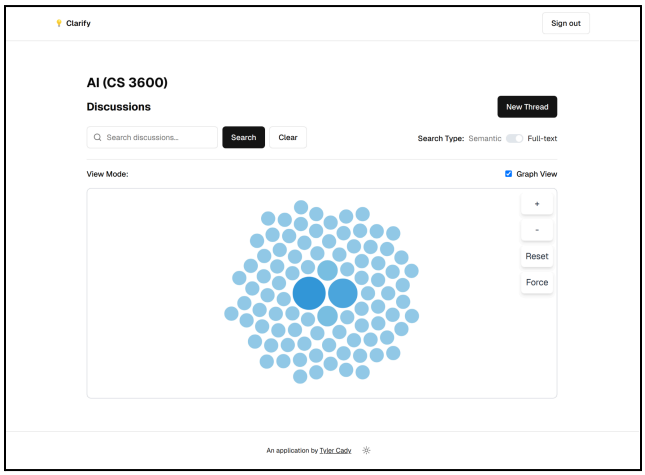


Figure 21: Bubble Knowledge Graph (not shown: mobile responsive & information appears upon hovering over nodes)

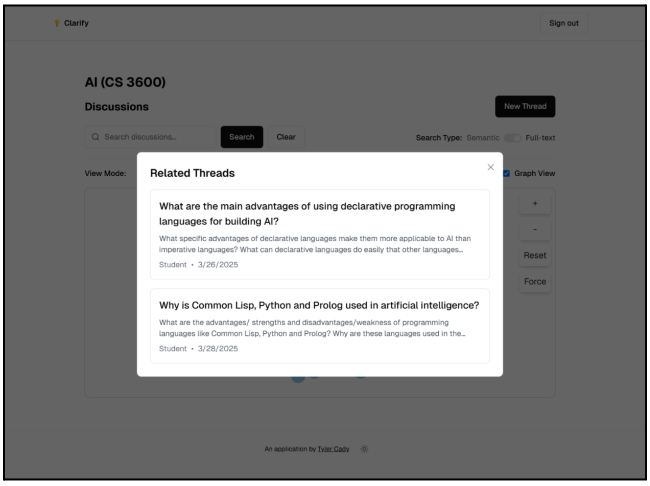


Figure 22: Node Navigation Modal

```python
import os
import csv
import uuid
import datetime
import openai
from supabase import create_client
from bs4 import BeautifulSoup
from typing import List
from dotenv import load_dotenv

load_dotenv()

openai.api_key = os.environ.get("OPENAI_API_KEY")
COURSE_ID = os.environ.get("COURSE_ID")
CREATOR_ID = os.environ.get("CREATOR_ID")

def generate_embedding(text: str) -> List[float]:
    """Generate embedding for text using OpenAI's API."""
    response = openai.embeddings.create(
        model="text-embedding-3-small",
        input=text
    )
    return response.data[0].embedding

supabase =
create_client(os.environ.get("NEXT_PUBLIC_SUPABASE_URL"),
os.environ.get("NEXT_PUBLIC_SUPABASE_ANON_KEY"))

def read_csv(file_path):
    """Read a CSV file and return a list of dictionaries."""
    with open(file_path, 'r', encoding='utf-8') as f:
        return list(csv.DictReader(f))

posts = read_csv('data/ai/Posts.csv')
tags = read_csv('data/ai/Tags.csv')

tag_post_map = {}
for post in posts:
    if post['Tags'].count('<') == 1:
        tag = post['Tags'].strip('<>')
        if tag not in tag_post_map:
            tag_post_map[tag] = []
        if len(tag_post_map[tag]) < 1: # use one tag per thread
            tag_post_map[tag].append(post)

threads = []
for i, (tag, post_list) in enumerate(tag_post_map.items()):
    for post in post_list:
        title = post.get("Title", "")
        body_html = post.get("Body", "")
        body_text = BeautifulSoup(body_html,
features="html.parser").get_text()

        content = f"{title}\n{body_text}"
        post_embedding = generate_embedding(content)

        thread = {
            "id": str(uuid.uuid4()),
            "course_id": COURSE_ID,
            "title": post.get("Title", "No Title"),
            "content": BeautifulSoup(post.get("Body", "No
Content"), features="html.parser").get_text(),
            "tags": [tag],
            "creator_id": CREATOR_ID,
            "creator_role": "Student",
            "created_at":
datetime.datetime.now(datetime.timezone.utc).isoformat(),
            "updated_at":
datetime.datetime.now(datetime.timezone.utc).isoformat(),
            "embedding": post_embedding
        }
        threads.append(thread)
    if i == 50: break # set to the number of threads (plus one) you
want to create

response = supabase.table("threads").insert(threads).execute()
print(response)
```

Snippet 3: Dataset Thread Database Population Script (note: modified slightly from GitHub for report rendering)

v. Testing and Deployment

Regarding the testing and deployment for Clarify which involves finalizing the application and its documentation, this section will be utilized to demonstrate the associated results. Throughout the compilation of this report (which serves as our core artifact regarding documentation), many experimental tests took place throughout the process of acquiring our figures, which resulted in the fixes of many bugs (e.g. replies not rendering the appropriate data), in addition to the newly implemented client-side pagination of threads [Figure 23]. Beyond this, we have finalized and fully refined our Supabase database schemas and their

dependencies to exist properly and efficiently for the sake of our application [Schema 2][Schema 3]. After implementing all of the necessary tweaks and changes to our system, we have finally ensured a tentatively stable and deployed production of our application, which at the time of this report can be accessed at https://clarify-4365.vercel.app/, using Vercel as our hosting provider. Thus, we have successfully implemented our fifth and final milestone and in its entirety.

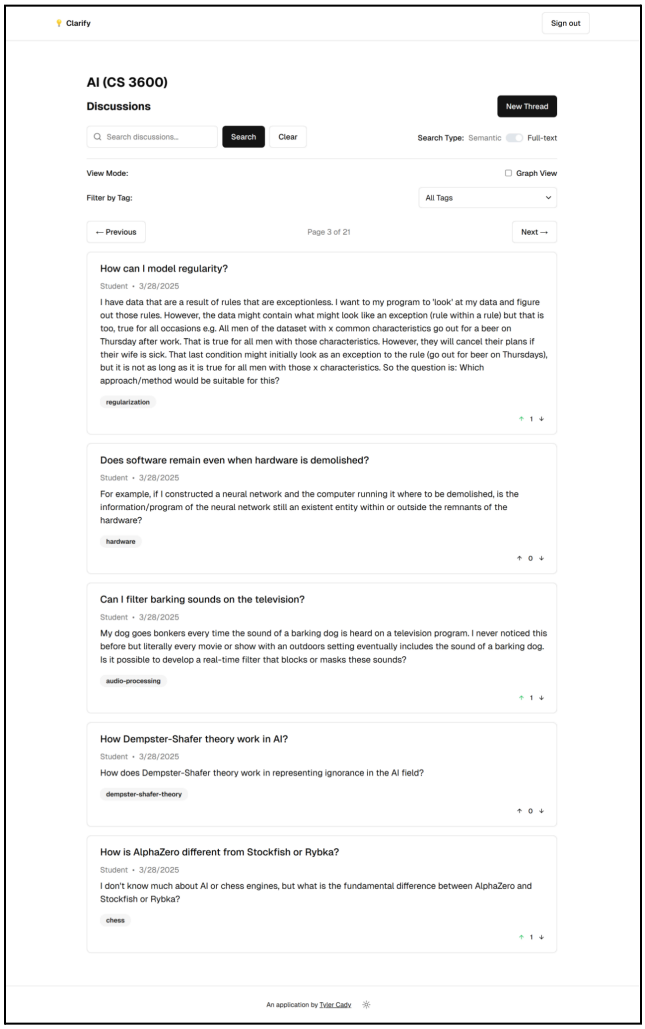
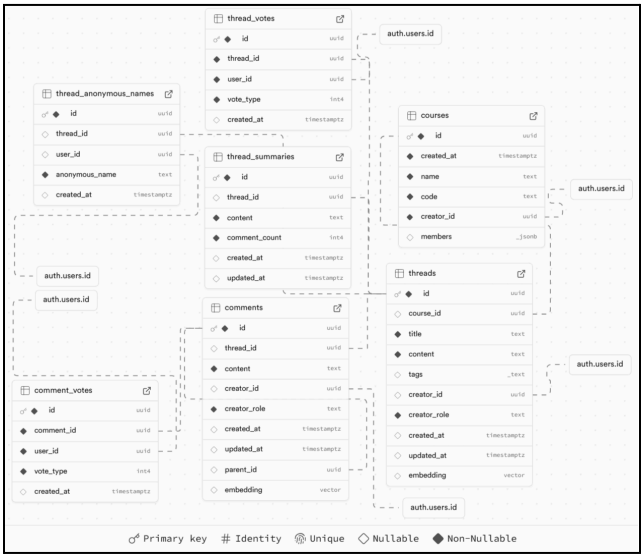


Figure 23: Thread List Pagination

Schema 2: Independent Database Tables



Schema 3: Dependent Database Tables

V. Conclusion

As previously mentioned, Clarify challenges the preconceptions of how a traditional educational forum operates and what it should provide. With the support of the aforementioned solutions presented and implemented in their entirety, we readily allow our new forum to take advantage of the reality of knowledge obsolescence and rejuvenation to stray from current leaders like Piazza and Ed Discussion, who abandon older, yet important information while obfuscating current up-to-date resources. Throughout this section, we will present a self-evaluation with respect to the successful conclusion of this project, potential avenues of future work, all of our deliverables for our completed project, and finally, an in-depth analysis of all of our skills learned and factually how so.

i. Self-Evaluation

Objectively serving as the most important part of this report and having completed all of our milestones in their entirety at this point in time, we shall spend a moment self-evaluating the work done this semester. At the beginning of this report, we mentioned the following success criteria: a "successful" final system will be one that is (1) a fully functioning autonomous production deployment that (2) showcases at least three out of four of the proposed features, while (3) containing already populated forged/exemplar data to eliminate the possibility of an empty demo. And after having spent the majority of this report showing the quality implementation of all of the proposed features plus some, the integration of a real-world dataset plus exemplar data, and the resulting live production deployment viewable via Vercel, I believe that it is quite obvious of the successes of this project from a technical, planning, and autonomous learning viewpoint. Turning a semester-long project into the greatest application I have built as a team of one is an incredibly rewarding experience, and to solidify my pride in and belief that this is a great project that fueled autonomous learning tenfold, I will briefly summarize the scope, match, and factual evaluation points below.

Scope

The scope of our project was quite large and path breaking in my opinion. If we had only proposed a singular feature, I could see it being an unimpressive software solution, or if we presented features that already exist in the educational forum space, I would think the same. However, having cast a wide net of novel potential features and actually implementing them serves as proof of our impressive scope and its intent from the beginning.

Match

The match of our final project in comparison to our proposed project was near perfect in execution in my opinion. Every milestone was either completed in full as planned throughout the proposal, as backed by the previous executed solution and results section's 23 figures and additional schemas and snippets, or was supplemented even further by the addition of new features and functionalities.

Factual

The factual level of our project and its claims is immensely high in my opinion, which is once again supported through the previous section incredibly so, as well as with the live deployment accessible for anyone to verify our claims themselves. In addition, and to drive this strong factual claim home, the later appearing skills learned section includes 30 links backing the implementation and learning claims being made with respect to each skill's specific progress.

ii. Future Work

Throughout the implementation of this project, I have taken note of some potential avenues of future work both at an application level and at more of a technical level regarding what could be completed at a later point in time. At the user/application level, a few of the things that we could look into implementing could be (1) adding thread and comment counts, (2) supporting image inputs among other modes, and (3) enforcing character limits. At a more technical level, a few potential areas that we could look to improve could be with respect to (1) better prompt engineering for our AI summarization model, (2) deeper fine-tuning of knowledge graph hyperparameters, and (3) the exploration of additional data visualization methods. With all of these supplemental avenues to explore, I could see how this project could morph into a multi-semester project where someone else builds upon what has already been created thus far.

iii. Deliverables

The main deliverables for this project are (1) the GitHub repository and (2) the deployed live production site.

GitHub Repository: https://github.com/tylerrcady/clarify

Live Production: https://clarify-4365.vercel.app/ (due to the requirement of administrative privileges, you won't be able to utilize certain features unless you are an admin, thus, please request administrator privileges from me at tcady3@gatech.edu if you'd like to view these features firsthand)

iv. Skills Learned

Throughout the development of this application and self-evaluation, I believe that a lot of skill growth has occurred. Below, I will mention my progress towards each of my originally mentioned skills with factual supporting evidence linked for claimed progress.

L1: Basic and general skills

Progress: To progress this skill, the continuous programming and interactions with various languages, platforms, and tools are exceptionally advancing my reading and writing technical skills.

Milestone 1: I have utilized TypeScript as the core language of our Next.js application as well Python to help generate non-confidential data of the same format as the Canvas enrollment export data, which actively enhanced my programming skills in these languages. This can be viewed below at link [1] which shows the languages utilized in the application thus far. Further, I have worked with a PostgreSQL database through Supabase to enhance my SQL skills, which can be confirmed by the database schema, which is viewable at links [2] and [3].

Milestone 2: For this milestone, I have once again heavily utilized the languages of TypeScript and SQL for developing the aforementioned key forum features, which continuously enhances my programming skills in these languages. The use of Typescript can easily be seen throughout the GitHub repository at link [1], and while the use of SQL is implicit through our integration with Supabase, I have included an example procedure that I wrote in PL/pgSQL for course deletion (which perfectly ties in the tables created in this milestone) at link [4], to factually support this claim.

Milestone 3: For this milestone, apart from the continued use of TypeScript and SQL to support the development of the aforementioned functionalities (which continuously enhances my programming skills), which can be seen at link [1], I also utilized OpenAI's API and models seen at link [5] and [6], which helped me develop deep interactions and knowledge surrounding the importance of considering cost and performance when determining which model to use, as well as properly providing quality context and security with the usage of such artificial intelligence API's.

Milestone 4: For this milestone, apart from the continued use of TypeScript and SQL to support the development of our application (which continuously enhances my programming skills), which can be seen a link [1], we also utilized Python once more to support the integration of our datasets, as well as the Beautiful Soup, OpenAI, and Pandas libraries/packages to support the parsing of our datasets and the retrieval of embeddings prior to populating our Supabase back-end, all of which can be seen at links [7] and [8], which further helped to advance my technical toolbox tenfold.

L2: Meta-skills through technical practice

Progress: To progress this skill, the integration of multiple platforms and learnt skills throughout integration are continuously testing how my theoretical plans hold up in practice by challenging my assumptions.

Milestone 1: I have integrated Resend and Cloudflare into our tech stack to provide for the safe sending of custom and unrestricted/unlimited 2FA emails, of which all are sent from clarify@clarify.tylercady.com as seen at link [9], and further, where the domain can be confirmed protected by Cloudflare at [10]. This allows me to rectify the fault in my previous belief that 2FA would be easier to set up, but in practice, I was met with my emails going to spam and being limited in their sending until integrating with Resend. I have also utilized the integration of our Next.js application with Vercel to allow for the simplified hosting of our application, which can be attested through the production domain at link [11]. This integration worked seamlessly, which helped show where my planning went right by considering how the tech stack could interact in certain realms with other technologies included.

Milestone 2: I have spent quite a bit of time implementing and integrating (RESTful) APIs into my application to support the core forum features, of which can be factually viewed at link [12] for comments and link [13] for threads. Throughout this process of integration, I continuously found out that a lot more testing than originally planned was needed, which is why many of my API route.ts files have excessive error logging. However, I believe that the use of Next.js here for API integration was a strong move on the behalf of my planning, since the ease of Route Handlers backed by Next.js and their App Router approach made creating and calling APIs incredibly simple and allowed me to focus more on the logic of the API's function than the connection at hand.

Milestone 3: Originally, I assumed the integration of OpenAI's API to be simpler than in reality, as calling an API is quite trivial, however, apart from the determination of which model to utilize, there were still quite a lot of remaining necessary considerations. I decided during testing that due to the multiple desired AI features, that it would be intelligent to store embeddings as Supabase vectors with both comments and threads on creation and update as seen in part at link [14] and [15], respectively, to limit the number of API calls to reduce computational and economical cost, as well as storing the AI-summaries in a new table to ensure that we only regenerate summaries if there is new content (or tentatively if the last summary was generated over a day ago), to further reduce cost as seen at link [16]. I believe that my theoretical planning could have better considered the implementations necessary to respect

cost savings, something I will definitely account for going forward in my planning.

Milestone 4: At first, it was my belief that creating a visibly appealing graph in the front-end would be easy due to the many libraries that exist like D3.js, however, there was actually quite a learning curve in the integration seen at links [17] and [18]. Determining the nodes and links (i.e. edges) of each graph required a lot of tuning of the aforementioned parameters, as well as creating a batched version of an existing RPC in Supabase to speed up the fetching of the similarities for each thread, as at first, the rendering of our graphs was incredibly slow. Yet beyond this, ensuring that the graphs rendered responsively (despite device size) and had zooming features (among many others) required a deep dive into various documentations and examples of D3.js. By planning which library to use ahead of time (i.e. D3.js), it definitely helped to speed up this milestone, however, in hindsight, a deeper dive into how knowledge graphs could have been technically implemented and integrated ahead of time could have proven to be incredibly useful, and is something I will definitely do better with going forward.

L3: Advanced skills in trend recognition

Progress: To progress this skill, the exposure to specific application pulls coming with technological push has consistently provided me with a new view into how a focus with regards to optimization is easily permitted with such improvements.

Milestone 1: Originally, I wasn't sure how Supabase authentication would work, as I had only utilized Google OAuth through Next.js authentication libraries prior, however, the ease in connection with Supabase via authentication (technological push) allowed me to spend more time on focusing on my necessary enrollment flow (application pull) at all levels. To support this, you can view the provided sign-in and sign-up action functions provided by Supabase utilized at the top of the relevant file at link [19] to handle authentication, as well as the connectivity in our database seen with the connections with auth.users.id at links [2] and [3], which is essential for our authorization and management of users.

Milestone 2: The implementation of anonymized core forum features, which once could be a project in and of itself, was accomplishable within two weeks due to the incredible technological push provided through PL/pgSQL for stored procedures and improvements in generative AI. At link [20], you can see an RPC being called in the GET request to easily get comments with their anonymized names thanks to PostgreSQL capabilities (which allows for efficient testing), and further, at link [21], you can see a function called getRandomName(), which generates random AdjectiveAnimal names thanks to OpenAI's ChatGPT, which easily provided lengthy lists of animals and adjectives to combinatorially give us theoretically infinite unique names after a singular prompt. In making the cumbersome yet simple tasks easier with even just these few (but not limited to these few) technological pushes, I was allowed more time to optimize and implement additional features like upvoting and replies, which otherwise I am not sure there would have been time for.

Milestone 3: The technological push of readily accessible and strong artificial intelligence models via API easily allows for the optimization of educational forums through the implementation of AI functionalities at efficient speeds. By having access to many of OpenAI's top models through their API for a significantly minimal cost made a task that once would be computationally expensive with potentially poor results, inexpensive (by outsourcing the computation) with results subject to state-of-the-art models. The success of the implementation of the application pulls of AI within the educational forum realm to enhance understanding and search can be factually supported by the example figures provided at links [22], [23], [24], and [25]. This access to newly developed technologies made a once daunting task, much more manageable with the allotted time, allowing the implementation of additional and novel features that prove to be incredibly useful as well.

Milestone 4: The increased availability of meaningful and large datasets, like the one utilized within this milestone at link [26], fueled by technological pushes (e.g. better generative AI and data parsing strategies), allows for the optimization of all data-focused application features and pulls through the simplified access and integration of such meaningful datasets. By having application functionalities that require hyperparameter tuning, like with the knowledge graphs of this milestone, incorporating realistic data is necessary for successful results as seen at links [27], [28], and [29]. With these improvements in technology and dataset accessibility, the integration of one of our datasets was quite simple as seen at link [30], and all things considered, allowed not only optimized features, but an incredible saving of time to permit additional features to be developed (e.g. graph zooming), whereas

before, the time would have likely been spent creating an appropriate dataset manually.

References

Below you can view all of the indirectly linked article references and dataset references mentioned within our report.

Articles

[ResetEra2021] AlanOC91, (2021). What are some important required forum features? Accessed: 2025-02-04. https://www.resetera.com/threads/what-are-some-important-required-forum-features.405657/

[EdDiscussion2025] Center for Teaching Innovation, (2025). Ed discussion: Center for teaching innovation. Accessed: 2025-02-04. https://teaching.cornell.edu/learning-technologies/collaboration-tools/discussions/ed-discussion

[Piazza2025] Piazza, (2025). New piazza features. Accessed: 2025-02-04. https://support.piazza.com/support/solutions/articles/48001181453-new-piazza-features

Datasets

[CourseraForumsDataset] https://github.com/elleros/courseraforums

[EducationDialogueDataset] https://github.com/google-research-datasets/Education-Dialogue-Dataset

[StackExchangeDataDump] https://archive.org/details/stack-exchange-data-dump-2023-09-12