V-Room Final Product Submission

1. Product Description

V-Room (Virtual Room) is a web application which mainly aims at improving the virtual office hours experience in an educational setting. There are three major functionalities in V-Room.

1. Students can see their position and total number of people in the queue while waiting to be let into the office hour meeting by the host.
2. Students have access to a question-answer portal in each course that they are enrolled in, which allows them to post questions, answer other questions and vote on questions and answers.
3. V-Room also helps students, professors and teaching assistants manage their courses by giving an opportunity to create courses, assign roles to members, invite students via link, and delete or archive courses.

2. Product Functionalities and Guide

Users can navigate to https://v-room.vercel.app/ to use our product V-Room.

2.1 Login to V-room

When a user first visits V-Room, he/she will be asked to sign in using their university google email account. Users don't have to remember another set of username or password for logging into V-Room. Instead of that, they can simply login using their google account itself. After the user successfully signs in using google account, V-Room will be asking users to complete the registration process.

Fig 1: Landing page of V-Room

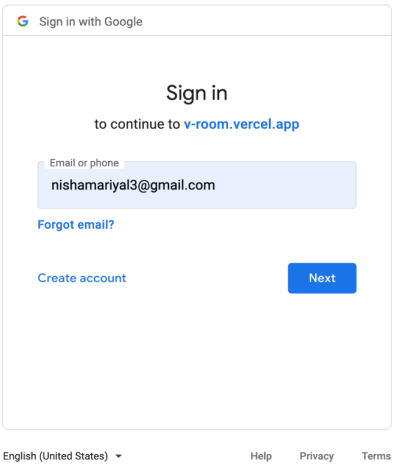

Fig 2: Google's sign-in page

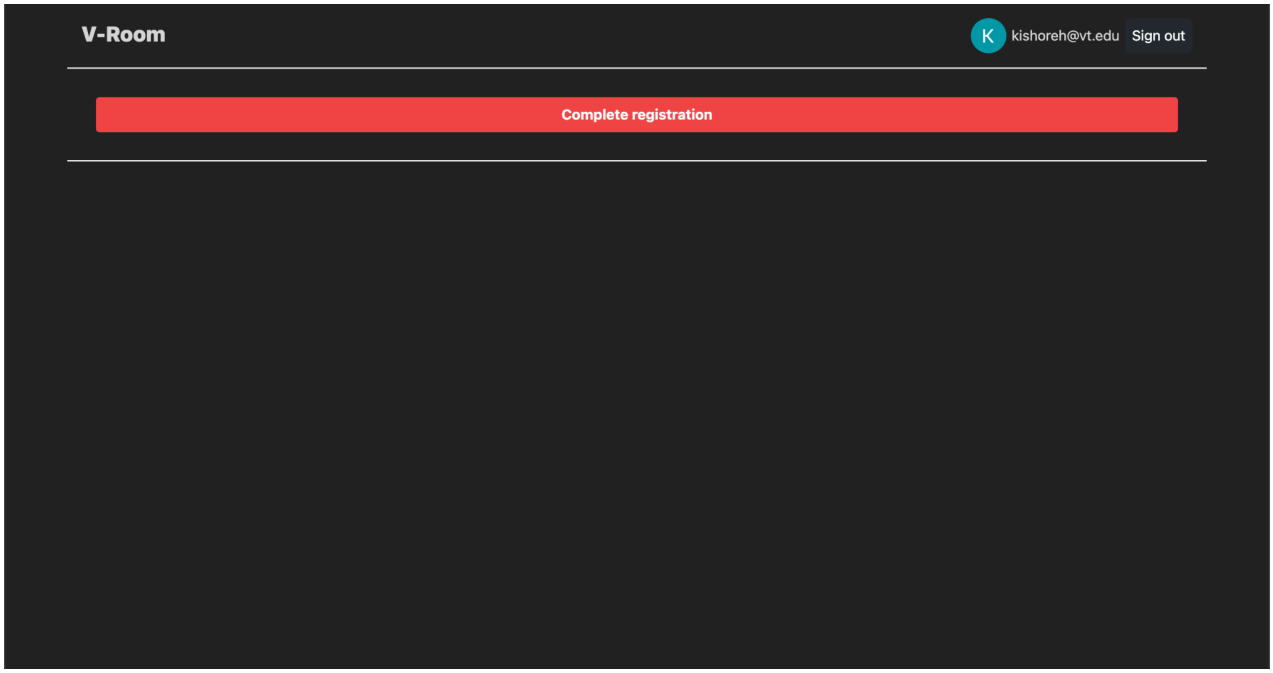

Fig 3: Landing page after successful login

2.2 Register and Update Profile

After the user successfully logs in to V-Room, users will be asked to update their name and pronouns if needed as part of the registration process. If the user is not registered already, they will be prompted to register their information such as name

and pronouns, in order to complete their registration on V-Room. After the user finishes the registration process, V-Room will be redirected to the classroom page which shows the list of classrooms that user is enrolled in.

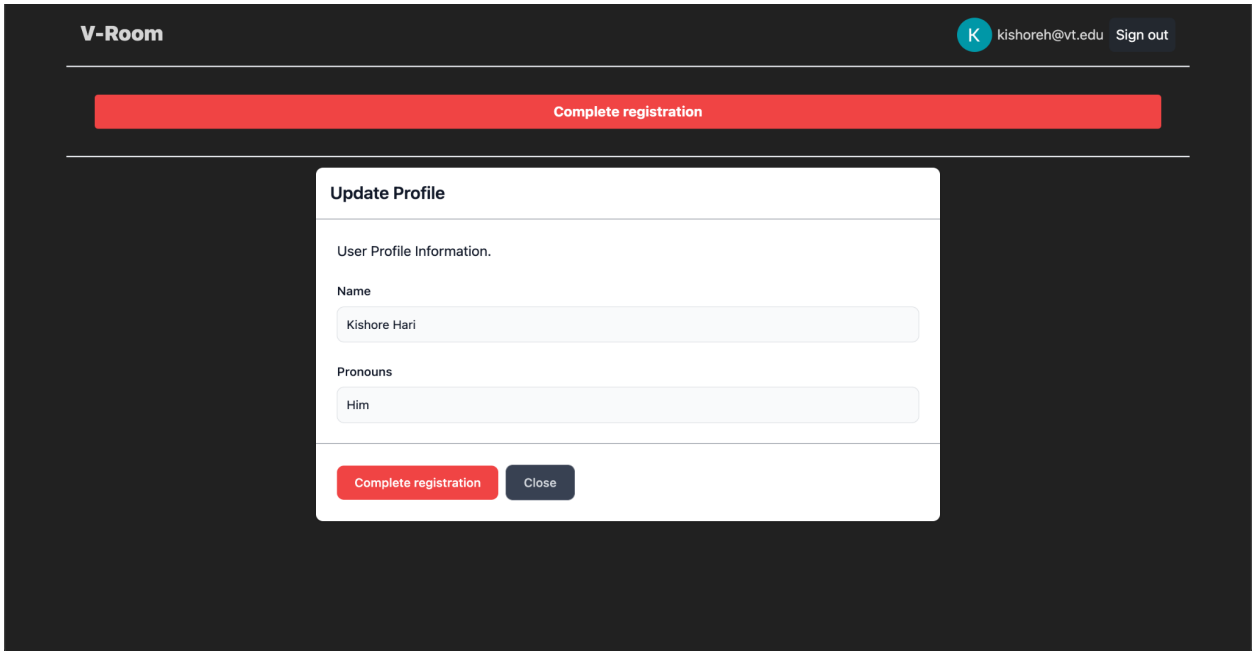

Fig 4: Update Profile/Registration page

2.3 Create classroom

Once logged in, users can access the classroom list page, which lists every classroom they're currently a member of. Depending on their role within the classroom, different options are available to them. Clicking the "Add Classroom" button brings up a modal, within which users can input information before clicking the "Create Classroom" button to add the new classroom. After adding the classroom, the list updates with the new classroom and assigns the creator as the instructor, displaying the proper fields to them, including the invite code and the ability to delete the classroom.

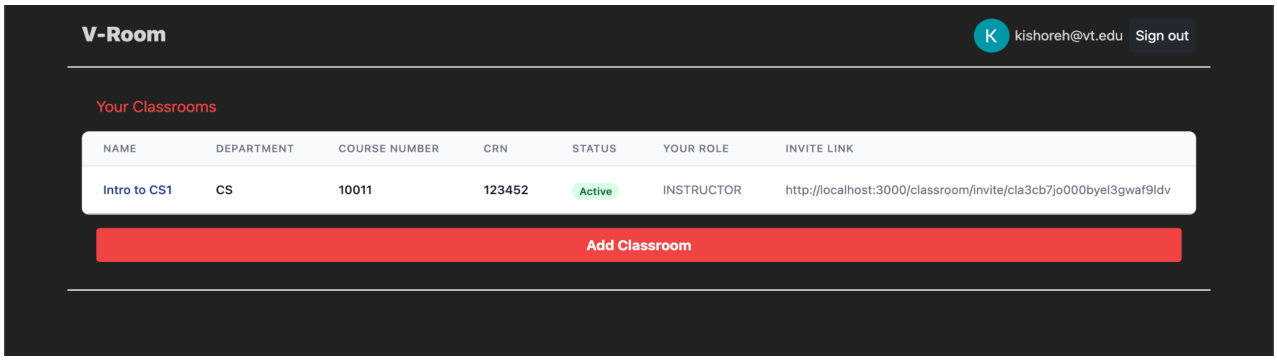

Fig 5: Classroom List Page

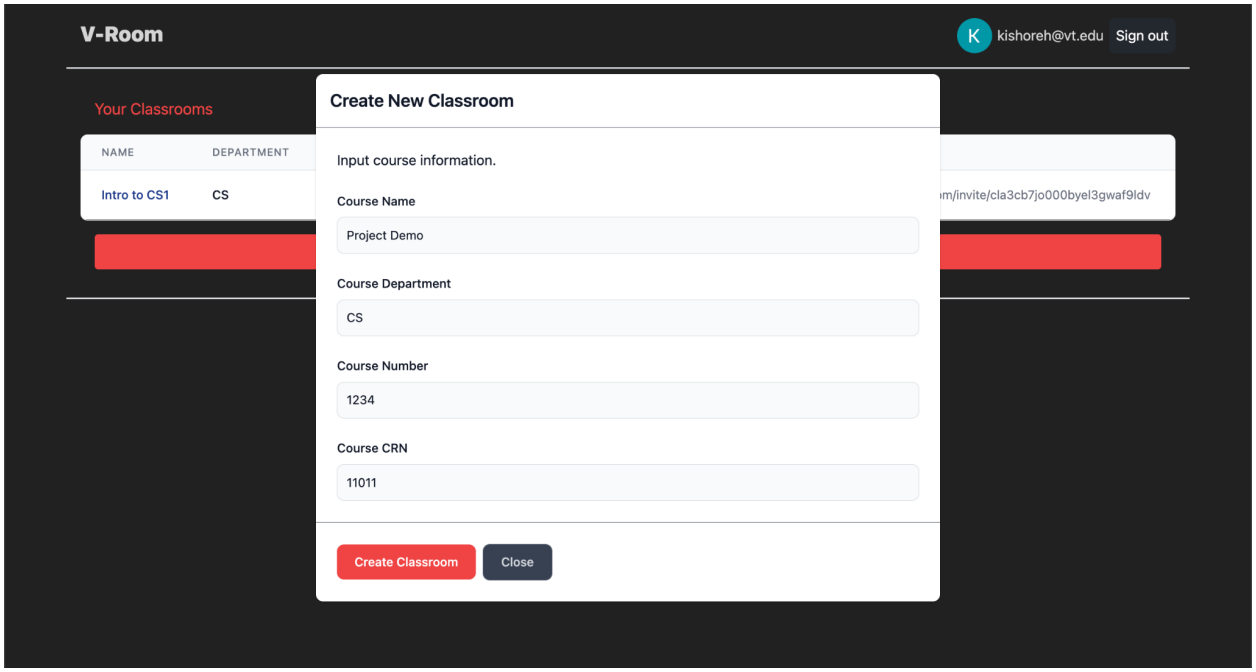

Fig 6: Create Classroom Modal

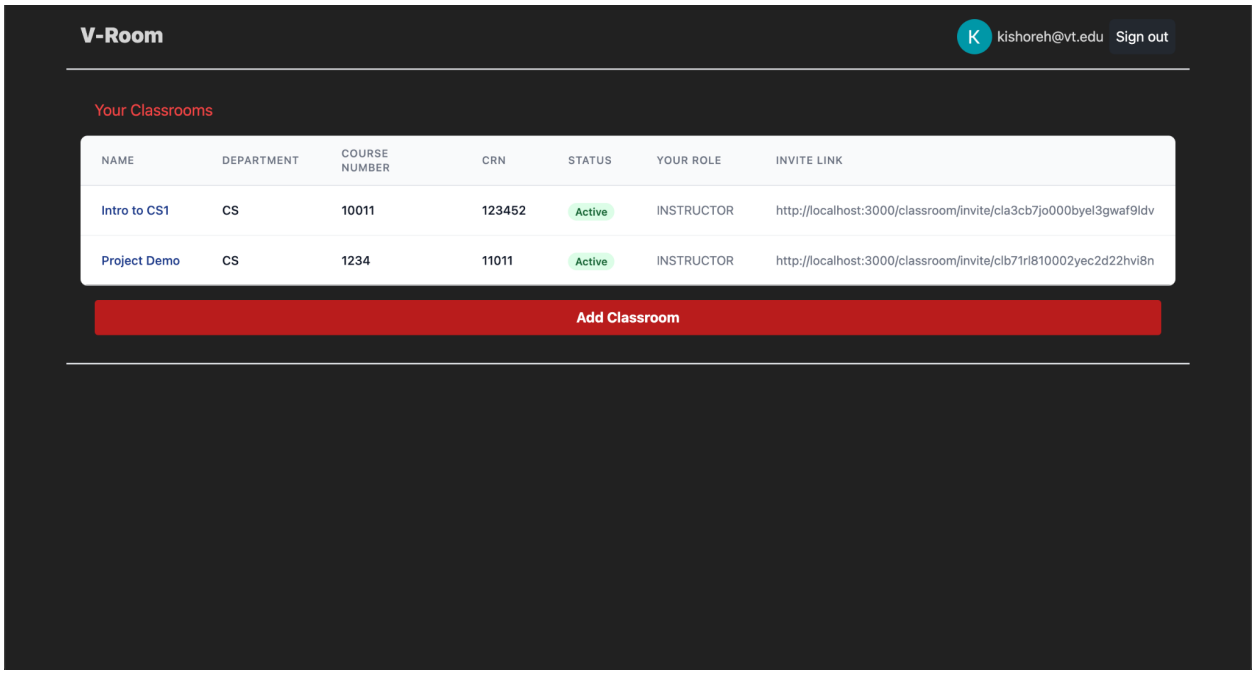

Fig. 7: List of newly created classrooms

2.4 Archive / Delete classroom

On navigating to the Classroom Detail, the instructors of the classroom can either delete or archive the classroom. This is hidden to all other roles. When the user clicks on the “Delete Classroom” button, it will clear all the data related to the classroom

from the database. When the user clicks on the “Archive Classroom” button, it will hide the classroom from the user, but the data will still be present in the database.

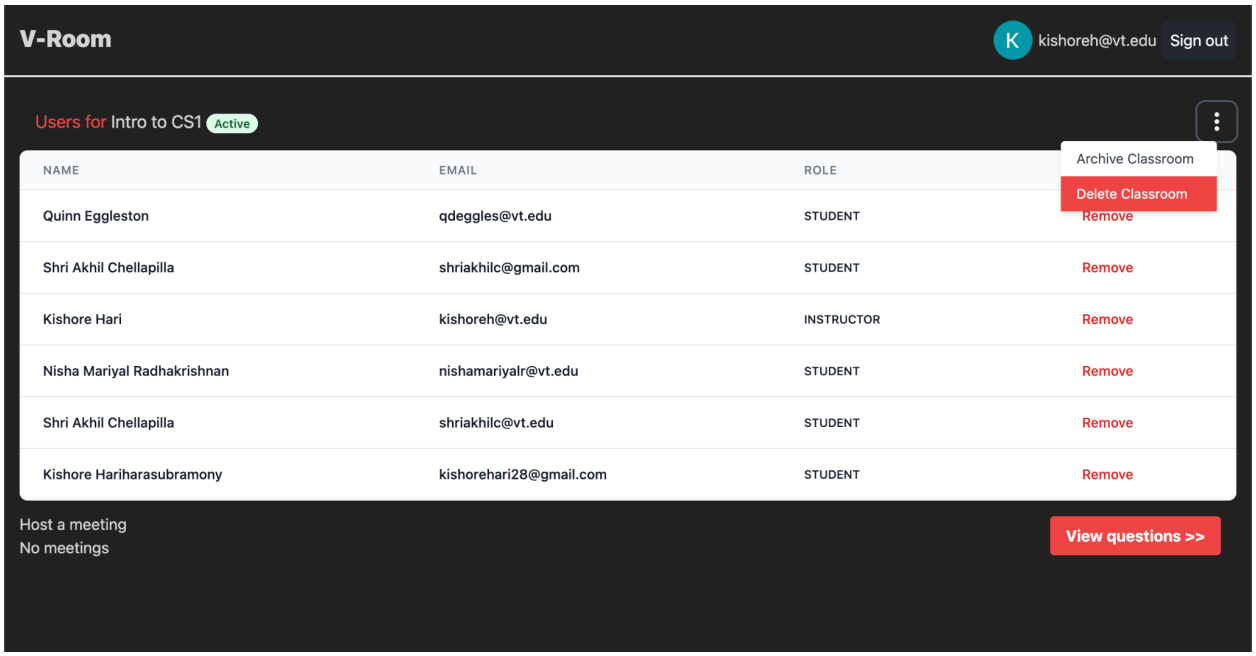

Fig 8: Option to delete/archive classroom

2.5 Add/Remove Users from Classroom

The Classroom Detail page also allows logged in instructors to add other users to the classroom and select what role to assign to them. They can also remove any users already in the classroom. This is hidden to Students.

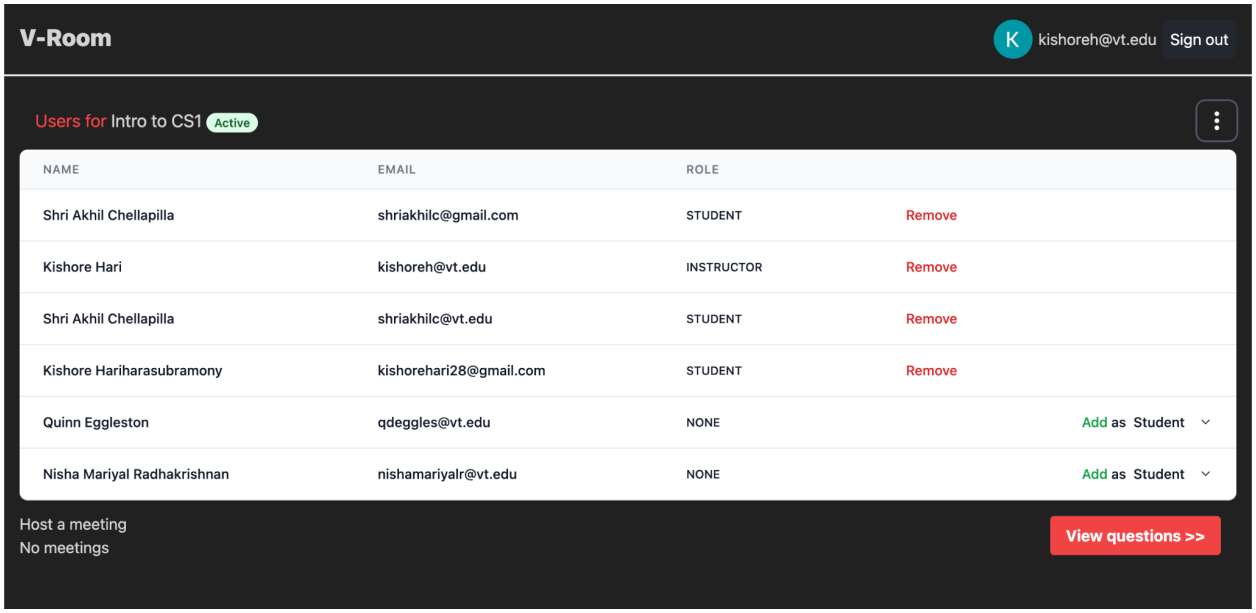

Fig. 9: Page showing Add/Remove users option

2.6 Invite students using invite code

Instructors of a classroom have access to an invite link as an alternative to manually adding students. Another logged in user who navigates to this link will automatically be added as a Student for this particular classroom. This is an alternative way to add users to the classroom. This mainly avoids instructors/TAs to manually add each student by entering their email address.

On the user's side, they are shown the name of the classroom ("Intro to CS1" in this case referring to Fig:11) and asked for confirmation to join. Upon confirmation, the database is updated and they are redirected to the classroom details page.

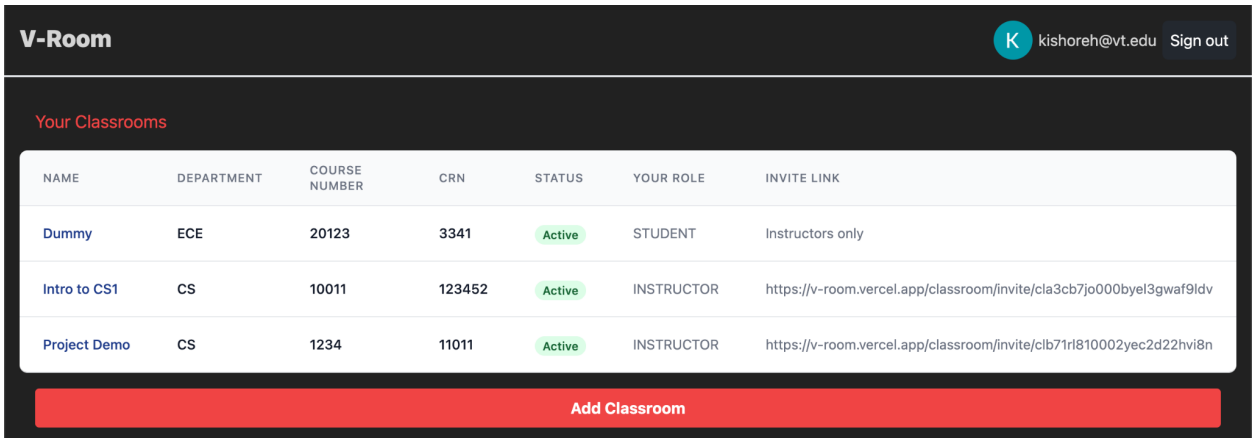

Fig 10: Classroom page showing Invite link

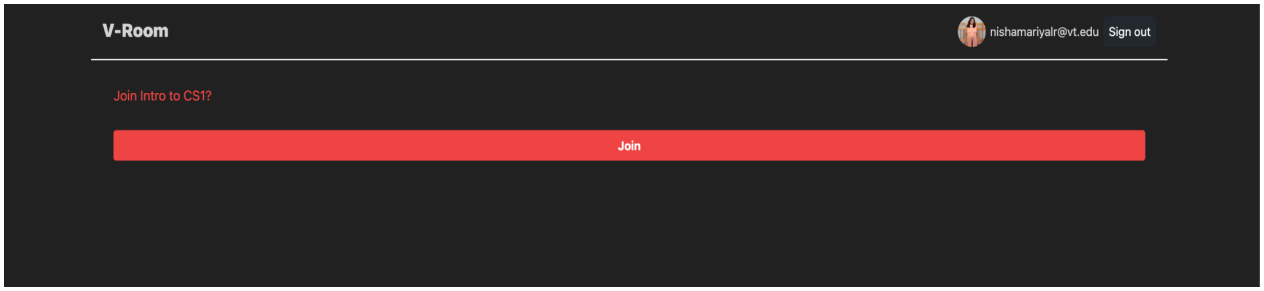

Fig 11: Invited student's view after clicking on Invite Link

2.7 View Questions in a Classroom

If the users wish to view the list of questions/answers that are posted in a particular class, they can click on the "View Questions >>" button on the bottom of the classroom page. Once the user clicks on it, V-Room will be directed to the page that shows the list of questions and answers that are posted by different people in that classroom including instructor and TA.

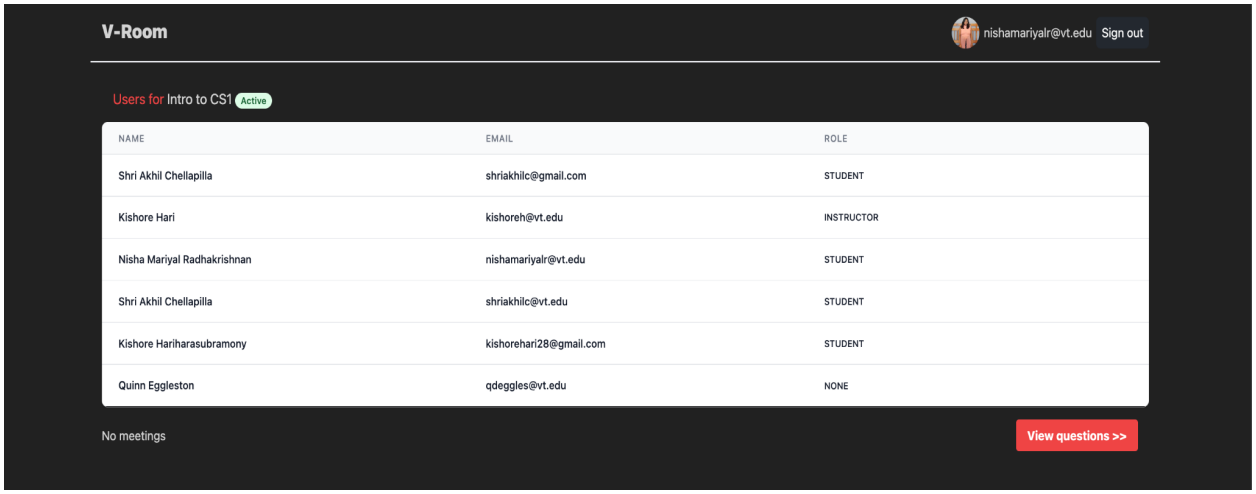

Fig 12 : Classroom page showing “View questions >>” option

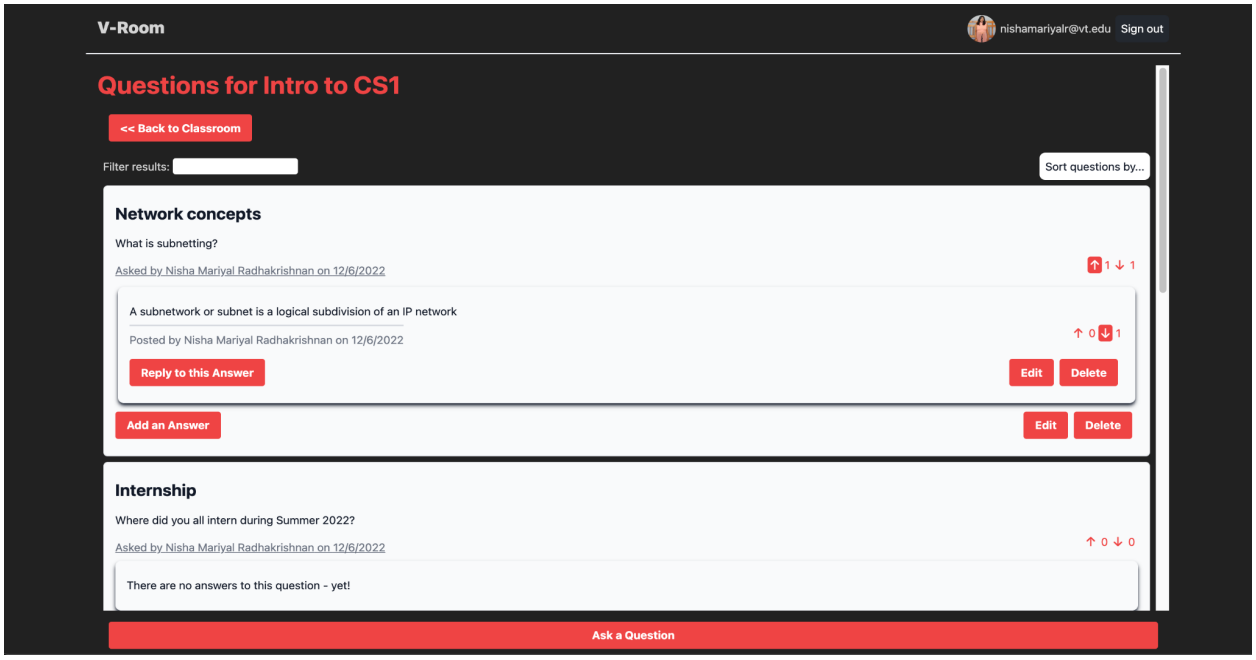

Fig 13: Questions forum of the ‘Intro to CS1’ Classroom

2.8 Ask a question outside office hours

‘Ask a question’ dialog is shown on the classroom details page. All users are able to ask questions and every question must be associated with a classroom. A user is able to edit / delete their own questions.

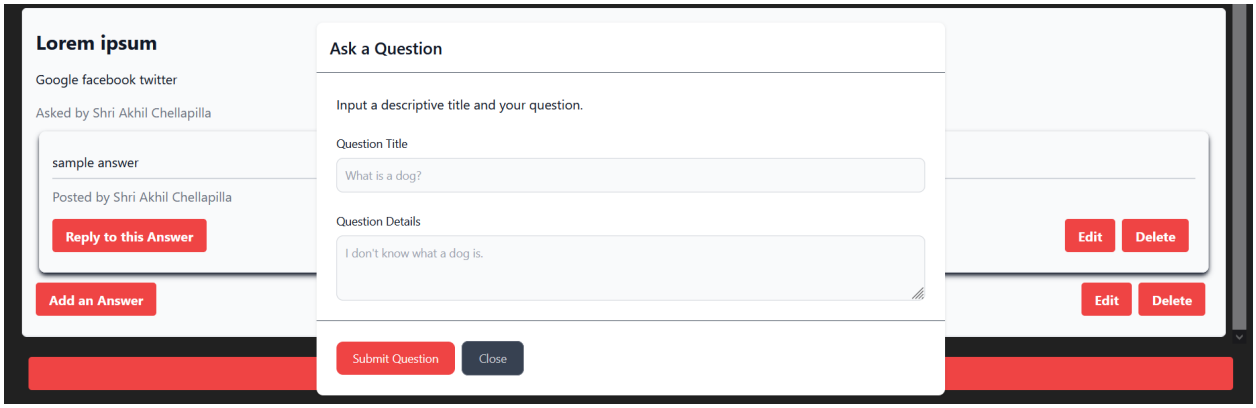

Fig 14: Modal showing Ask a Question feature

2.9 Answer the questions posted by the student

Any user can submit an answer to a question. They are also able to edit / delete their own answer(s).

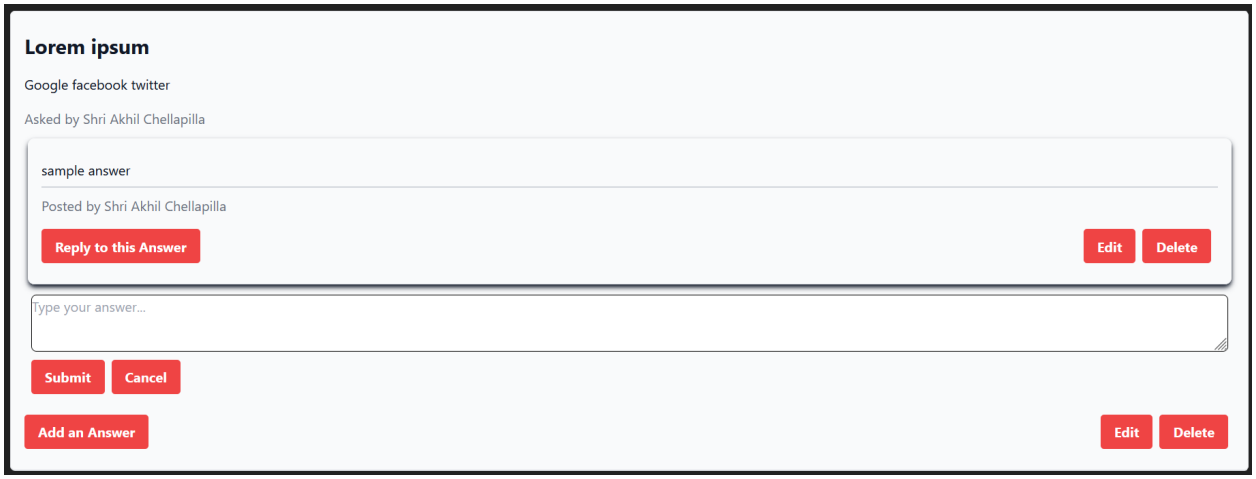

Fig 15 : Page showing text box to reply for users to reply for question

2.10 Vote on a Question / Vote on an Answer

If the user clicks on “View Questions” in a classroom that he belongs to, he/she will be able to view all the questions and their corresponding answers. Users will be able to vote on the questions as well as the answers provided by other users in the classroom. Users can either upvote or downvote on both the questions and answers. Users can also view the number of upvotes and downvotes for each question and answer in the classroom.

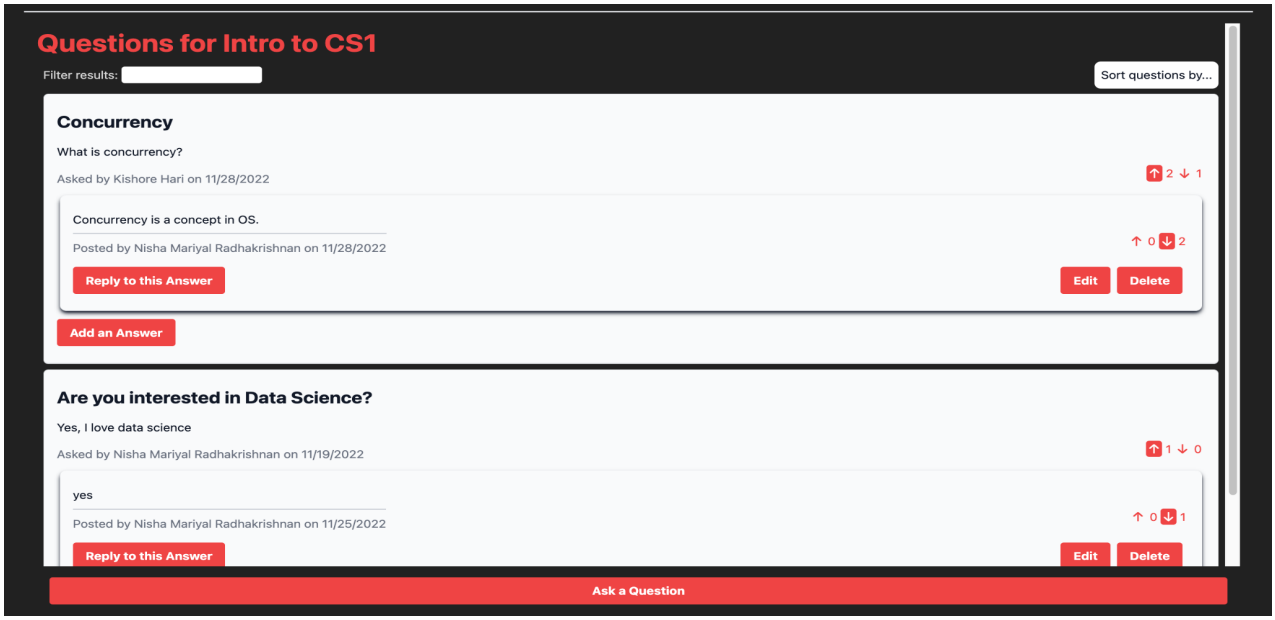

Fig 16 : Upvote-downvote feature in V-Room

2.11 Reply to an Answer

By navigating to the Questions page for any classroom, users will be able to see a “reply to this answer” button on any existing answer. Clicking “reply to this answer” will bring up a text box within which the user may type a response. They may cancel their response by clicking “cancel” or submit it by clicking “submit.” After clicking “submit,” the new response will be added to the set of responses to that particular question. If enough responses are created, to make page loading less slow, there will be a “load more” link at the bottom of one of the responses. This “Load more” link will load another few responses as shown in Fig 20.

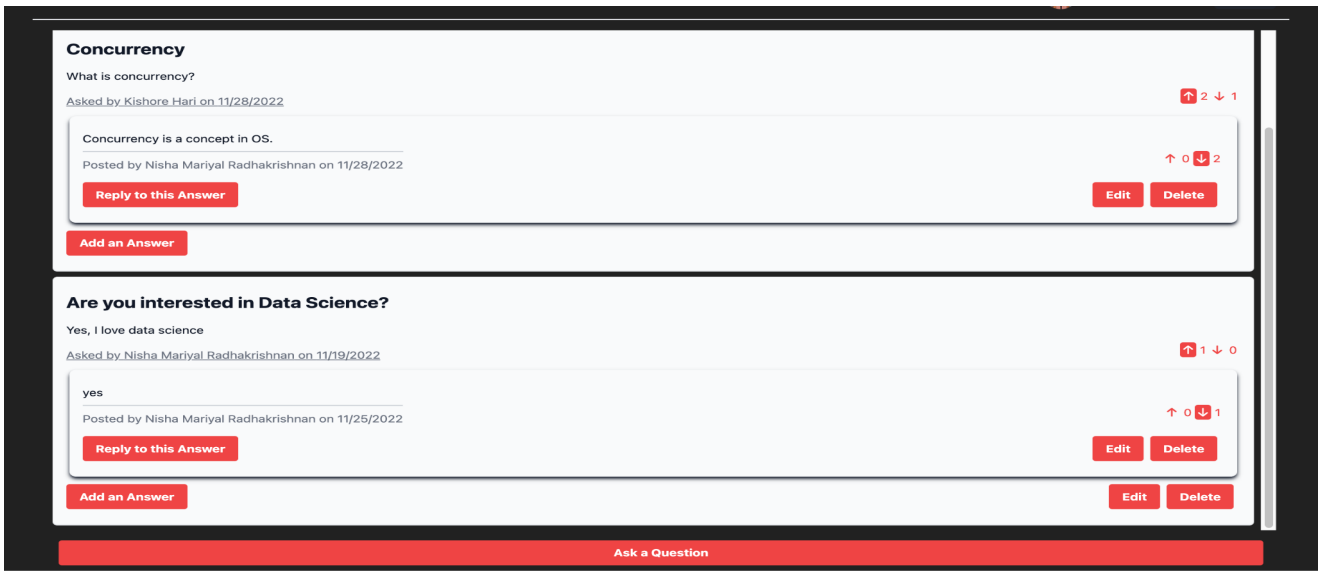

Fig 17: “Reply-to-answer” button showing up on answer posted by a user

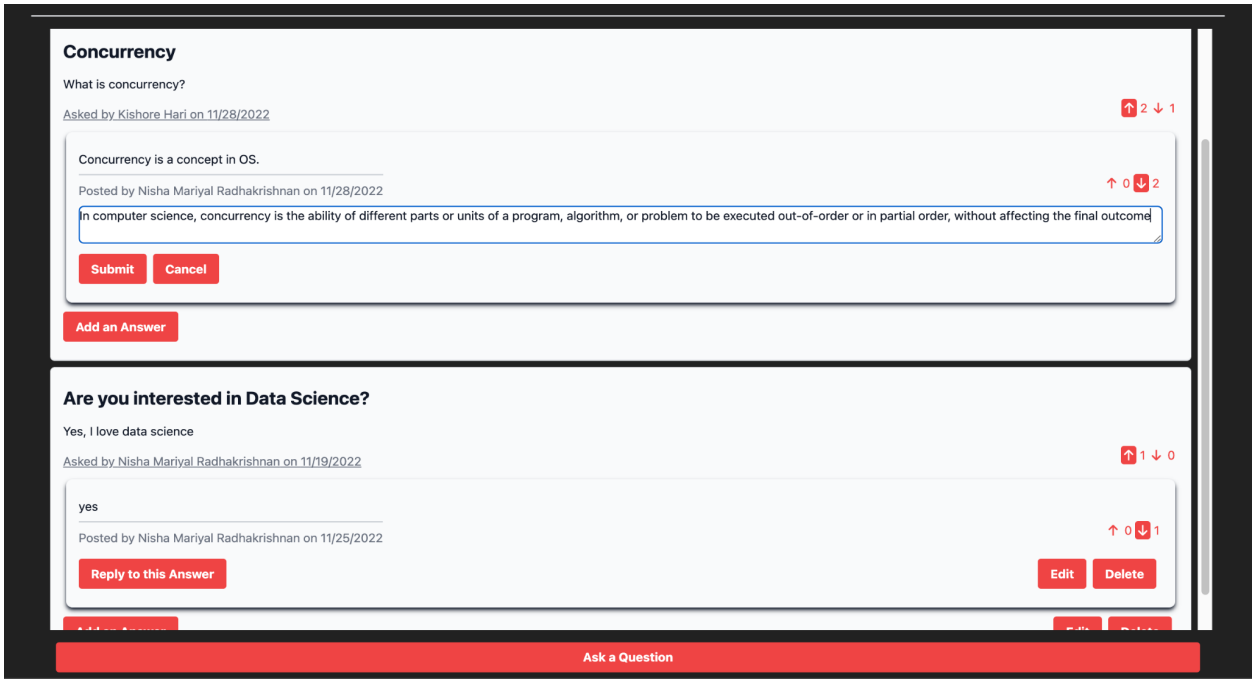

Fig 18: Adding an answer to another answer

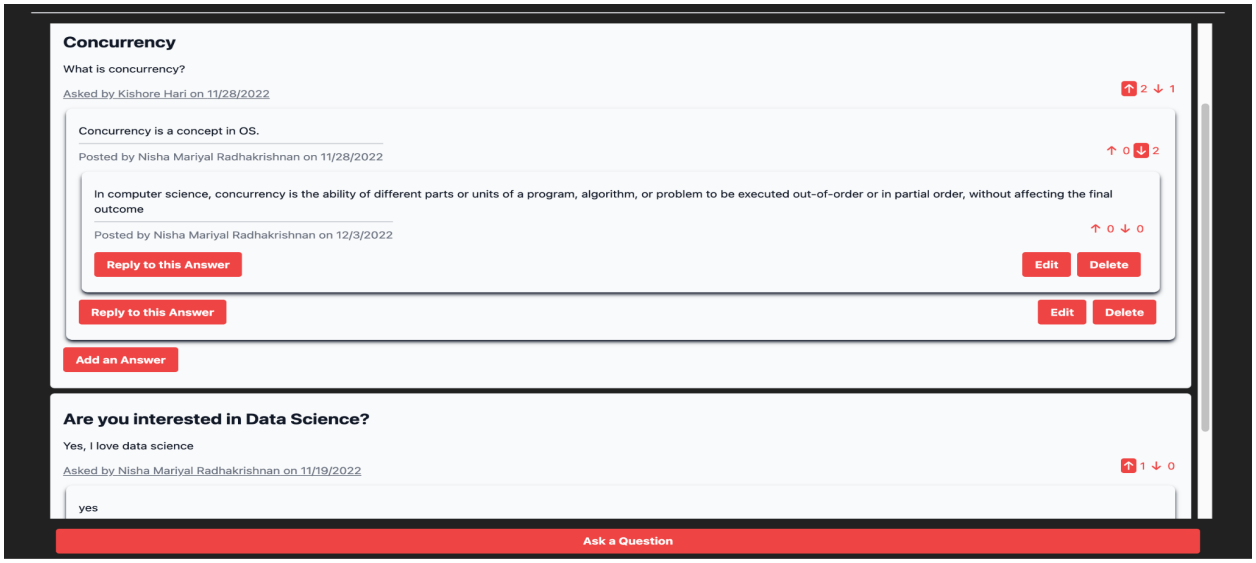

Fig 19 : Adding an answer to another answer in a thread of answers

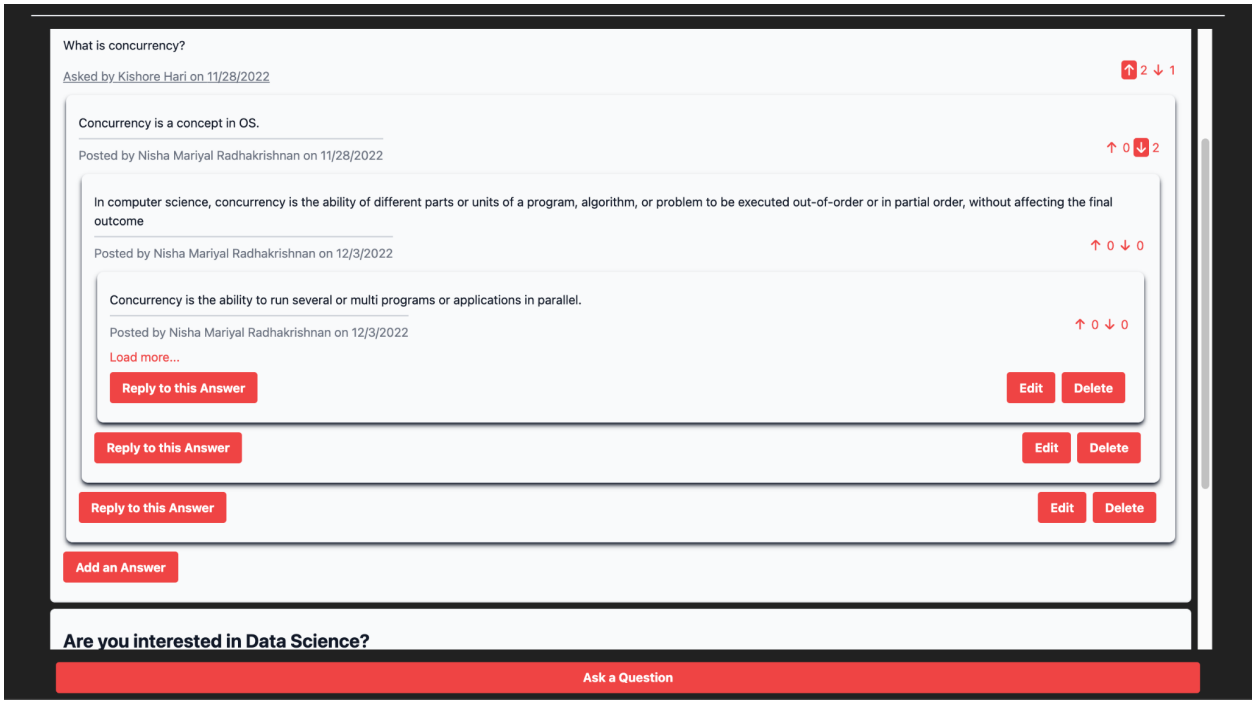

Fig 20: Users can view more responses by clicking 'Load More'

2.12 Sorting Questions

If the users of a particular classroom want to sort the questions, V-Room offers users to sort questions based on the order of Name, Date Posted and User(User who posted the question). Inorder to sort the questions, user has to click "Sort Questions by..." button on the right side of the page that shows the list of questions. Once the user clicks on the button, a dropdown will be shown where the user has to click on either Name (of the question), Date Posted or User. This will sort the questions based on the user's selection.

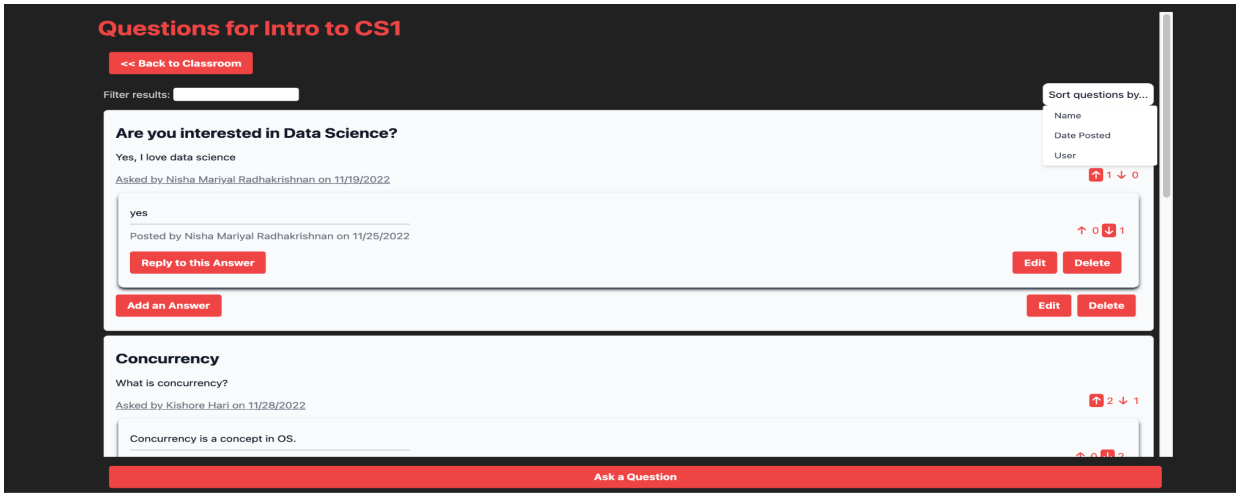

Fig 21: Page showing option to sort questions

2.13 Searching for questions by keyword

If the users want to filter or search for a question by keyword, they can go to the questions page first. After navigating to the questions page, there will be a textbox that shows “Filter results” where the users can type the keyword they want to search. This is a full text search feature where users will be provided with all the questions that match the keyword. In the given figure (Fig 22), the user is trying to search for the keyword “data science” which displays all the questions that match the “data science” keyword.

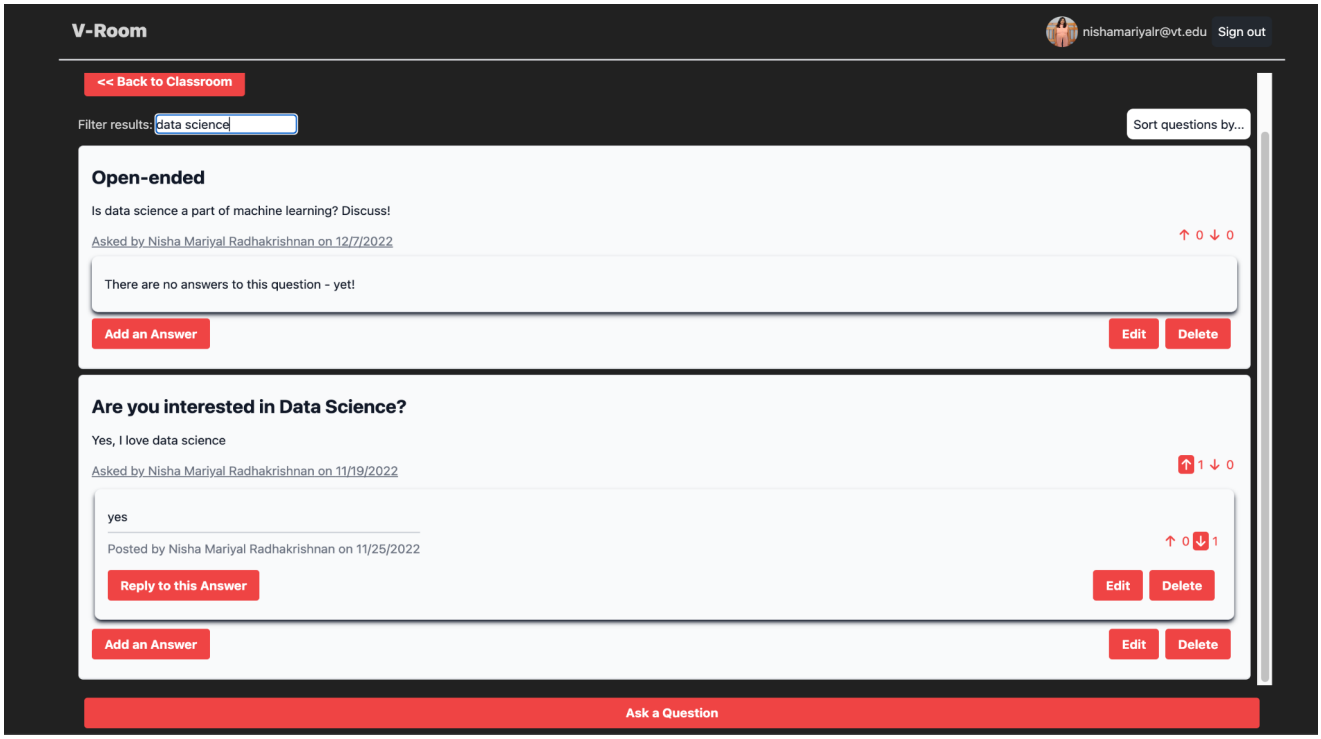

Fig 22: Page showing option for keyword search of questions

2.14 Question Links

If the user wants to view a particular question in detail, each question will have a separate link that redirects the user to a page that shows only the details of that particular question. Below each question, there will be a link that will be hyperlinked as the data showing the person who posted the question along with the data. For example, in the below figure (Fig 23), “Asked by Kishore Hari on 11/28/2022” is a link which when clicked, the user will be redirected to the detailed view of that particular question. This is also useful for users to share the question to their classmates.

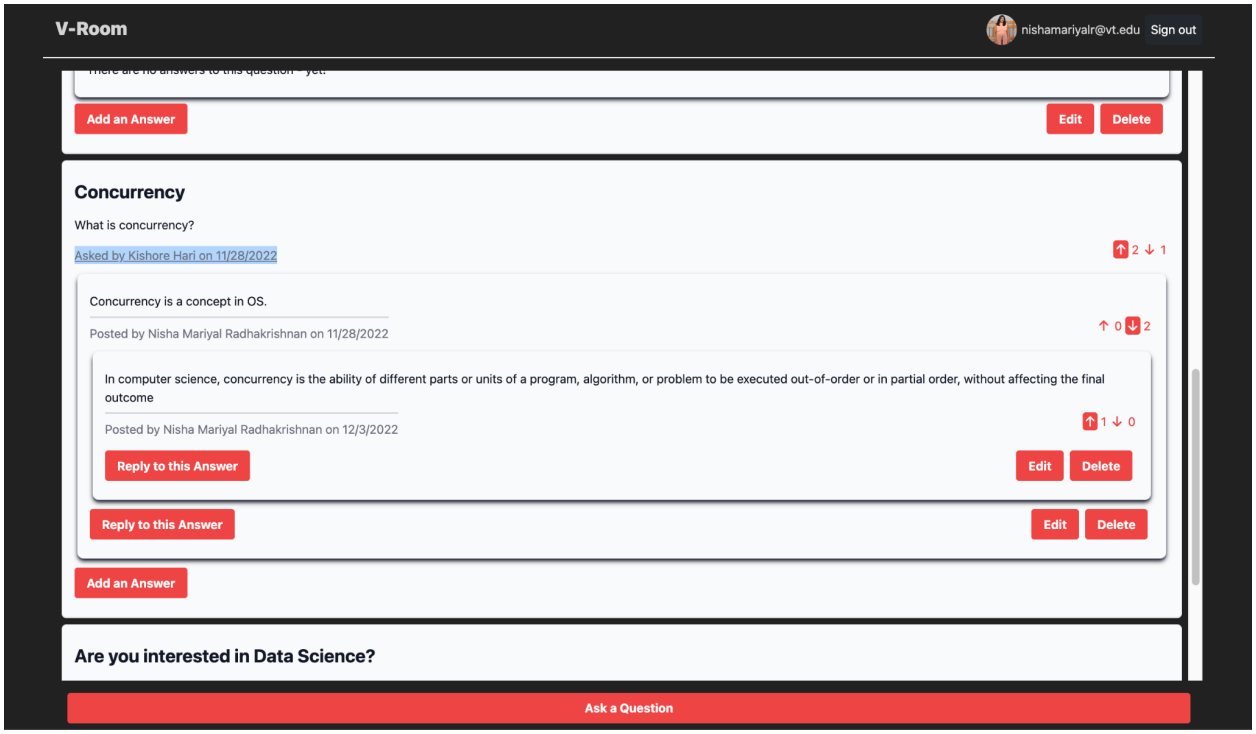

Fig 23: Page showing link for each question

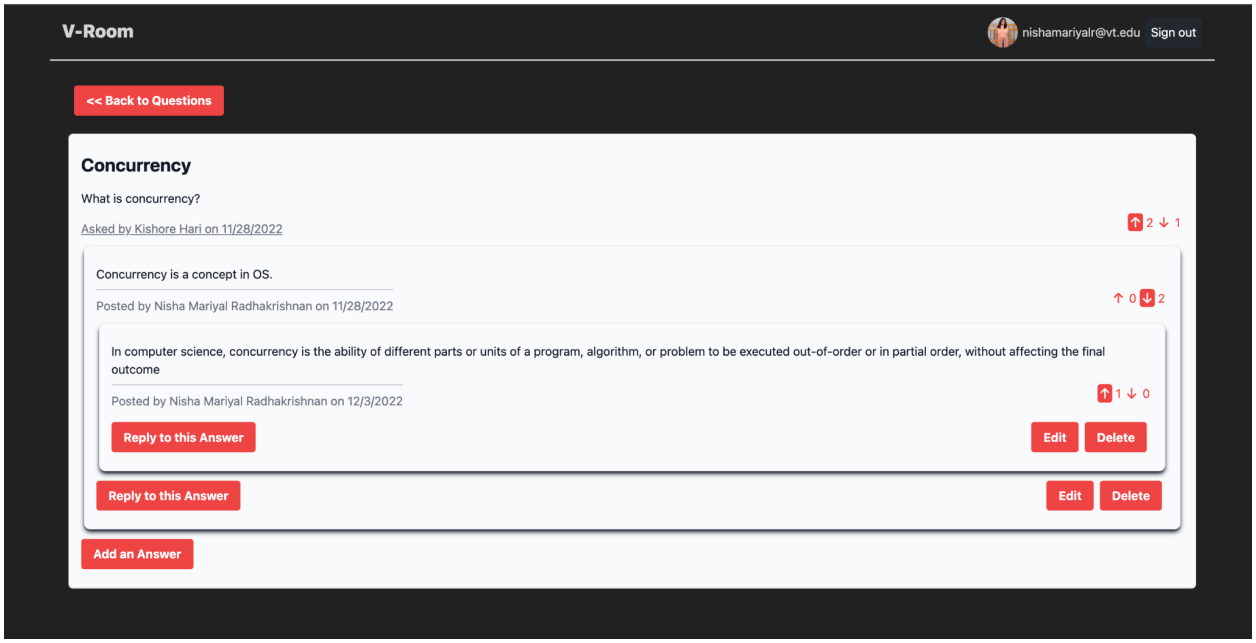

Fig 24: Page showing detailed view of a question after user clicks on the link

2.15 Create and Join a meeting

Only instructors or TAs can host a meeting. They will be able to host a meeting by clicking on the “Host a meeting” button in the classroom page. Host of the meeting can see the number of people in the meeting and the waiting room respectively. They also get a list of all names with the option to allow participants into the meeting by pressing the green check mark. Only the host can kick a participant, mute all participants or end the meeting.

Once the meeting is hosted, the host can see the number of people in the meeting and the waiting room respectively. They also get a list of all names with the option to allow participants into the meeting by pressing the green check mark. Only the host can kick a participant, mute all participants or end the meeting.

Every participant can see all others in the meeting, and the list of names. Only the host can see the list of names of those in the waiting list, others only get the total number.

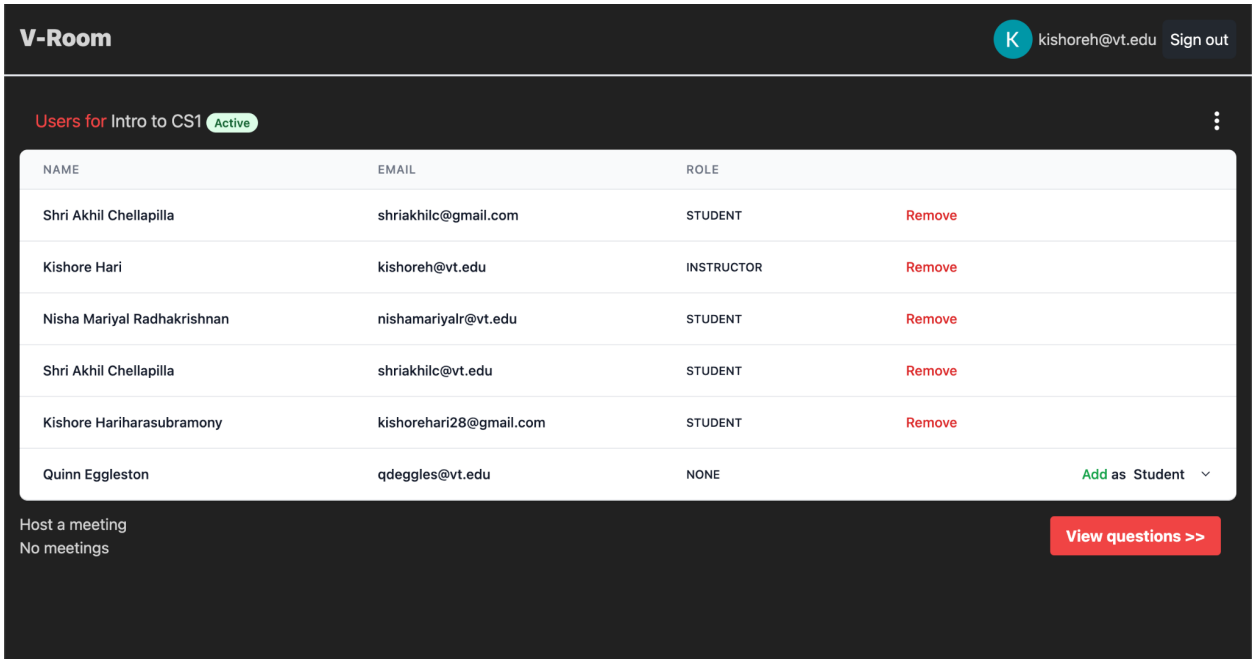

Fig 25: Classroom page showing “Host a meeting” option (Only viewable for Instructors/TA)

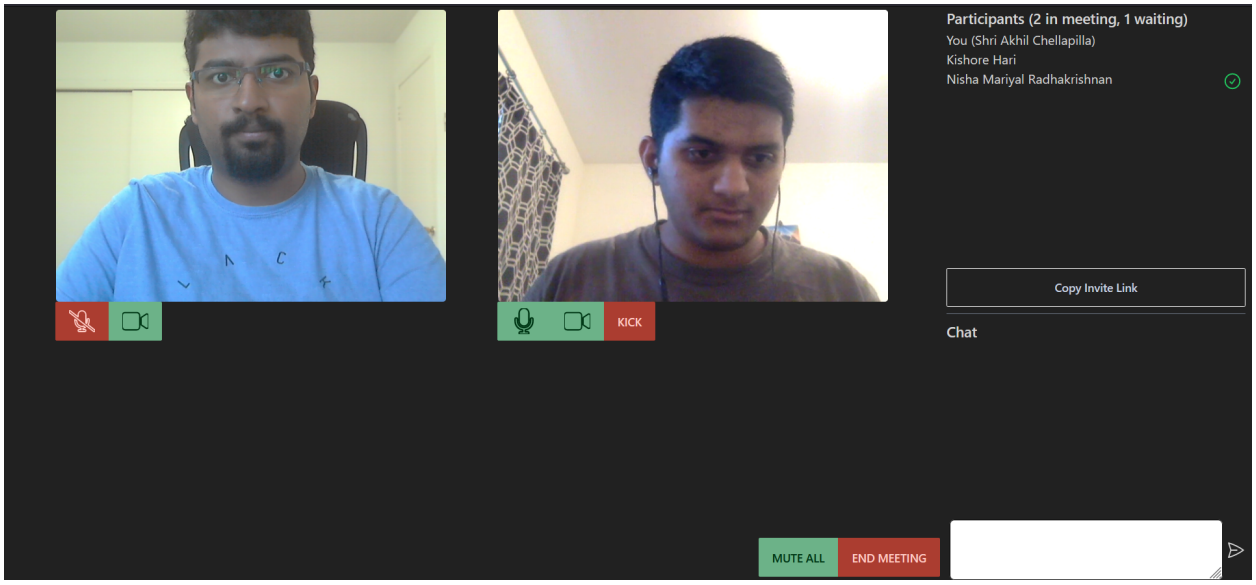

Fig 26: Host view that shows the list of participants waiting in the queue and meeting

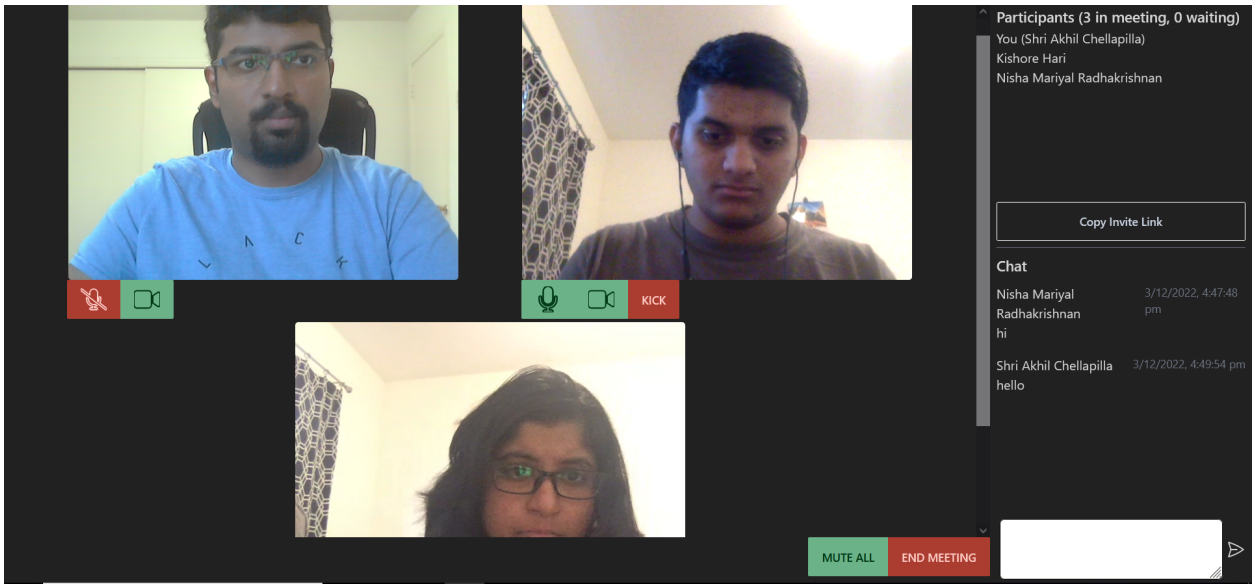

Fig 27: Page showing multiple participants with their information

2.16. Queue for a Meeting

Participants can use the chat to communicate with the host. For security reasons, they cannot see which users are in the meeting / waiting room, only the number. Once they are allowed to join the meeting, their view becomes similar to the host's below. Participant waiting in the queue can see their own position (# 1) out of the total number in the waiting room (1 person).

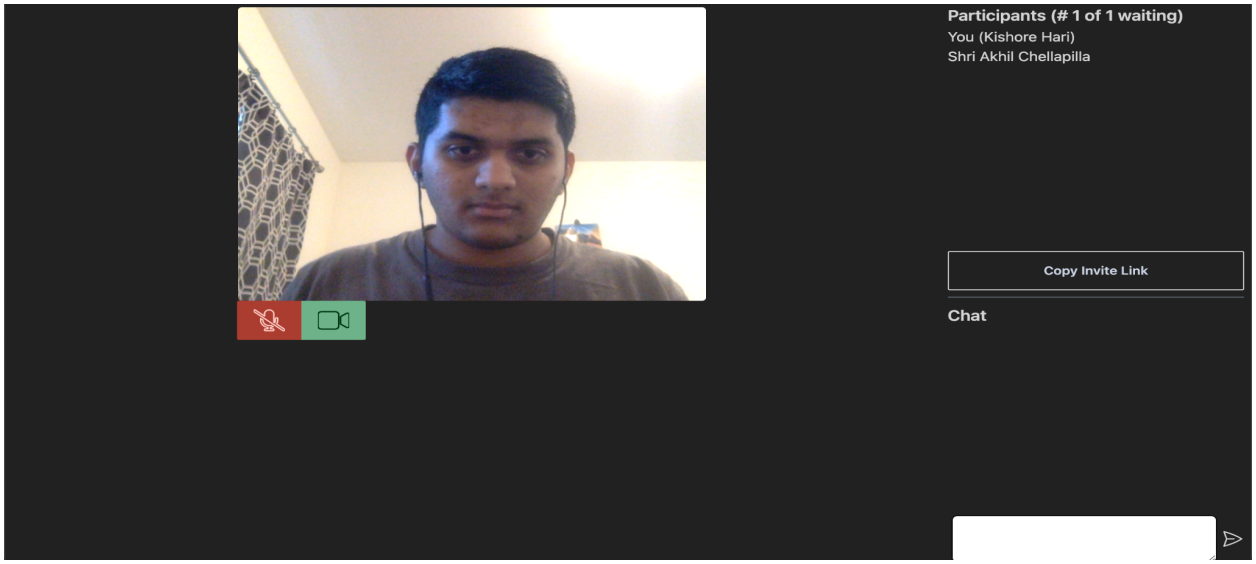

Fig 28: A participant view of meeting showing how many people are waiting in the queue

2.17. Communicate via Chat

The users can also chat in a meeting. Currently, we only allow users to post messages to everyone in the meeting and not in private. Each message is shown with a date, time and name of the person who posted it. The layout for meetings has been updated to better match standard expectations. Video streams of multiple participants can be displayed at once, and the sidebar tracks participants as well as their chat messages. Only the host receives messages from those in the waiting room, but everyone in the meeting can exchange messages among themselves

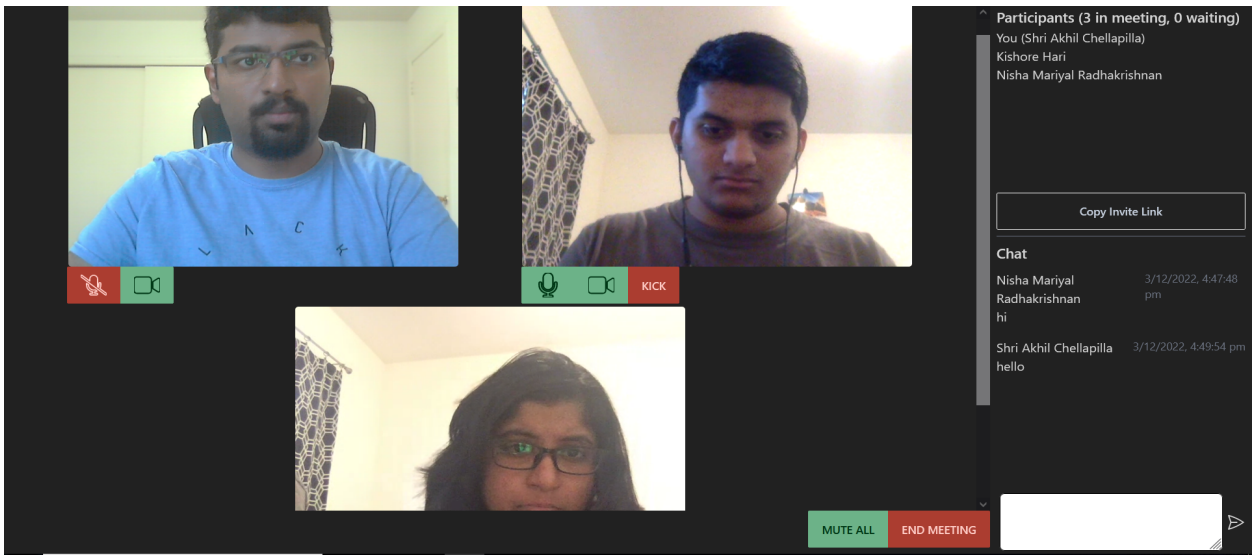

Fig 29: Meeting screen showing Chatting feature

3. Design

3.1 UML Diagrams

3.1.1 Domain Model

Description: Our domain model visually represents the important concepts in our V-Room application. In our domain model, our users are divided into three conceptual classes - Professors, TAs and Students. Each one of these may edit a minimal Profile associated with them, containing their name, preferred pronouns, and email address. Users are grouped through Classrooms, which are created by Professors. Professors may then assign Students to Classrooms, and identify TAs from those Students. Overall, Professors take a supervisory role within Classrooms. We have the invite link attribute for each classroom, using which students can join a classroom directly. The Classroom is the central concept within our design, and is used to limit users to seeing only information they should be able to access.

There is also a discussion forum linked to each classroom where students could ask questions and answer other questions as well. As a classroom can have many questions and a question can have many responses, the relationships here are also one-to-many. We have the votes attribute on both Questions and Answers. Users can vote either positively or negatively on questions and answers lodged in their classroom's forums, but not both. They only have one vote per question or answer.

Professors and TAs may open and join Meeting Rooms, while students may only join Meeting Rooms, which represent the video meetings facilitated by our platform. There is a chat message functionality to our meetings which enables participants to chat with each other. As a meeting can have multiple people chatting, the multiplicity is one-to-many. We also added a MeetingQueue item to represent the queue of Users in line to enter a classroom's office hours. This has one attribute, a group of users, which is representative of the users waiting to join the meeting.

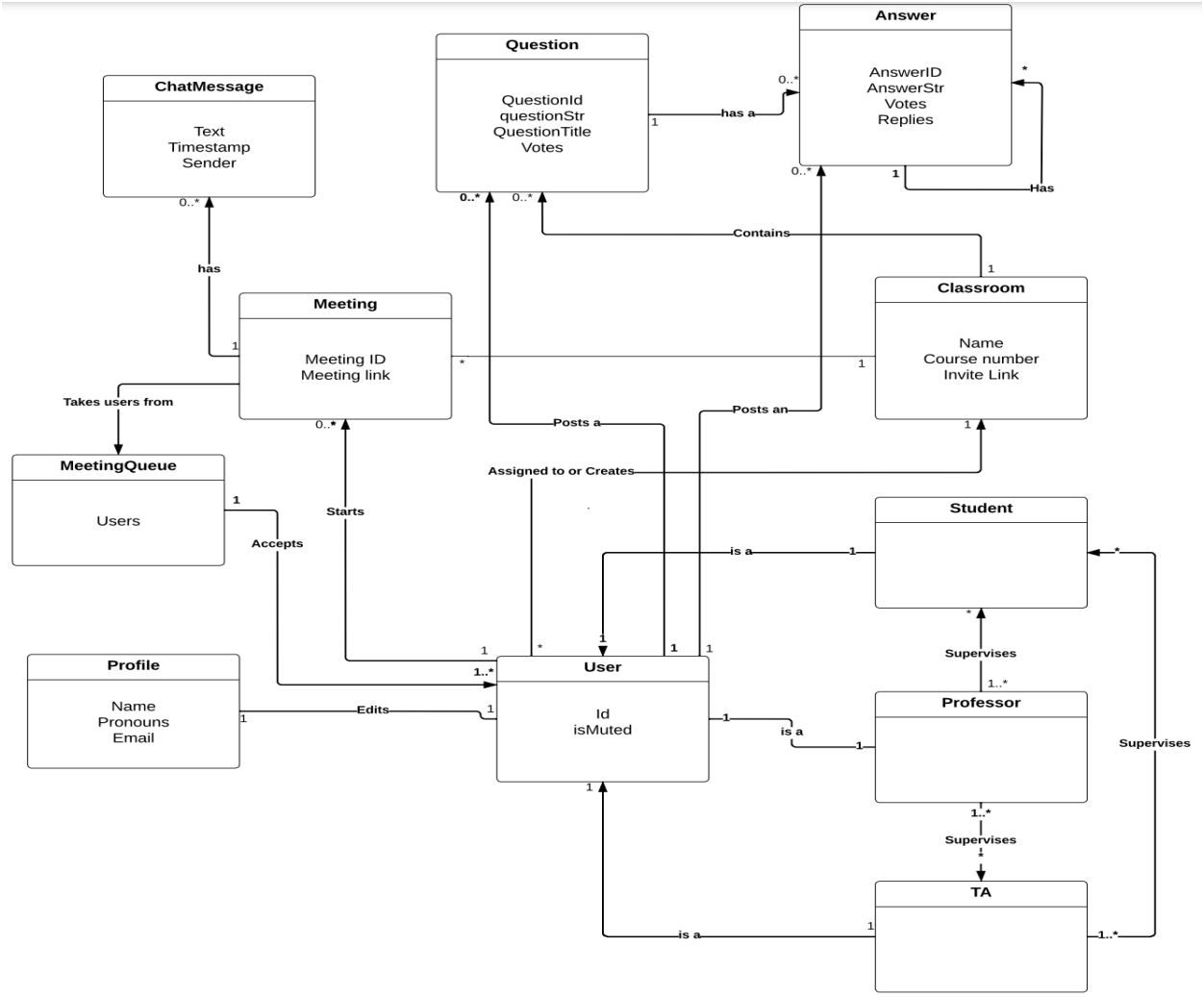

Fig 30: Domain model of V-Room

3.1.2 Interaction Diagrams

Here are some interaction diagrams that we had designed during the design phase of each sprint which helped us get a clarity on how to start working on a particular feature. We have only included the most important interaction diagrams here and not everything that we had designed.

a. System Startup Diagram

Description: In the below diagram, we illustrate what happens when V-Room is deployed. We use NextJS as the basic framework for running the server. As a first step, NextJS generates an instance of Prisma which we use for performing database related tasks. In parallel, NextJS also creates an instance of TRPC (used to make our APIs type-safe). For implementing the OAuth functionality inorder for users to

register and login to V-Room using their Google accounts, we use NextAuth framework. These are the instances/events that are getting created during the system startup. Fig 31 shows the System Startup Diagram for V-Room.

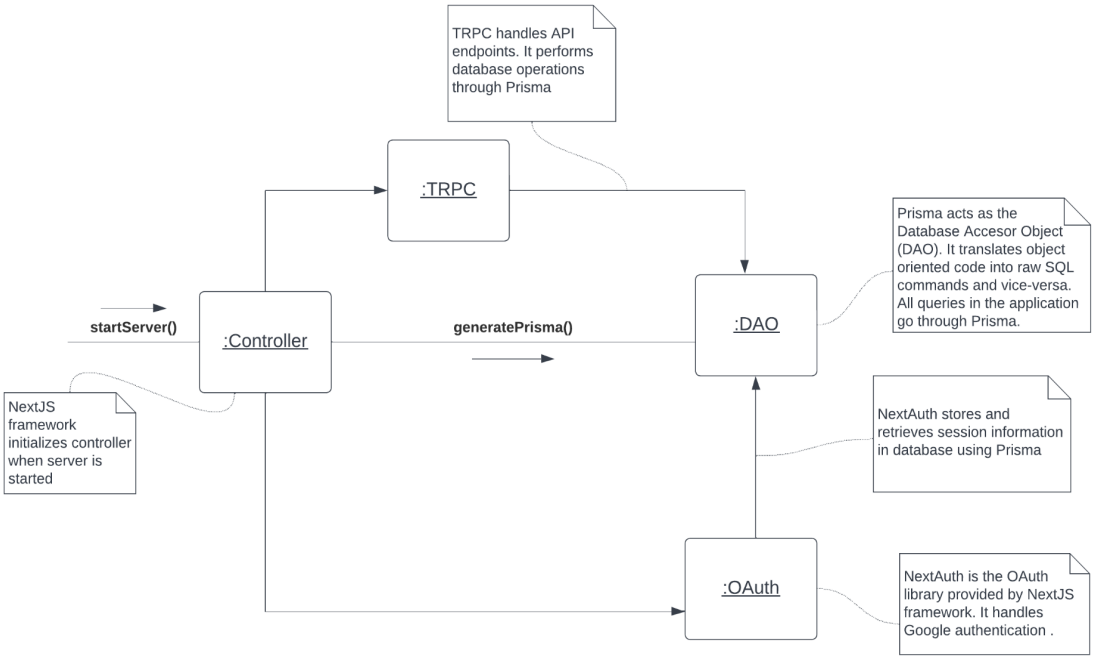

Fig 31: System Startup diagram of V-Room

b. Register and update profile

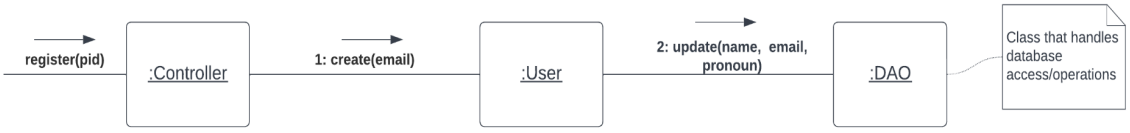

Fig 32: Collaboration diagram of register and update profile

Description:
When an individual without an existing User chooses to register with their PID, the Controller instance will create a User object with their corresponding VT email. The User object then contacts its DAO with personal information given by the user (email, along with their name and pronouns) to create a database entry for it.

c. Login to V-Room

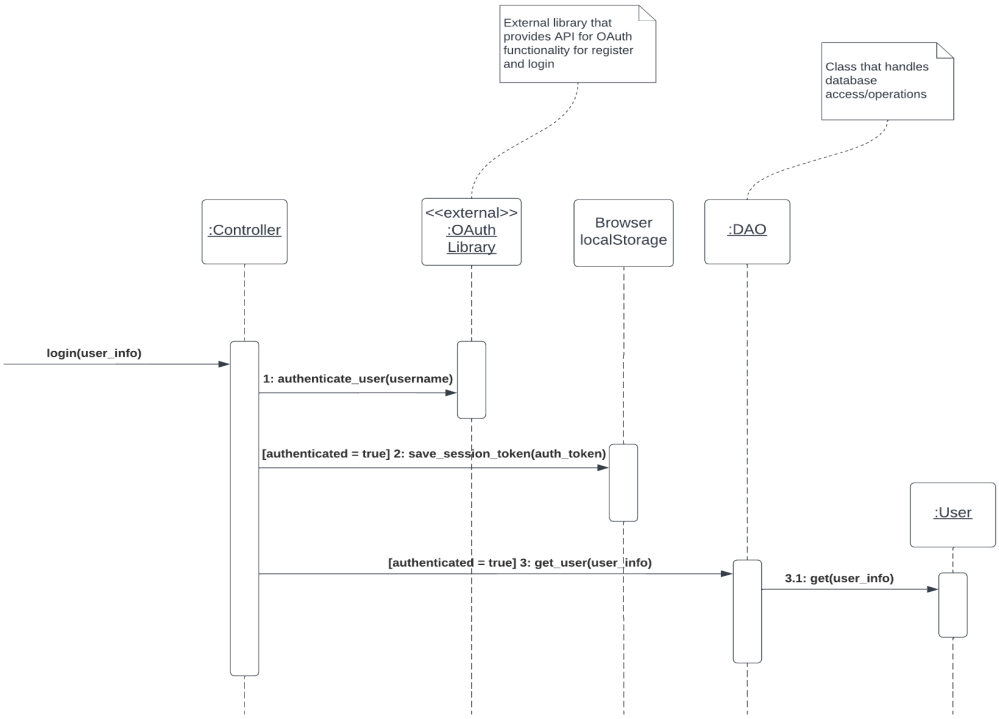

Fig 33: Sequence diagram of login

Description:
When the user clicks the login button, it triggers a system event that is handled by the controller instance. At first, a call is made to the external OAuth library to authenticate the user's credentials. If the user is already registered, the authentication becomes successful and the session is saved in the browser's local storage. All the user information is obtained from the database and the user's session is successfully created.

d. Create classroom

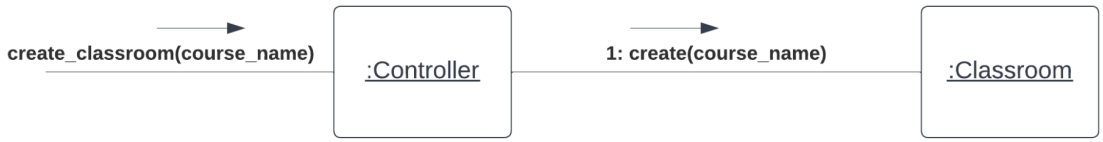

Fig 34: Collaboration diagram of classroom creation

Description:
Once the user(instructor) clicks on the “Create a classroom” button, this will create a classroom for that particular course work and post all the information about the course. This will call the create() in the Classroom class. This will create an entry in the Classroom table and save the data in the database. Once the classroom is created, the user can start adding students and TA’s for the classroom.

e. Add users to classroom

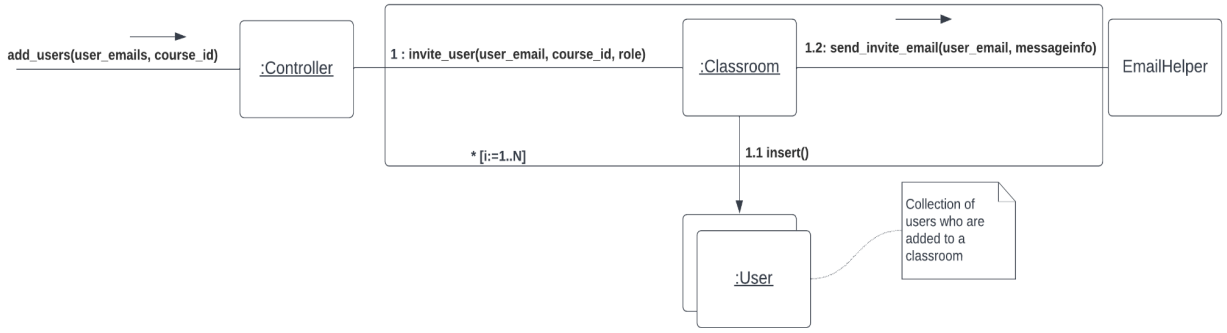

Fig 35: Collaboration diagram of adding users to classroom

Description:
When a user with the *instructor* or *assistant* ClassRole wants to add a new user to the classroom, the controller instance will add that user to the desired role in the ClassRoom and send them an invite to their email address to inform them that they have been added to that Classroom.

f. Delete Classroom

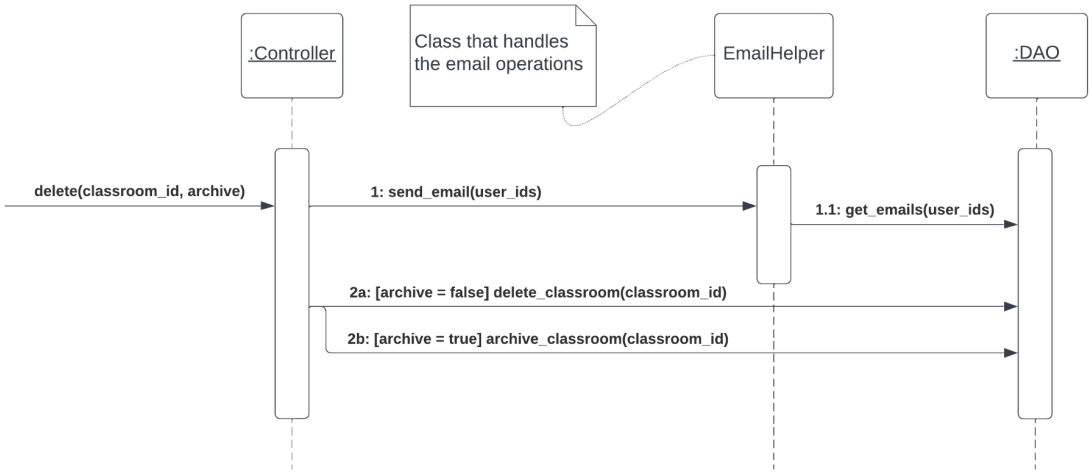

Fig 36: Sequence diagram of classroom deletion

Description:
When the user (professor) wants to delete a classroom, an email notification is first sent to all the users (students and TAs) in the classroom and for this EmailHelper class tries to get the emails of the users from the database. After this, depending on whether the professor wants to delete or archive the classroom, the classroom is deleted permanently from the database or the classroom is archived in which case it will still exist in the database.

g. Post a Question to classroom

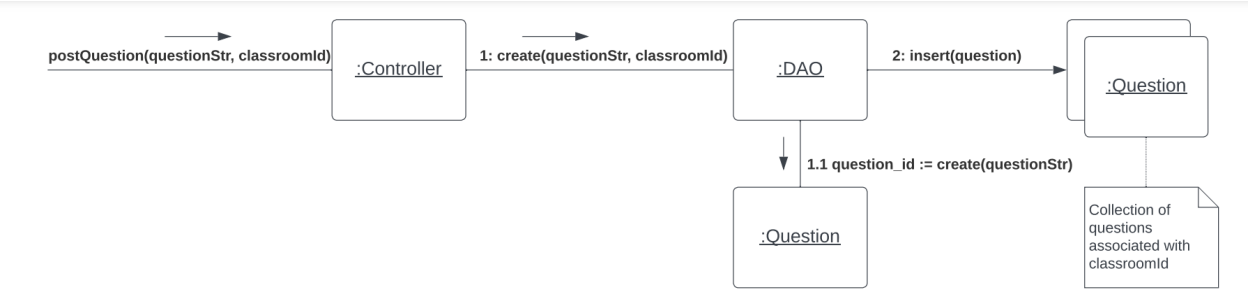

Fig 37: Collaboration diagram of posting questions

Description: Whenever the user wants to create a question, the controller class will invoke the Database Accessor Object (DAO) to insert a question into the question table using the questionStr and classroomId. The DAO internally creates an instance of Question class as well. After this, the question gets added to the collection of questions associated with the classroom.

h. Chat with participants

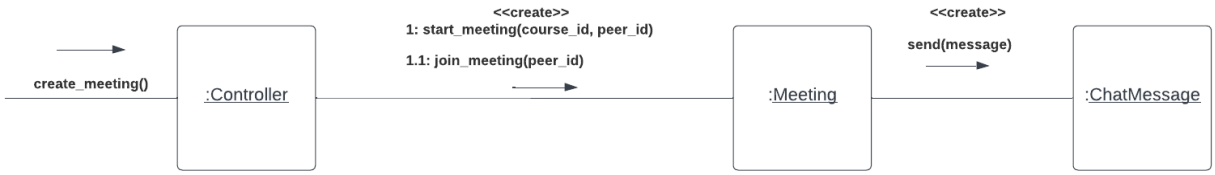

Fig 38: Collaboration diagram of chatting with participants

Description: If the user wants to chat during the meeting, we need to make sure that the meeting has started and participants have joined the meeting. For this, the controller class will first call start_meeting() function using the Meeting instance. Then users can join through join_meeting(). If the user wants to chat or post a message during the meeting, the Meeting class instance will call send() by passing the message/chat details. This will create an instance of the ChatMessage class. For every message sent during the meeting, an instance of ChatMessage class will be created.

i. Vote on a question

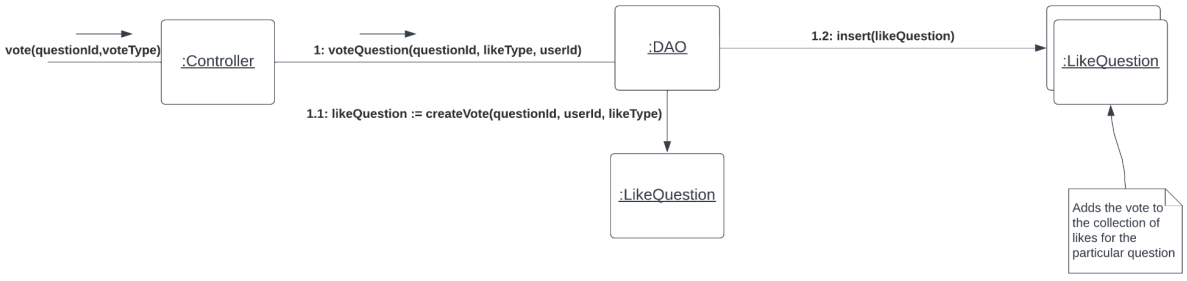

Fig 39: Collaboration diagram for voting on a Question

Description: Whenever a user wants to vote on a question in a discussion thread, the controller will receive a vote event and will access the database to create a LikeQuestion object of the appropriate type. Then the collection of LikeQuestions in

local memory will be updated to display properly on the frontend. The user can either like a question or dislike a question.

j. Video conferencing features

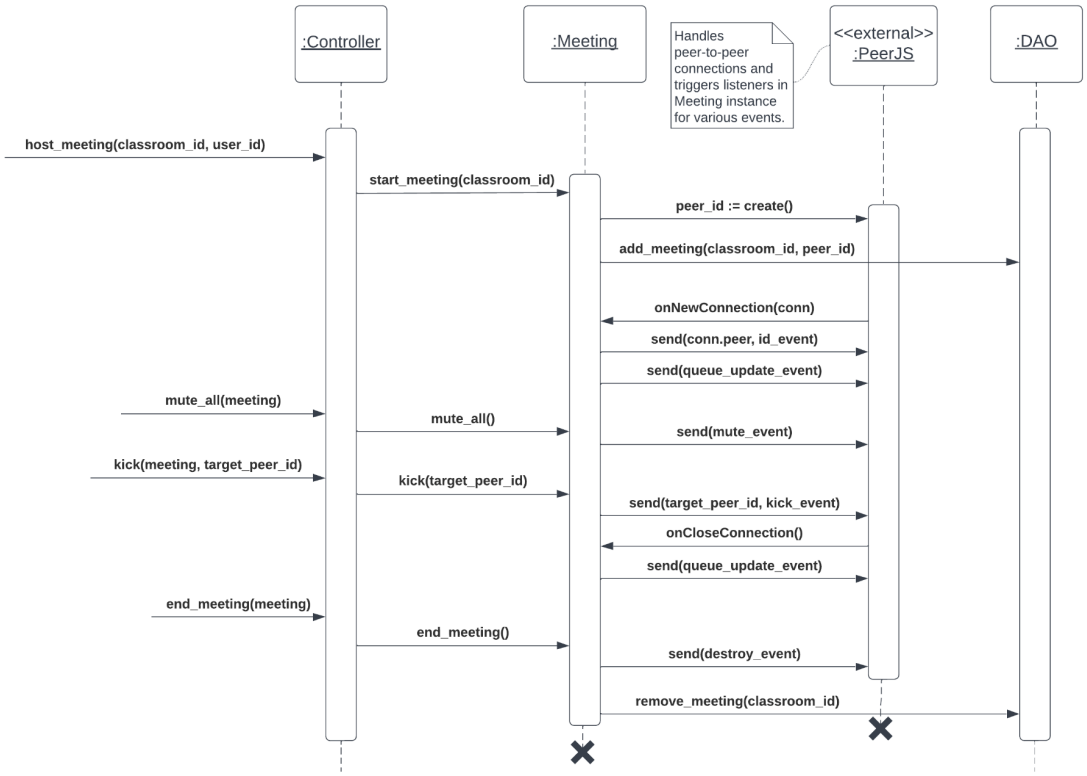

Fig 40: Sequence diagram for video conferencing features

Description: The diagram above is a possible sequence of method calls that may occur during a meeting. Since most of these actions are user-triggered, not all of them are guaranteed in a meeting. Further, there are a number of event listeners used to appropriately handle asynchronous events such as a participant connecting to the waiting room, joining the meeting with their media stream, and leaving the meeting.

The core idea is that the Host acts as a de-facto server for all peers, tracking the queue order and all associated data and media connections. Any new connection must wait for the Host to let them in before being able to establish a media connection. They also automatically connect and share media with all other existing meeting participants.

Most events change the queue in some way, and on every such change a message is sent out to peers to keep their position in the queue up-to-date. Moderation actions by the host are also achieved through custom messages, making peers leave the meeting or mute themselves as necessary.

If the host ends the meeting or leaves by closing the tab, all other participants are kicked out.

k. Reply to an answer

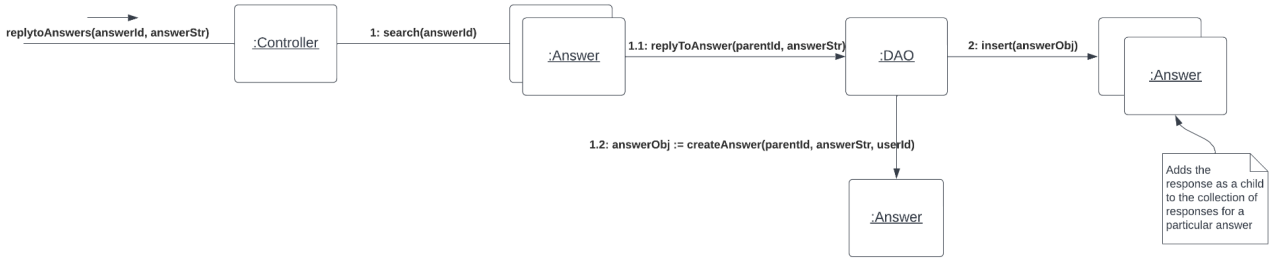

Fig 41: Collaboration diagram for replying to an Answer

Description: Whenever a user wants to reply to an answer in a discussion thread, the controller class will search for the answer in the collection of answers and fetch its id. Using this as the parentId and the answerString provided by the user, the Database Accessor Object (DAO), internally creates an instance of the Answer class. After this, the answer gets added to the collection of answers associated with the particular answer in the thread.

3.1.3 Design Class Diagram

Description: There are two main sections of our Design Class Diagram - the meeting section and the forum section. These two are united by the Classroom model, which handles participation in both through associations with users. In our structure, Users may be associated with any number of Classrooms, with a particular role designated by the UserOnClassroom relation and the ClassRole enum. Users may then submit Questions and Answers, as well as vote on Questions and Answers within the Classroom, represented by their respective models. Though it will not be covered exhaustively here, Users cannot perform actions that stray outside their permissions - for instance, they cannot delete the classroom if they are only Students.

On the Meeting side, most features are implemented by means of messages shared between peers on the data channel. Thus, we created interfaces for the payload and each type of event to ensure type-safety while sending and receiving the data. Multiple event listeners are used to handle scenarios such as new peers joining or leaving the meeting, allowing everyone to have the latest set of information.

The ParticipantInfo interface stores the name, data connection and media connection for a specific peer, and is used to maintain an ordered Map of participants on the Host's side. All other members in the meeting get updates from the Host for their display needs, but they do not have access to those in the waiting room. New entrants receive a list of peer IDs from the host, and then establish data and media connections with all existing participants.

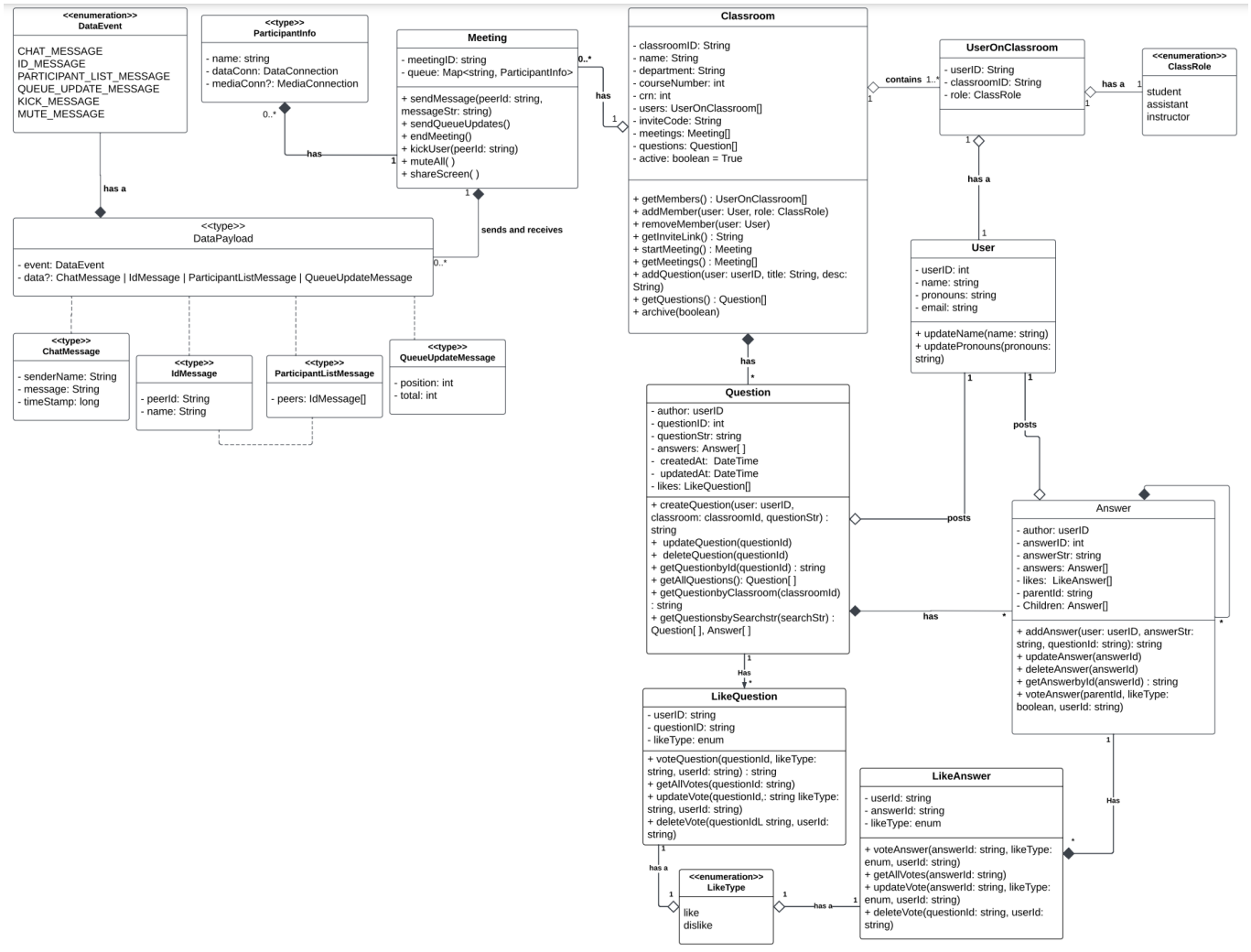

Fig 42: Design Class Diagram of V-Room

3.2 Technical details and Tools Used

Our schema design was centered around the concept of a classroom, in order to remain in line with its educational objectives. On the frontend, we attempted to make our design simple and straightforward to use. Using Google OAuth for logging in to V-Room simplifies the registration and login process, and we attempted to avoid overcrowding and emphasize color contrast in our pages. Similarly, our meeting room layout design has minimal distractions. Technical decisions for the meeting-related things were dictated by a peer-to-peer library. We carefully chose our technical tools in order to minimize errors during product use and avoid leaking extraneous information. This is exemplified in the type safety of Typescript and tRPC's querying features.

3.2.1 Frontend

Our product frontend was created with NextJS and React, using Typescript. NextJS provided an easy-to-use routing framework, allowing us to take advantage of URL parameters to adapt page contents as necessary. React was used to componentize our frontend code, which made the development process more flexible and adaptable.

3.2.2 API

Our API endpoints were implemented using tRPC, a library that allows API calls to be made without relying upon vanilla Javascript's Fetch API. It also allows Typescript to interpret data returned from queries as types when working with an ORM like Prisma. This allowed us to guarantee type safety and harden our database against invalid requests, while making the development process more comprehensible on the frontend.

3.2.3 Database

Our data is hosted on PlanetScale, a serverless platform which uses MySQL internally. We interacted with the MySQL database through the Prisma ORM, with which we could push and pull schema updates and easily test out new designs.

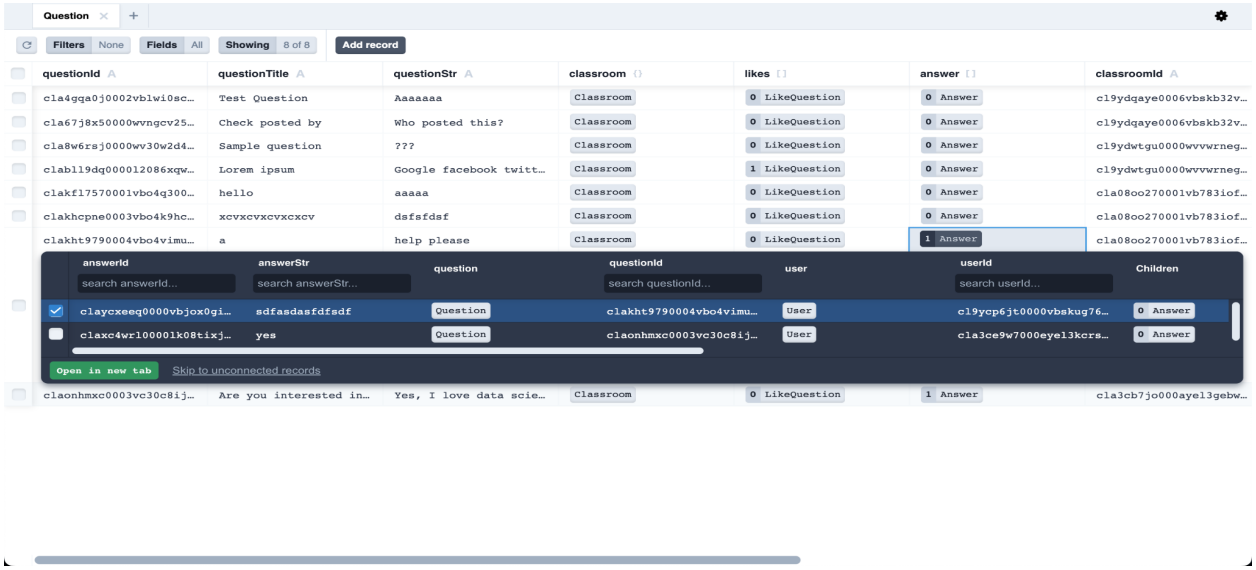

Fig 43: Prisma Visualization of “Question” Table

3.2.3 Meetings

PeerJS is our main tool for creating our meeting room structure. We chose PeerJS because it does not explicitly require a central server setup, which would have been additional work on top of our already large body of tasks. In our design, users each connect to each other upon joining a meeting, thus forming a web of connections through this peer-to-peer system.

3.3 Uniqueness of V-Room and how it is different from others

V-room is a meeting platform integrated with a classroom and A discussion forum structure, specialized to a particular purpose - that of facilitating smooth teacher assistance through office hours and answering questions. While many platforms implement education-adjacent features, no platform sets out with this explicit goal, and thus combines this set of features. While it’s true that some of the more established video platforms exceed V-room in terms of overall feature robustness, the core features of V-room are still sufficient to differentiate it from the competition.

Using V-Room’s video conferencing feature, students can know how many participants are ahead of them in the waiting room and can get an estimate how long it would take for them to meet with the instructor or the GTA. Similarly, instructors and GTAs can also know how many students are in the waiting room and plan for an extended meeting or offer additional office hours to students. This key feature of V-Room is absent in existing video conferencing applications like Zoom.

Additionally, on V-Room, a student can discuss questions about class content and then hop directly into an office hour meeting smoothly, without needing to context switch between Zoom or another application, say, Piazza. The most context switching ever necessary while working with V-room is that of changing to a different tab.

4. Retrospection

4.1 What went well and what went wrong

Generally, one of the best things about our three sprints was communication. Our teammates usually did not hesitate to communicate about issues, bugs, and other relevant subjects throughout our work, and this allowed us to build a better product overall. Unfortunately, we took a very optimistic view of what we were able to get done, and thus small features also ended up being cut towards the start of the project. This was also contributed to by the exit of one of our group members about midway through the project.

Overall, our division of work throughout the project was effective. Though in the first sprint, this was a little rough, and contributed to many of the slowdowns in our first burndown chart, we improved upon this significantly in the second sprint and maintained those same policies through the third sprint. In the end, this was a major factor in allowing us to complete all of our major features.

4.2 Lessons learned and skills gained from building V-Room

We also learnt a lot of lessons while working on this project. One of the main lessons that we learnt was to effectively communicate with our team members. This was really helpful to get a lot of input from other team members whenever one of us was stuck in a particular situation.

Also, splitting up the frontend and backend tasks were really helpful as we were able to spend more time on each of the tasks individually. We also learnt to prepare for the worst and coped up with it efficiently as two of our team members had to leave this course in the middle of the semester. We still were able to achieve most of the tasks that we planned.

Over the course of this project, we gained a lot of knowledge and skills about the different technologies that we worked on. First, we were able to gain deep knowledge about Prisma, an Object Relational Model (ORM) we have used to interact

with the database and perform database operations. As a result of using Prisma, designing the database schema became easy and less time consuming. It was also very easy to even implement some complex features like “Replying to a particular answer”, “Upvotes and downvotes for question and answer” etc.

None of the team members have worked on implementing a google authentication before in any of the projects. As a result of using NextAuthJS, a library provided by the NextJS framework, it was easy to integrate Google authentication in V-Room. Since few of the team members were not familiar with frontend frameworks, with this project, they learnt ReactJS and Tailwind CSS which were very helpful in building the UI for V-Room.

One of the most difficult features to implement was the meeting related feature using PeerJS. Even though it was difficult in the beginning, we gradually learnt from different online resources and implemented most of the features that we planned to implement. V-Room gave us a great learning experience and we hope to use this experience in our future projects.

4.3 Actions we could have taken to avoid the problems we encountered

A significant thing we could have done to avoid our problems was to communicate more throughout the first sprint and to furthermore plan for worse outcomes. Due to a group member’s exit in the first sprint, we had to significantly rework our video chatting features, and our API endpoints needed to be overhauled during the second sprint due to previous misunderstandings. Overall, some of the failures in this project were failures of foresight and planning.

4.4 How can we make V-Room better?

We could improve V-room on the technical side by doing a review of our schema, API structuring, and front-end optimization. This should improve user experience by streamlining the user flow and fixing edge-case bugs, and could also make development smoother in the future. However, this would likely be the least impactful improvement from a user’s point of view.

V-Room can be made better by having some additional video-conferencing features to enable better communication during virtual office hours. One such feature would be to enable participants to share screens during meetings, sparing the participants the time and effort of having to send files, documents, or videos and make communication smoother.

The other feature that would make V-Room better is participants being able to submit a question while in the waiting room during an office hour meeting. This is a feature that we had planned to incorporate during the design phase but due to reduced capacity we could not complete the user story. By being able to submit questions while in the waiting room queue, the TA or the professor could group students together if they have similar questions and thus save a lot of time during the office hour meetings enabling them to attend to more students.

Additionally, if we choose to pivot from a peer-to-peer to centralized server based meeting infrastructure, it might allow for additional moderation tools and features that are currently harder to implement.