

TutorsRUs

Team 25 TutorsRUs

Design Document

By Malia Marquez, Shreya Suri, Evans Tang, and Jharna Suresh

Index

a. Purpose	3
i. Functional Requirements	3
ii. Non-Functional Requirements	6
b. Design Outline	7
i. High-Level Overview	7
ii. Sequence of Events Overview	9
c. Design Issues	10
i. Functional Issues	10
ii. Non-Functional Issues	12
d. Design Details	14
i. Class Design	14
ii. Description of Classes and Interactions Between Classes	14
iii. Sequence Design	16
iv. UI Mockups	21

Purpose

Finding a reliable tutor and connecting with fellow students who are struggling with similar class material can be a daunting task. There are some resources such as WISP that focus on science classes, but not many tutoring networks are advertised on campus that offer services for all types of classes. While students can also find people offering to tutor for specific subjects on Reddit or linked on Brightspace, a list of names lacks the personal connections and individual customizations that TutorsRUs culminates for each user. For example, a prevalent issue for international students whose first language is not English is finding a tutor who speaks their primary language in order to better understand the material. TutorsRUs will allow students to search for available tutors with this specific filter and many more.

Additionally, tutors that Purdue students find on tutoring forums might not even have real experience with Purdue classes. Our website would streamline the process of connecting students with qualified student tutors and make it easy for students to schedule a time and place with rated tutors. Many students may have had bad experiences with specific tutors because of a lack of communication or knowledge. Tutors can also have bad experiences teaching students because of a lack of willingness to learn or refusal to pay on time. TutorsRUs allows students and tutors that have had tutoring sessions together to rate each other so other users can make an informed decision before connecting with them. Beyond tutoring, students have the opportunity to form a support system by connecting and creating study groups with other students in their classes. By utilizing Google Calendar, students can schedule a tutoring session with someone they have connected with as well as group study sessions with other students in their class.

Functional Requirements

1. User Account
 - a. As a user, I would like to be able to

- i. create and register my account
- ii. log in to, log out of, and manage my account
- iii. reset my password
- iv. deactivate my account
- v. follow other users' accounts
- vi. utilize a “mailto” link to email other users directly from their profiles
- vii. be automatically logged out of my account after a certain amount of time
- viii. (If time allows) have a tutorial/help page when I log onto the website for the first time that walks me through how to navigate TutorsRUs

2. Customizations

- a. As a user, I would like to be able to
 - i. have an account settings page to alter my courses and preferences after creating my account
 - ii. Upload my transcript to be verified as a tutor
 - iii. list my courses when creating my account
 - iv. list my preferred language spoken and contact info on my page
- b. As a student, I would like to be able to list the courses I am seeking tutoring in
- c. As a tutor, I would like to be able to
 - i. list the courses I am offering tutoring in so that students can easily find me
 - ii. specify what semester I took a course in (ex. Fall 2021) and with what professor

3. Scheduling

- a. As a student, I would like to be able to
 - i. schedule group study sessions with other classmates via google calendar
 - ii. schedule or join group tutor sessions via google calendar

- iii. easily pay my tutors post tutoring session
- b. As a tutor, I would like to be able to
 - i. track a student's attendance
 - ii. quickly access my schedule via google calendar
 - iii. create group tutoring sessions
- 4. Discussion Boards
 - a. As a user, I would like to be able to
 - i. post questions on a discussion board for classes I am taking or tutoring for
 - ii. post documents such as practice problems, study guides, notes, or solutions to class material if permitted by my professor
 - iii. report posts that violate Purdue University guidelines
 - iv. reply to another user's post in a discussion board
 - v. (If time allows) have all posts filtered out so I don't have to see inappropriate words
 - vi. (If time allows) upvote a post on a discussion board.
 - b. As a student, I would like to be able to
 - i. access discussion board pages for classes that I am taking
 - ii. (If time allows) tag users in a discussion board post
- 5. Search for tutors
 - a. As a student, I would like to be able to
 - i. search for all available tutors for my classes
 - ii. find tutors that are teaching multiple classes I am taking
 - iii. filter available tutors that:
 - 1. Had my professors previously
 - 2. Are within my budget
 - 3. Speak my preferred language
 - iv. sort available tutors by rating and price
 - v. find specific tutors by name
 - vi. share a tutor's individual profile link with a classmate
- 6. Rating system

- a. As a student, I would like to be able to
 - i. rate the tutors I have connected with based on their helpfulness and response time
- b. As a tutor, I would like to be able to
 - i. see a student's rating
 - ii. rate the students I have connected with based on their willingness to learn and promptness of payment

Non-Functional Requirements

1. Performance

- a. As a developer, I would like the application to
 - i. start up in 5 seconds.
 - ii. function properly without crashing.

2. Security

- a. As a developer, I would like to
 - i. Limit one account per email
 - ii. Use an authentication token to automatically log a user out after a certain amount of inactive time

3. Usability

- a. As a developer, I would like
 - i. an easy-to-use interface for simple and quick navigation
 - ii. an aesthetically pleasing and functional design
 - iii. a site that is accessible from all devices and browsers

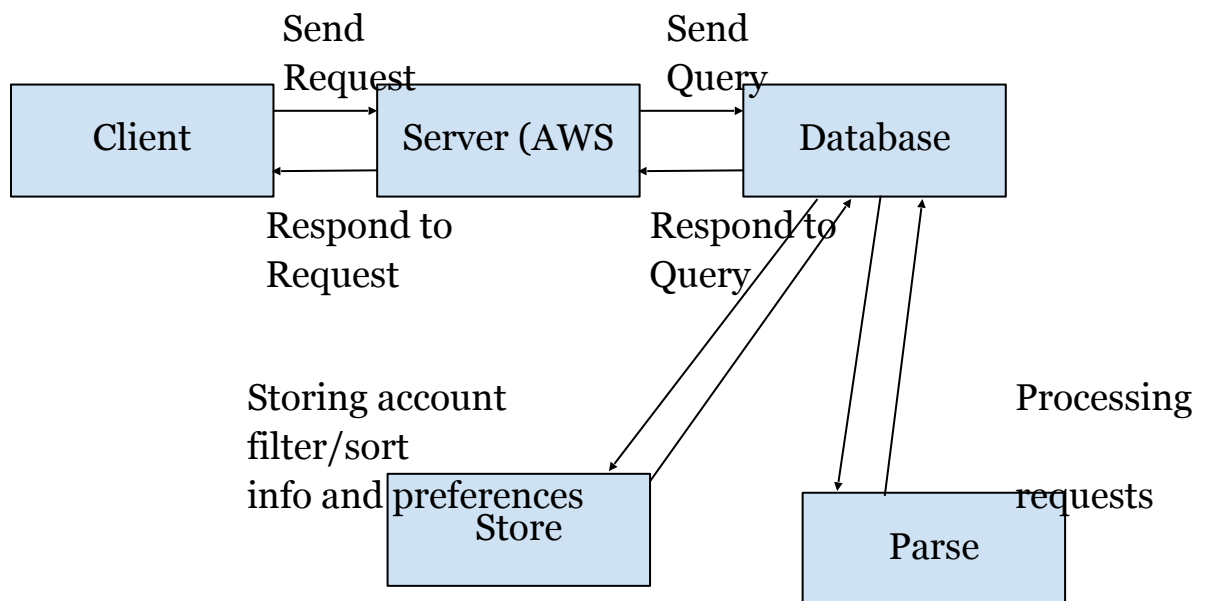
4. Scalability

- a. As a developer, I would like
 - i. accommodate most Purdue students
 - ii. handle larger amounts of data without having to redesign any of the software

Design Outline

High-Level Overview

This project will be a web application that connects students and tutors with one another based on preferences such as price, professor, and rating. We will be using a client-server model where one server will handle numerous requests from clients using the Node JS framework. In response to client requests, the server will access or save data in the database and provide feedback to the client(s) as necessary.



1. Client
 - a. Presents an interactive user interface through a web application
 - b. Sends HTTPS GET and POST requests to the server in response to user interactions to access and modify information.
 - i. GET requests are used to access user data, lists of tutors, discussion boards, calendars, etc.
 - ii. POST requests are used to change user data, post materials & questions/reply to discussion boards, modify calendars, etc.
 - c. Obtains and deciphers responses from the server and then subsequently updates the user interface

2. Server

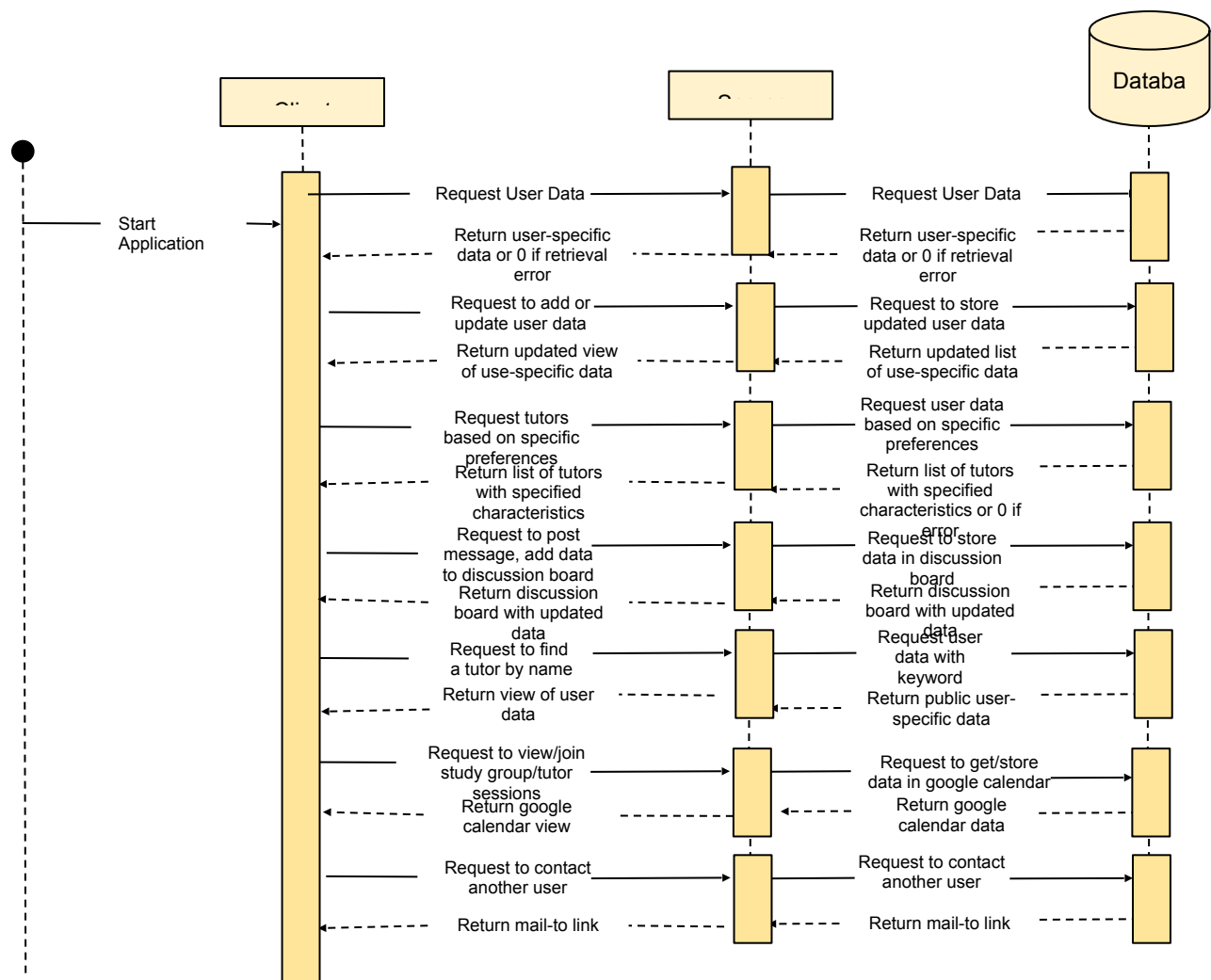
- a. Server receives and handles requests from the clients
- b. In order to add, change, or retrieve data on demand, the server verifies requests and sends queries to the database
- c. Server sends appropriate responses for requests to the corresponding client like displaying resulting updates that were requested by client (ex. showing updated calendar after client requests to create/join a session, showing updated account settings page after user has modified personal information, etc.)

3. Database

- a. Firebase is a NoSQL database that allows us to store and sync data in real time. This will be utilized to store all of our data which exists as user information
- b. Firebase replies to the server's queries and returns the data it has stored
- c. The database will have two modules, one that stores the information and one for processing.

Sequence of Events Overview

1. User interacts with the interface which is rooted in the client
2. Client communicates with the server via HTTPS GET and POST requests
 - a. We utilize GET for retrieving information from the database through the server as well as listing tutors based on preferences
 - b. We utilize POST for altering/declaring account preferences and declaring a user's status to the server
3. Requests are sent in the JSON format to the backend, which is comprised of a server derived from AWS and a Firebase database, which then sends data back to the client



Design Issues

Functional Issues

1. How do we calculate the ratings of users?
 - a. Option 1: Average all of the user ratings based on surveys
 - b. Option 2: Base rating on participation within discussion boards, user interactions, and participation

Decision: Option 1

Justification: While it would be a valid way to rate the users based on their activity, it wouldn't be all-encompassing. We want our users to be able to trust the ratings displayed on the site and thus we are going to display the average so that the users know *exactly* why an individual has a certain rating.

2. How will users communicate with each other?
 - a. Option 1: Direct Messaging
 - b. Option 2: Discussion Board

Decision: Option 2

Justification: We decided to use a discussion board for our users to communicate with one another to foster openness and allow users to be better able to form connections with one another. Using direct messaging would limit user contact in general and wouldn't allow for the sharing of resources and thoughts that can be used to help students. Our discussion boards will ensure that users are getting the most out of their tutoring experience.

3. How will students and tutors coordinate meetings?
 - a. Option 1: Direct Messaging
 - b. Option 2: Google Calendar

Decision: Option 2

Justification: In order to streamline the process for both students and tutors we decided to coordinate meetings via Google Calendar. Both student and tutor can communicate via their linked contact information; however, a Google Calendar will take out the confusion of trying to coordinate your schedule with

someone you don't know. This way students and tutors both don't have to waste time and can easily get together.

4. How will students' attendance be tracked?
 - a. Option 1: User Survey
 - b. Option 2: Tracking user participation based on site visits

Decision: Option 1

Justification: Ultimately we decided that we should base tracking attendance on user input. We won't know why a user didn't show up or there could be issues with us tracking said participation. The student could have communicated their absence prior with the tutor or they could have met up and forgotten to visit the site. Therefore, we are going to track attendance based on the post-meeting survey that all users will receive after they meet in which they'll mark attendance as well as rate their student.

5. How will students pay tutors?
 - a. Option 1: Paying through the site
 - b. Option 2: Linking an external payment site

Decision: Option 2

Justification: To make life easier for students who are trying to pay their tutor and for tutors who will have to deposit said money in their bank accounts, we are linking an external payment method so that users do not have to set up a new way to deposit money in and out of their account. Additionally, we would not have to save and manage payment information in our own database.

6. What qualifies you to be a verified tutor?
 - a. Option 1: Creating an account as a tutor
 - b. Option 2: References
 - c. Option 3: Transcript

Decision: Option 3

Justification: The tutors on our site should be qualified to teach said subject just like tutors and TAs at Purdue are vetted prior to working. Just as Purdue does it based on either GPA or class grade, we too will ensure that our tutors are qualified in this manner by having tutors upload their transcript.

7. How will tutors make extra resources available for users?

- a. Option 1: On Discussion Board
- b. Option 2: On their page

Decision: Option 1

Justification: As we want our tutoring site to be available to all and have resources there to benefit all students, just as students communicate via discussion board we will have tutors do that as well. This way tutors are being transparent about the materials that they are sharing, and it is another added step to ensure that the resources being shared are aligned with Purdue's Academic Honesty Pledge.

8. What information is required for users to create an account?

- a. Option 1: Name, Purdue email
- b. Option 3: Name, Purdue email, student or tutor option, transcript

Decision: Option 1

Justification: We chose to separate creating an account from entering set-up information. In order to create an account, we only require users to enter their name and account so that their email can be confirmed through email. From there, users can then enter their information when setting up their account.

9. When should personal information and preferences be required for an account?

- a. Option 1: Within the settings page
- b. Option 2: During sign up

Decision: Option 2

Justification: In order to create a more individualized tutoring app, we believe it is important to include one's primary language, courses, and other personal information. We want everyone to get the best possible tutor for their courses so by requiring everyone to put in the same information from the start rather than adding ones they want later on in settings, it ensures that users have a good experience.

Non-Functional Issues

1. What web hosting service should we use?

- a. Option 1: AWS Lambda
- b. Option 2: Surge
- c. Option 3: Netlify
- d. Option 4: Azure

Decision: Option 1

Justification: AWS Lambda has many free options that allow for up to a million requests per month. AWS Lambda also allows you to add custom logic to AWS resources. AWS Lambda will be best to use to implement with our React JS frontend and Node JS backend.

2. What frontend language/framework should we use?

- a. Option 1: raw HTML + JavaScript
- b. Option 2: React
- c. Option 3: Angular

Decision: Option 2

Justification: React is straightforward and simple to use with our current background. Creating components in React that have their own logic and rendering saves time and work because they can be reused whenever needed. The use of virtual DOM also guarantees a quick update time to the real DOM leading to higher performance.

3. What backend language should we use?

- a. Option 1: Node JS
- b. Option 2: Springboot (Java)
- c. Option 3: PHP

Decision: Option 1

Justification: It will be easy to use Node JS with our React JS frontend. Node JS is good for handling multiple requests from the client at once. It also offers easy scalability and high performance.

4. What database should we use?

- a. Option 1: MongoDB
- b. Option 2: Firebase
- c. Option 3: MySQL

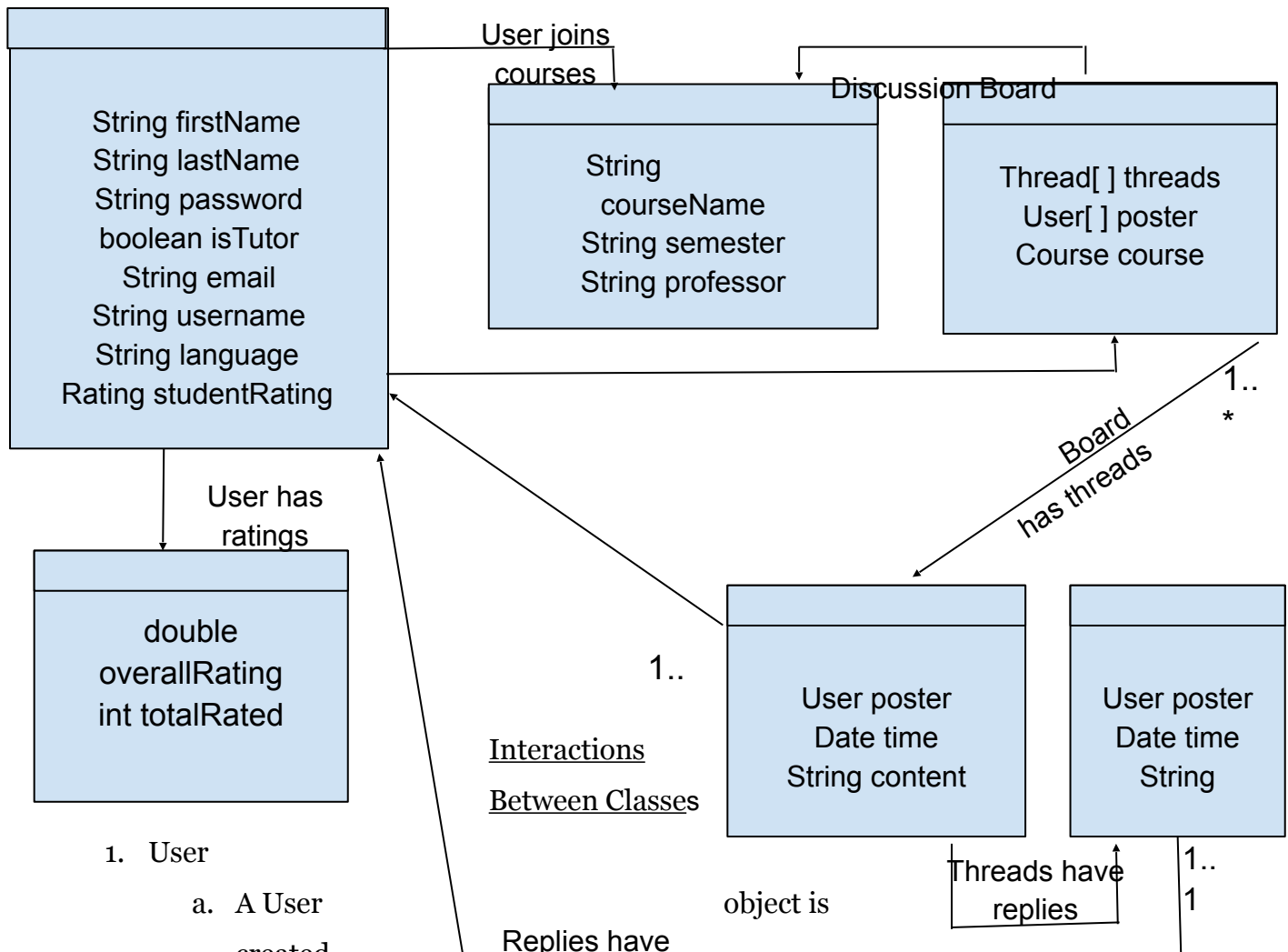
Decision: Option 2

Justification: Firebase will work well with our choice of Node JS as our backend software. Firebase is useful because it is just key/value pairs rather than having schema for the database. This will help improve the speed of requests while still accommodating large amounts of simultaneous users.

Design Details

Class Design

Description of Classes and



1. User

- a. A User created when someone signs up for an account.
- b. User has string attributes for first name, last name, email, username, password, and primary language spoken.
- c. User has a boolean isTutor that indicates if they have uploaded their transcript to be verified as a tutor or not. If they are not verified as a tutor, then their tutor rating and past courses will be null.

- d. User also has 2 Rating attributes: one for their rating as a student and one for their rating as a tutor.
 - e. The user will have 2 lists of courses: one of courses currently taking for the purpose of seeking tutoring and one for the courses they have taken in the past that they are offering tutoring in.
 - f. User will have a list of DiscussionBoards that they are a part of, either as a student or as a tutor.
2. Course
- a. A Course object will be created when a user lists a course that does not exist yet in either of their Course lists. If it does exist, it will simply be added to their list.
 - b. Course has string attributes for course name, professor, and semester.
3. DiscussionBoard
- a. A DiscussionBoard object would be created when a new discussion board is created for a class by a user (either student or tutor)
 - b. DiscussionBoard has lists of
 - i. Thread objects that are posts from users in the board
 - ii. User objects that have access to the board
 - c. DiscussionBoard has a Course attribute that contains the information for the course that it is associated with
4. Rating
- a. Rating objects will be created when a User object is created (studentRating) and a user becomes verified as a tutor (tutorRating)
 - b. Rating contains attributes for the calculated overall rating and the number of users that have rated that user
5. Thread
- a. A Thread object will be created when a user posts on a discussion board
 - b. Thread includes attributes for who posted it (User) and what time and date it was posted
 - c. Thread includes a string that contains the content of the post
 - d. Thread includes a list of Reply objects as users reply to the original thread

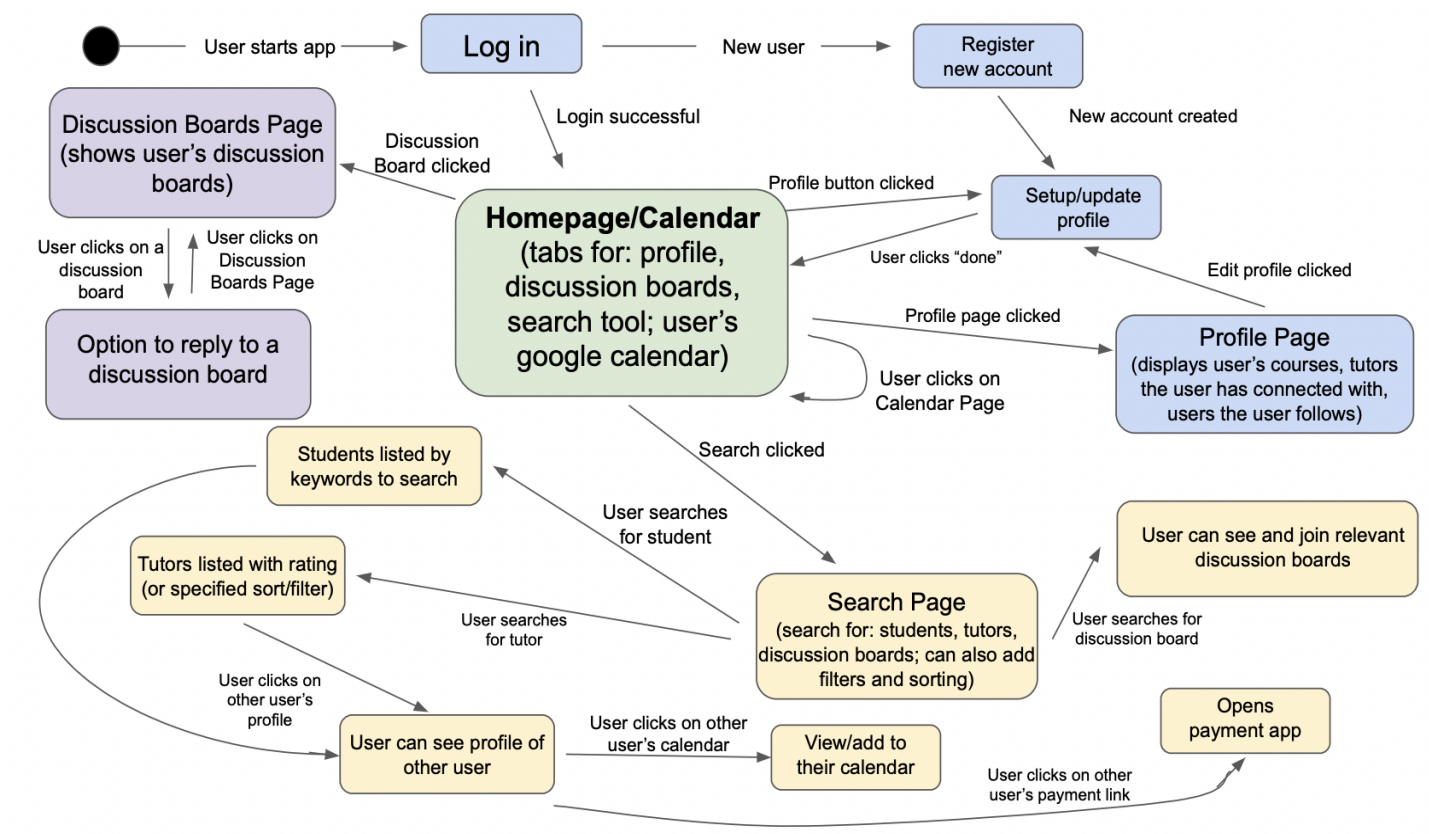
6. Reply

- A Reply object will be created when a user replies to a thread in a discussion board
- Reply includes attributes for who posted it (User) and what time and date it was posted
- Reply includes a string that contains the content of the post

Sequence Design

Activity Diagram:

This diagram shows how a user navigates to different pages on the website.

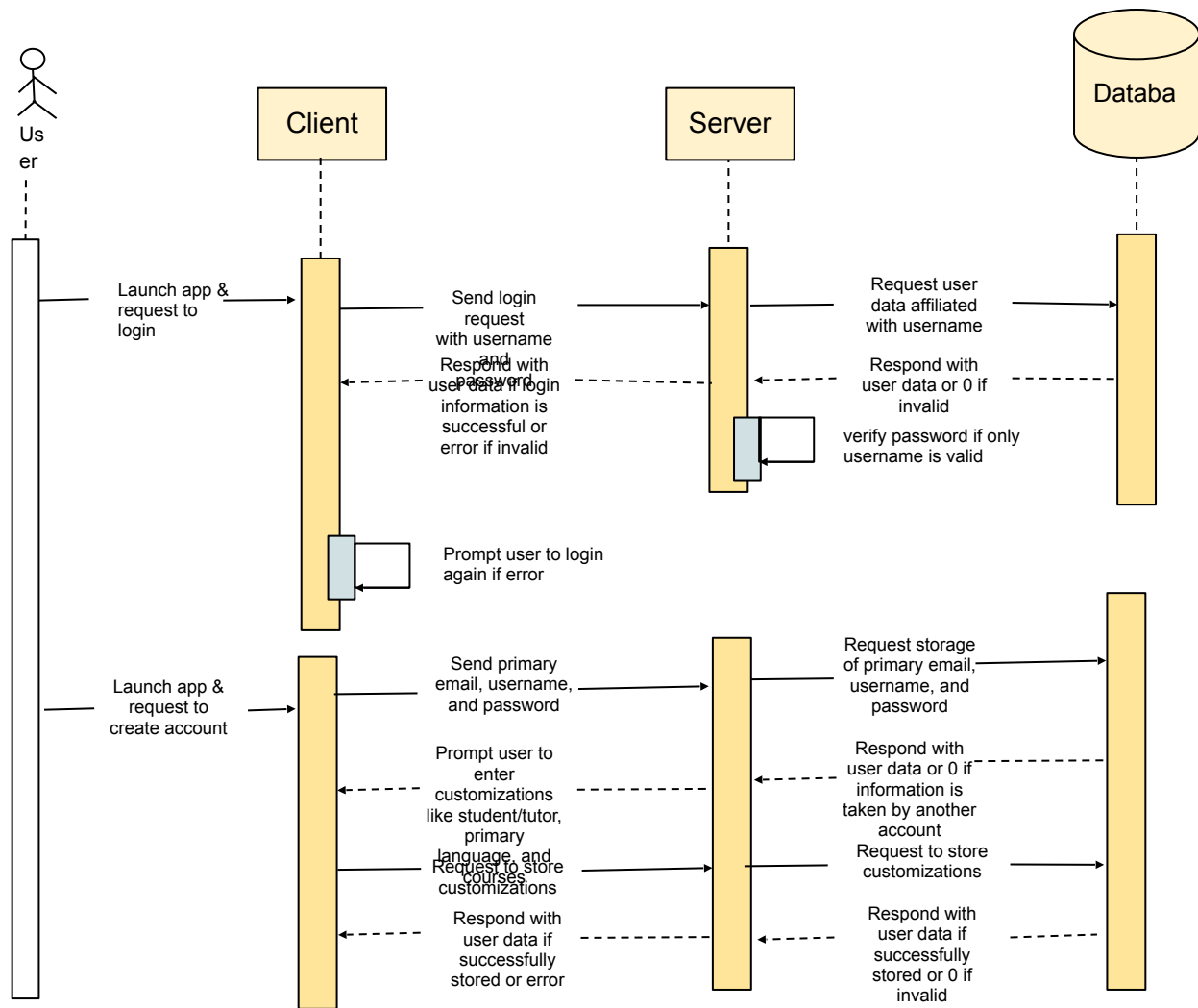


The following diagrams depict the sequence of events for the major events in our application, which are account creation and login, filtering, sorting, and searching for

tutors, scheduling and joining tutor/group study sessions, and accessing/modifying discussion boards. The diagrams convey the interactions between the client and server as well as the integration of a database for information storage. The client sends requests to the server in the JSON format to the backend, which then process and store the information that the client wished to access or modify and respond to the client with the updated information to display to the user.

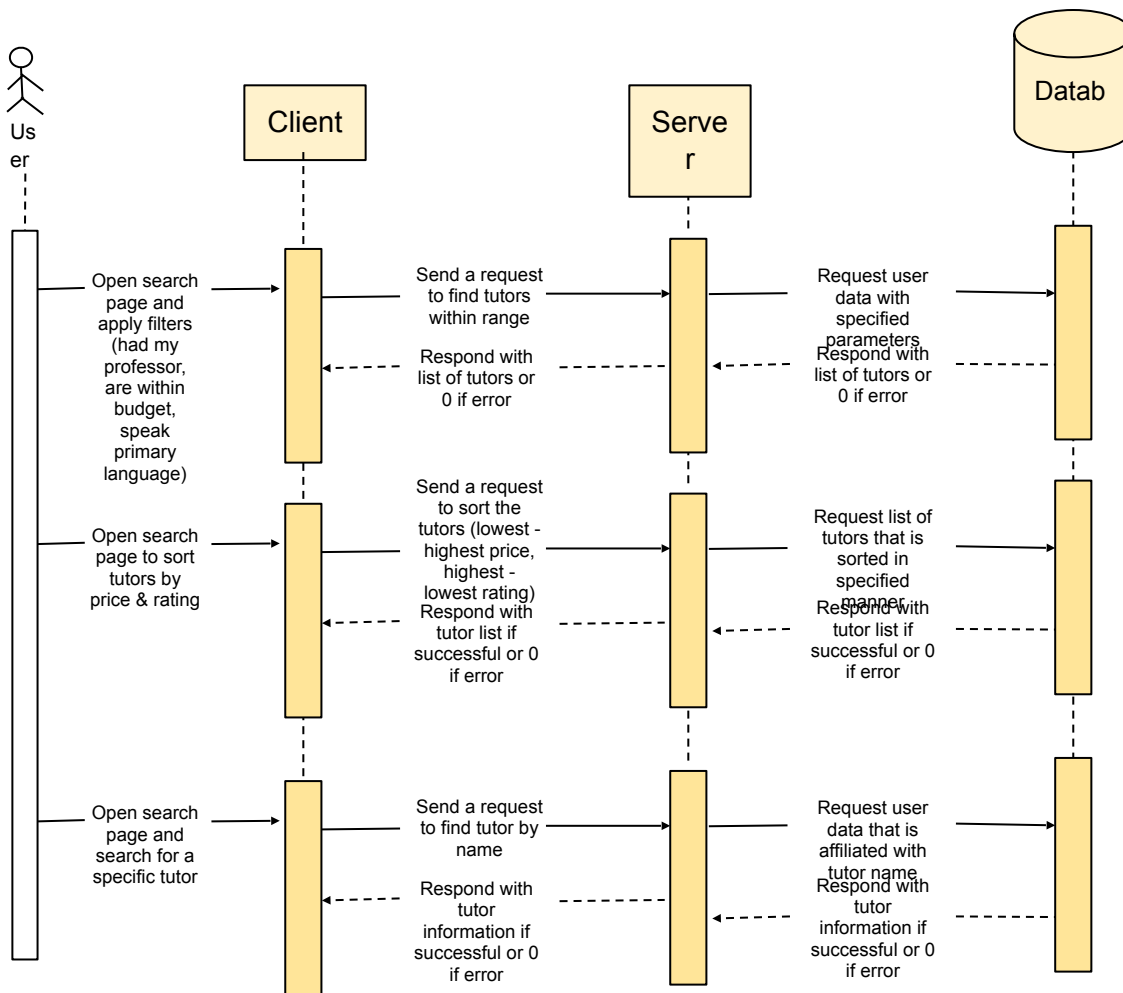
1. Account login/set up

This diagram shows the process of a user creating an account and logging in. The client and server interact as the user inputs their information and the server then sends it to the database to be stored/verified.



2. Filter/Sort/Search for tutors

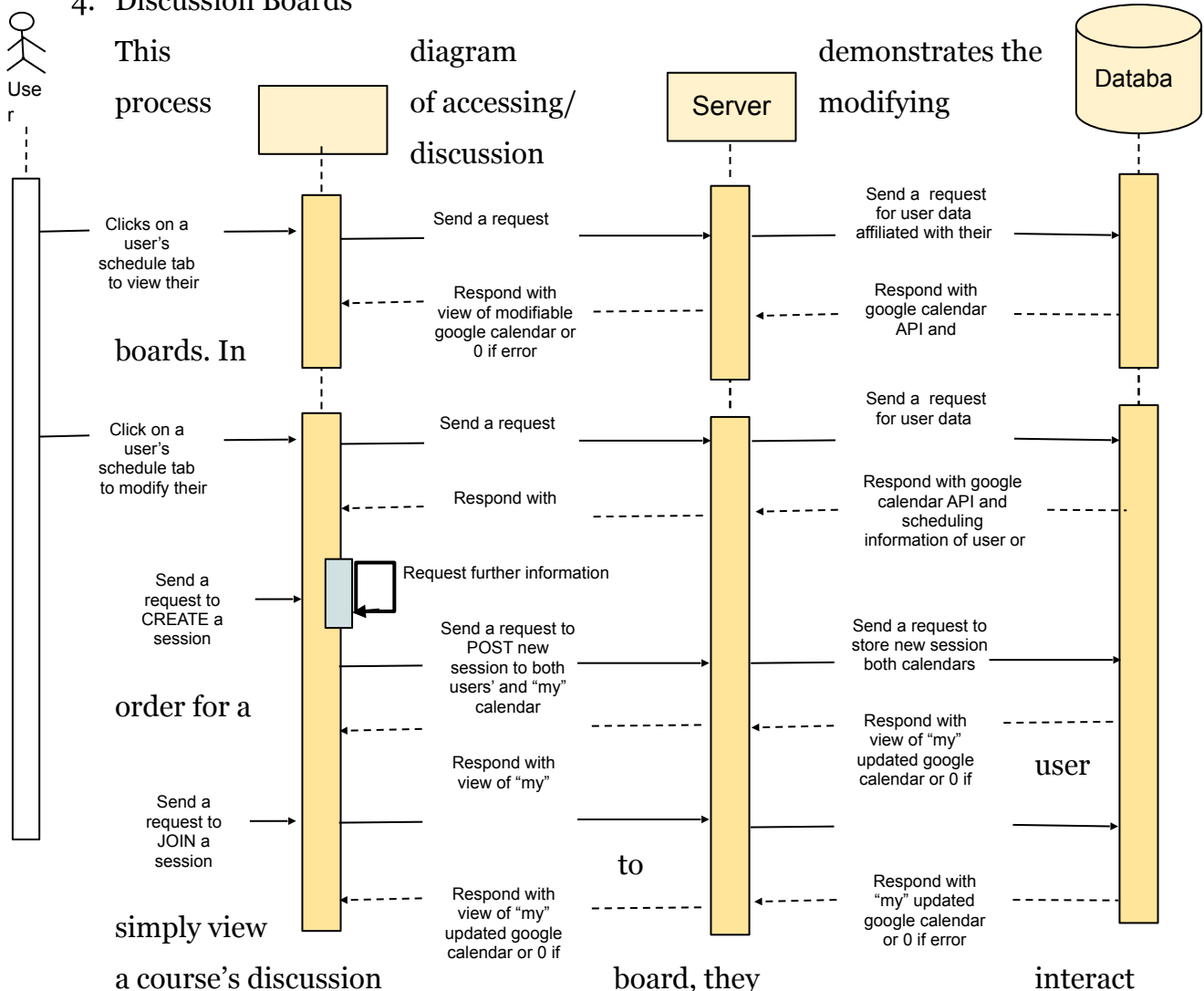
This diagram demonstrates the process of using the search, filter and sort functions of our application when trying to find tutors. To access any of these functionalities, the user would first open the search page. Then to filter tutors by the user customizations, a GET request would be sent to the server to get the list of tutors with the appropriate customizations and the server then accesses the information from the database and then responds with a list of tutors. Then to sort the tutors by a specified parameter (price low-to-high, etc.), a similar process is followed, as displayed by the diagram. Finally, to search for a specific tutor, the client sends a GET request for the user (tutor) data affiliated with the username to the server which then gets the stored information from the database, and the tutor information is displayed to the client.

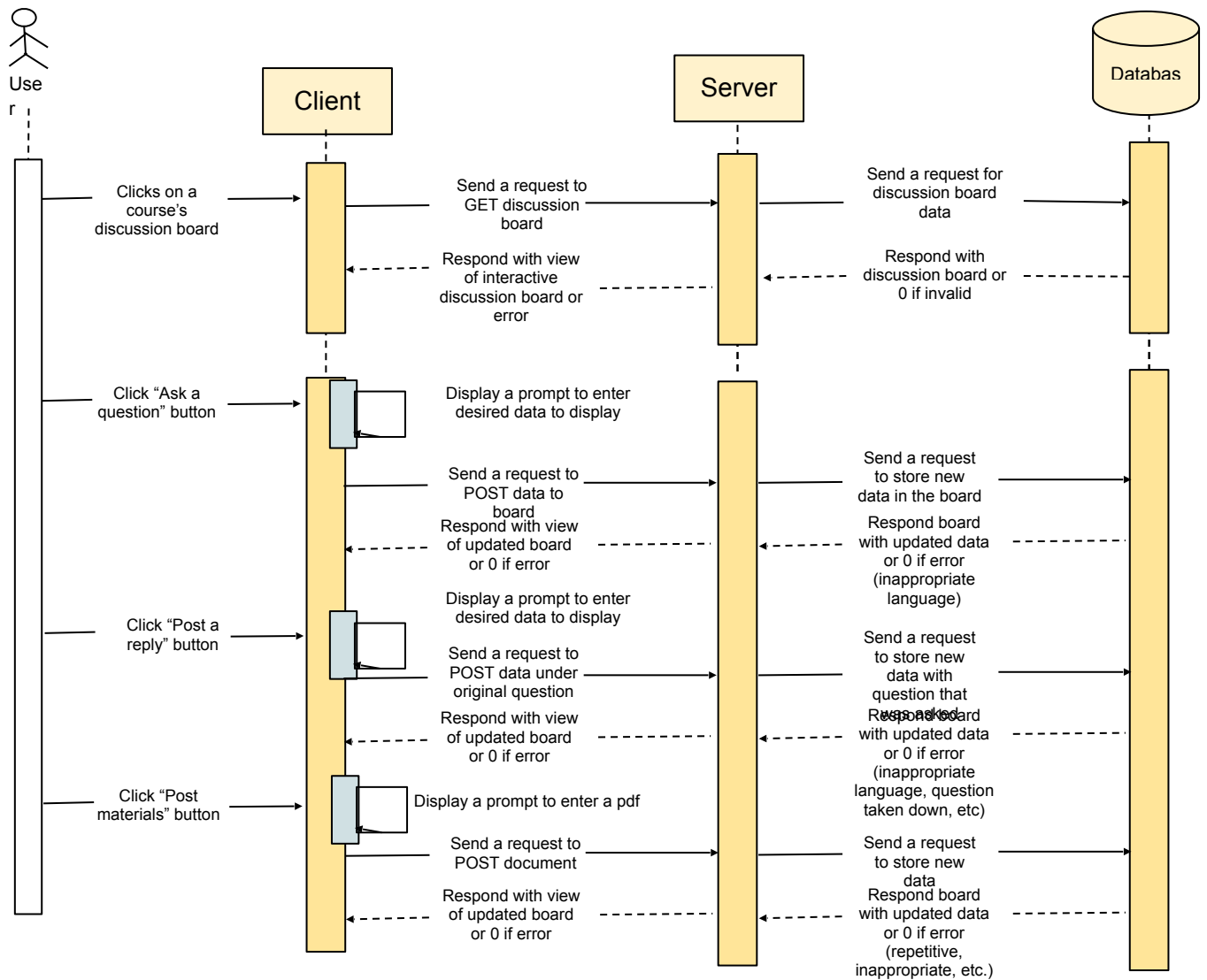


3. Scheduling sessions

This diagram shows the process of a user scheduling a tutoring or group study session. The user selects options to view/modify schedules and create or join a session. The client requests data about the schedules from the server and the server interacts with the database and Google Calendar API to assist in the process.

4. Discussion Boards





UI Mockups

Account Set Up Page

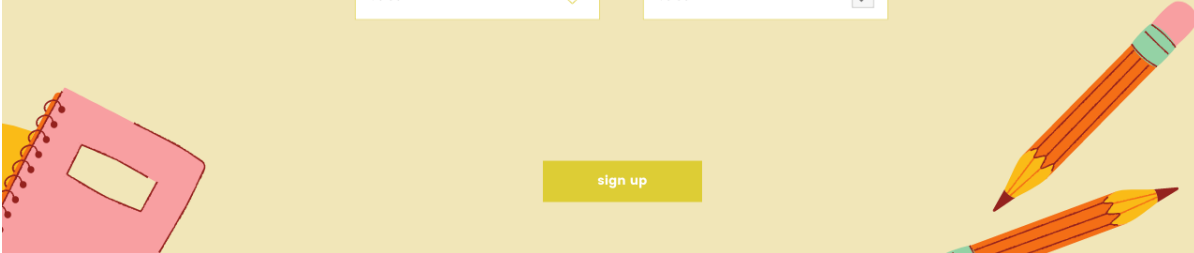
ooo

TutorsRU


Create new Account

FIRST NAME	USERNAME
<input type="text" value="Jane"/>	<input type="text" value="janedoe"/>
LAST NAME	EMAIL
<input type="text" value="Doe"/>	<input type="text" value="janedoe@purdue.edu"/>
PASSWORD	CURRENT COURSES
<input type="password" value="*****"/>	<input type="text" value="Select All"/> <input checked="" type="checkbox"/>
PREFERRED LANGUAGE	PAST COURSES
<input type="text" value="Select"/> <input type="button" value="v"/>	<input type="text" value="Select"/> <input checked="" type="checkbox"/>

sign up



Search Page



ooo


CS182

PREFERRED LANGUAGE: English

RATING: High to Low


PRICE RANGE: \$15-\$20

Tutors



Shreya Suri
CS182, CS240, CS180, CS251


★★★★★ 4.97



Evans Tang
CS182, CS240, CS180, CS251

★★★★★ 4.83

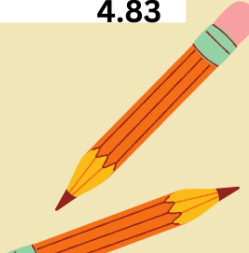
Boards




cs182 help

cs182 network

cs182 study group




Discussion Board



ooo HOME BOARDS PROFILE


CS182

CS182 HELP



Malia Marquez TUTOR

Creating a group tutoring session to go over induction!
I will post resources [here](#). I will see you there



Jharna Suresh STUDENT

Thanks! See you next week.

