Creating Online Tutoring Sessions within ASSISTments

An Interactive Qualifying Project
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science

by
Adam Grabowski

Date:
6 April 2022

Professor:
Neil Heffernan

Abstract

The goal of this project is to improve student learning and mitigate learning loss due to COVID-19 by implementing a school-based virtual or in-person tutoring program for students. A prototype of this program was designed and implemented with features like administrative overview of the system, tutoring session scheduling, and automatic Zoom meeting creation.

Acknowledgements

I would like to thank Neil Heffernan, my project advisor, for providing me with the opportunity to work on this project. Also, I would like to thank my supervisor Aaron Haim for helping to guide me through the technical aspects of this project. Finally, I would like to thank my partner Sean Jan for his work on the back end and database portions of this project.

Table of Contents

Chapter 1: Introduction

The team at ASSISTments proposed a tutoring support platform called TutorASSIST that acts as an extension of their ASSISTments website platform. TutorASSIST is meant to provide school administrators a way to assign students to volunteer tutors who would help guide them through coursework. The proposed features of the platform include administrative overview of the school system which would allow admins to assign tutors to students for online or in-person tutoring sessions. Automatic video recording of virtual tutoring sessions would be enabled so that students can replay sessions and an AI-Agent can suggest problems for the tutors to assign.

For this Interactive Qualifying Project, Sean Jan and I were tasked with implementing the user interface and administrative elements for the TutorASSIST platform. These included the admin meeting scheduler, the ability to view and join upcoming sessions as a tutor or student, a system to automatically create and record these sessions, and a system to authenticate real ASSISTments accounts.

Figure 1: Assistments Logo Graphic

Chapter 2: Background

ASSISTments applications like TutorASSIST use a Vue.js front end with typescript and Java back end with Apache Tomcat server for testing, and a PostgreSQL database management system. Due to our individual preferences and expertise within the group, I worked on implementing and designing the front end in Vue.js and Vuetify while Sean developed the back end. While we did focus on our aspects of the project, there were instances where I had to work on the back end and Sean had to work on the front end.

Libraries/Resources

The back end code was written in Java using Eclipse as an IDE. Apache Tomcat was used to host a local server for testing, and PostgreSQL was the database management system that held all of our persistent data for this project. The front end code was written using Vue.js and Typescript, and Node.js was used to host the code locally. Vuetify was the design framework used for the user interface of this project. Also, the Zoom API was used to create and schedule meetings automatically through Zoom.

Code Repositories

Our code repositories for the both the front end and back end implementations of this project can be found at the following links hosted on GitHub.

Front end (alpha branch): https://github.com/ahaim-specs/VirtualTutoringSessionsVue

Back end (alpha branch): https://github.com/ahaim-specs/VirtualTutoringSessions

Chapter 3: Design

In starting the design process, our team decided to implement views and functionality for four users: administrators, tutors, teachers, and students. Planning the design of our application included writing several use cases, creating a relational database schema diagram, and designing a user interface mockup with Figma.

Use Cases

Our use cases were split into four different categories, one for each of the four participating actors. A comprehensive list of our use cases detailing the desired functionality of a complete application is available in appendix A.

Database Schema Diagram

Multiple versions of the database schema diagram were created by Sean. The final iteration of the schema focuses on the creation of meetings and making changes to the table entries once they have been created. The relational database schema diagram is available in appendix B.

User Interface Mockup

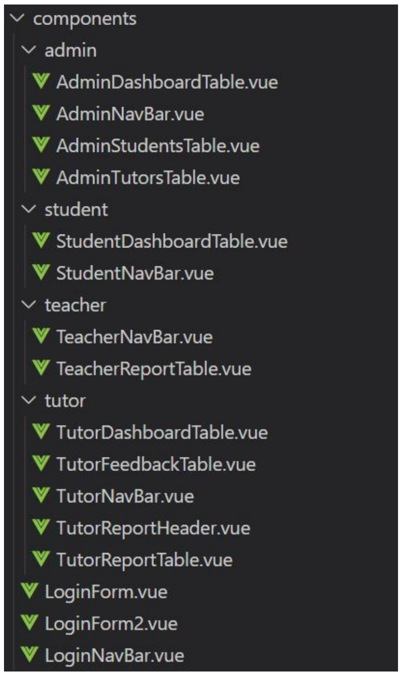
I designed the initial user interface mockup to match the theme of existing ASSISTments web applications. In total, nine pages were designed which correspond to different views that a certain participating actor would see. For example, a dashboard was created for administrators to see an overview of each meeting currently scheduled between tutors and students. These user interface mockup panels can be found in Appendix B with labels.

Chapter 4: Implementation

Our team's goal for this Interactive Qualifying Project was to create a minimum viable product that was demonstrated to ASSISTments leaders. At a minimum, the TutorASSIST application should include functionality for school administrators to schedule meetings between students and tutors. Also, the creation of this tutoring session needs to automatically set up a real Zoom meeting between them. Throughout this project, I focused on our front end implementation using Vue and Typescript, so I will be focusing on that in this section.

Components

In carrying out the front end implementation, I created views for each participating actor based on our user interface mockup. To simulate how these users would interact with our application, I created a login view where the user can sign in as an admin, tutor, teacher, or student. From there, the user can access all views associated with that type of user. The image below shows the structure of these components in our front end repository.



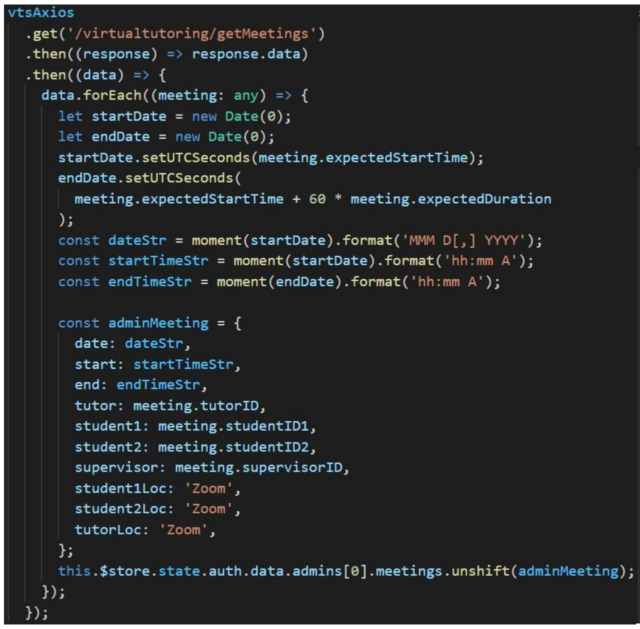
```
components
  admin
    AdminDashboardTable.vue
    AdminNavBar.vue
    AdminStudentsTable.vue
    AdminTutorsTable.vue
  student
    StudentDashboardTable.vue
    StudentNavBar.vue
  teacher
    TeacherNavBar.vue
    TeacherReportTable.vue
  tutor
    TutorDashboardTable.vue
    TutorFeedbackTable.vue
    TutorNavBar.vue
    TutorReportHeader.vue
    TutorReportTable.vue
  LoginForm.vue
  LoginForm2.vue
  LoginNavBar.vue
```

Figure 2: Components Structure

In order to create these views, I had to learn various Javascript libraries including Moment.js for converting Unix epoch seconds to a readable date and time for meeting. Also, I had to learn the syntax of Vuetify components like v-data-table, v-autocomplete, v-btn, v-card, and v-dialog, for the various menus and windows that had to be created. For example, the tables of meetings in the dashboard views are wrapped in v-data-table's and the pop-up menus for meeting creation and tutor addition use v-dialog.

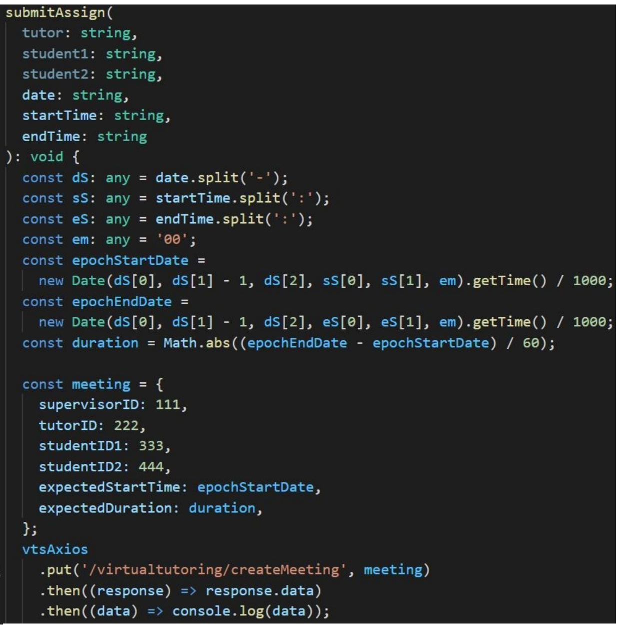
Meeting Scheduler

As stated earlier, the main goal of this project was to allow administrators to assign tutors to students and automatically create a Zoom meeting for that session. The first part of this was to create an API that allows the front end to interact with our database by adding a meeting to it. Sam created a table that stores the information needed for a meeting, including the meeting ID, supervisor ID, tutor ID, student IDs, start time, duration, Zoom ID and Zoom join link. To allow for the front end to add information to the database, Sam wrote a two API request on the backend, one that creates a meeting and one that gets a list of meetings. Then, by specifying the parameters of the Zoom meeting and providing an authorization token, we scheduled a Zoom meeting using just the Zoom API and create meeting request. The request that I made to get a list of meetings, create a meeting, and convert the data into presentable information on the front end are shown in the figures below.



```
vtsAxios
  .get('/virtualtutoring/getMeetings')
  .then((response) => response.data)
  .then((data) => {
    data.forEach((meeting: any) => {
      let startDate = new Date(0);
      let endDate = new Date(0);
      startDate.setUTCSeconds(meeting.expectedStartTime);
      endDate.setUTCSeconds(
        meeting.expectedStartTime + 60 * meeting.expectedDuration
      );
      const dateStr = moment(startDate).format('MMM D[,] YYYY');
      const startTimeStr = moment(startDate).format('hh:mm A');
      const endTimeStr = moment(endDate).format('hh:mm A');

      const adminMeeting = {
        date: dateStr,
        start: startTimeStr,
        end: endTimeStr,
        tutor: meeting.tutorID,
        student1: meeting.studentID1,
        student2: meeting.studentID2,
        supervisor: meeting.supervisorID,
        student1Loc: 'Zoom',
        student2Loc: 'Zoom',
        tutorLoc: 'Zoom',
      };
      this.$store.state.auth.data.admins[0].meetings.unshift(adminMeeting);
    });
  });
```



```
submitAssign(
  tutor: string,
  student1: string,
  student2: string,
  date: string,
  startTime: string,
  endTime: string
): void {
  const dS: any = date.split('-');
  const sS: any = startTime.split(':');
  const eS: any = endTime.split(':');
  const em: any = '00';
  const epochStartDate =
    new Date(dS[0], dS[1] - 1, dS[2], sS[0], sS[1], em).getTime() / 1000;
  const epochEndDate =
    new Date(dS[0], dS[1] - 1, dS[2], eS[0], eS[1], em).getTime() / 1000;
  const duration = Math.abs((epochEndDate - epochStartDate) / 60);

  const meeting = {
    supervisorID: 111,
    tutorID: 222,
    studentID1: 333,
    studentID2: 444,
    expectedStartTime: epochStartDate,
    expectedDuration: duration,
  };
  vtsAxios
    .put('/virtualtutoring/createMeeting', meeting)
    .then((response) => response.data)
    .then((data) => console.log(data));
```

Figures 2 and 3: Get Meetings and Create Meeting Requests

Authentication

After completing and presenting the minimum viable product of TutorASSIST, our team decided that the next step in completing the application was to implement authentication for real accounts. I began work on this step by setting up the authentication request, which checks if a user is logged in and responds with that accounts information. This information, such as the user's name and type could then be kept in the local store implemented in the files shown below for use in the application. Though substantial progress has been made, authentication was not completed at the time of writing this report.

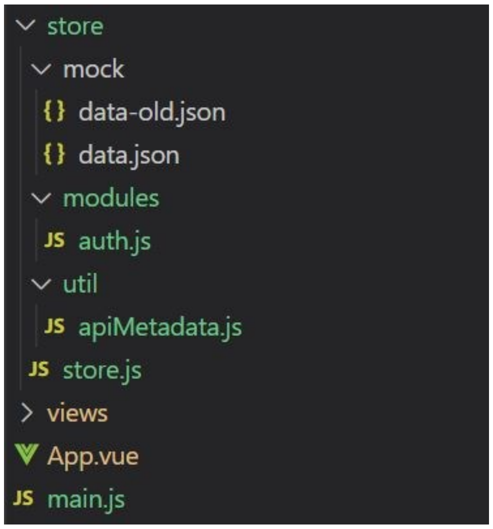


Figure 4: Authentication Implementation

In order to implement authentication, I had to create a method on the backend to receive a request that authenticates a particular user from the front end. When authenticating the user on our backend, in some database we would know whether the user is an administrator, tutor, or student. That information would be sent to our front end, where the auth module holds the state of the role and other account information. Through the commit context, the auth module then sets those states so it is persistent throughout the session.

Chapter 6: Future Plans

Besides completing user authentication, there are potential improvements that can be made to this project in the future. These include making the user interface easier to navigate, allowing the user to authenticate into Zoom, adding functionality to the teacher page for question assignment, and allowing tutors or teachers to write comments for sessions. This project was created from the ground up, so there is room for future project teams to build off of it.

Appendix A: Use Cases

Student Use Cases

UC 1: Join meeting
Participating actor: Initiated by student
Entry Condition: Session exists and student is not in tutoring session
Exit Criteria: Student has joined tutoring session
Flow of Events:
Student requests to join a meeting
App puts the student in the meeting

UC 2: Enter answer
Participating actor: Initiated by student
Entry Condition: Session has started and question exists
Exit Criteria: Answer has been entered
Flow of Events:
Student requests to enter answer
App enters answer and shows it on screen

UC 3: Submit wrong answer
Participating actor: Initiated by student
Entry Condition: Answer has been typed
Exit Criteria: Answer has been submitted and answer is wrong
Flow of Events:
Student requests to submit answer
App submits answer and shows the answer is wrong

UC 3: Submit right answer
Participating actor: Initiated by student
Entry Condition: Answer has been typed
Exit Criteria: Answer has been submitted, answer is right and question is locked
Flow of Events:
Student requests to submit answer
App submits answer and shows the answer is right

UC 4: Show recordings
Participating actor: Initiated by student
Entry Condition: There are no active meetings
Exit Criteria: recordings are shown
Flow of Events:
Student requests to show recordings
App shows recordings

UC 5: Select recording
Participating actor: Initiated by student
Entry Condition: Recordings are shown
Exit Criteria: A recording is selected
Flow of Events:
Student requests to select a recording
App selects a recording and deselects other recordings

UC 6: Show recording
Participating actor: Initiated by student
Entry Condition: A recording is selected
Exit Criteria: A recording is shown
Flow of Events:
Student requests to show a recording
App shows a recording

Tutor Use Cases

UC 7: Select student
Participating actor: Initiated by tutor
Entry Condition: None
Exit Criteria: A student has been selected
Flow of Events:
Tutor requests to select a student
App selects a student and opens up a student page

UC 8: Start tutoring session
Participating actor: Initiated by tutor
Entry Condition: Tutoring session exists and tutoring session has not started
Exit Criteria: A tutoring session has started
Flow of events:
Tutor requests to start a tutoring session
App starts a tutoring session and puts tutor in session

UC 9: Start meeting
Participating actor: Initiated by tutor
Entry Condition: A tutoring session is opened, tutoring session is not archived, and meeting does not exist
Exit Criteria: A meeting has started
Flow of Events:
Tutor requests to start a meeting
App starts a meeting, puts tutor in meeting and starts recording meeting

UC 9: Create question
Participating actor: Initiated by tutor
Entry Condition: A meeting has started and there are no unanswered or wrong questions
Exit Criteria: A question appears
Flow of Events:
Tutor requests to create a question
App creates a question and shows it to the student

UC 10: Skip question
Participating actor: Initiated by tutor
Entry Condition: A meeting has started and there exists an unanswered or wrong question
Exit Criteria: A question appears
Flow of Events:
Tutor requests to skip a question
App locks current question and creates a new question

UC 11: Open teacher notes
Participating actor: Initiated by tutor
Entry Condition: A tutoring session is opened
Exit Criteria: Teacher notes are opened
Flow of Events:
Tutor requests to show teacher notes
App opens teacher notes and shows it to the tutor

UC 12: Submit tutor notes
Participating actor: Initiated by tutor
Entry Condition: A tutoring session is opened and tutoring session is not archived,
Exit Criteria: Notes from tutor are submitted
Flow of Events:
Tutor requests to submit notes
App submits tutor notes

UC 13: End tutor session
Participating actor: Initiated by tutor
Entry Condition: A tutoring session is opened
Exit Criteria: Tutoring session is closed
Flow of Events:
Tutor requests to end session
App ends tutor session and archives tutor session

UC 14: Select tutor sessions
Participating actor: Initiated by tutor
Entry Condition: Student is selected and there are no active tutor sessions
Exit Criteria: A Tutor session is selected
Flow of Events:
Tutor requests to select tutor session
App selects tutor session and places tutor in tutor session

UC 15: Show recording
Participating actor: Initiated by tutor
Entry Condition: A tutor session is selected and archived
Exit Criteria: A recording is shown
Flow of Events:
Tutor requests to show a recording
App shows a recording
Teacher side use cases

UC 16: Select student
Participating actor: Initiated by teacher
Entry Condition: None
Exit Criteria: A student is selected
Flow of Events:
Teacher requests to select a student
App selects a student and opens up a student page

UC 17: Select tutor sessions
Participating actor: Initiated by teacher
Entry Condition: A student is selected
Exit Criteria: A Tutor session is selected
Flow of Events:
Teacher requests to select tutor session
App selects tutor session and places teacher in tutor session

UC 18: Assign problems
Participating actor: Initiated by teacher
Entry Condition: A Tutor session is selected and is not archived
Exit Criteria: Problems are assigned to tutor session
Flow of Events:
Teacher requests to assign problems
App assigns problems to tutor session

UC 19: Submit teacher notes
Participating actor: Initiated by teacher
Entry Condition: A Tutor session is selected and is not archived
Exit Criteria: Teacher notes are submitted to tutor session
Flow of Events:
Teacher requests to submit teacher notes
App submits teacher notes to tutor session

UC 20: Open tutor notes
Participating actor: Initiated by teacher
Entry Condition: A tutoring session is opened and archived
Exit Criteria: Tutor notes are opened
Flow of events:
Teacher request to open tutor notes
App opens tutor notes

UC 21: Show recording
Participating actor: Initiated by teacher
Entry Condition: A tutor session is selected and archived
Exit Criteria: A recording is shown
Flow of Events:
Teacher requests to show a recording
App shows a recording

UC 22: Show data for tutor session
Participating actor: Initiated by teacher
Entry Condition: A tutor session is selected and archived
Exit Criteria: Data for a tutor session is shown
Flow of Events:
Teacher requests to show data
Data for a tutor session is shown

UC 22: Show data for student
Participating actor: Initiated by teacher
Entry Condition: A student selected
Exit Criteria: Data for a student is shown
Flow of Events:
Teacher requests to show data
Data for a student is shown

Admin Use Cases

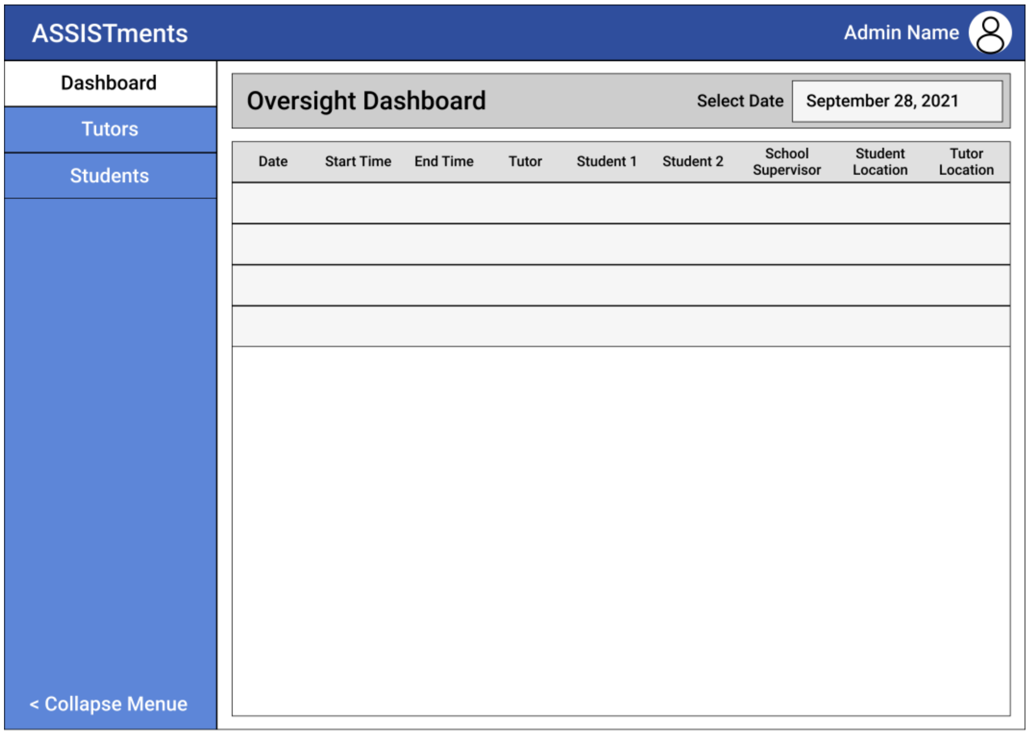
UC 23: Assign tutor to student
Participating actor: Initiated by admin
Entry condition: Student isn't already assigned to a tutor
Exit Criteria: Student is assigned to a tutor
Flow of events:
Admin requests for a tutor to be assigned to a student
App assigns tutor to student and creates tutoring sessions

UC 24: Unassign tutor from student
Participating actor: Initiated by admin
Entry condition: Student is already assigned to a tutor
Exit Criteria: Student is unassigned from a a student
Flow of events:
Admin requests for a tutor to be unassigned from a student
App unassigned tutor from student and deletes un-archived tutoring sessions

UC 25: View tutoring sessions
Participating actor: Initiated by admin
Entry condition: None
Exit Criteria: Admin views all tutoring sessions
Flow of events:
Admin requests to view tutoring sessions
App opens list of tutoring sessions and shows tutoring sessions

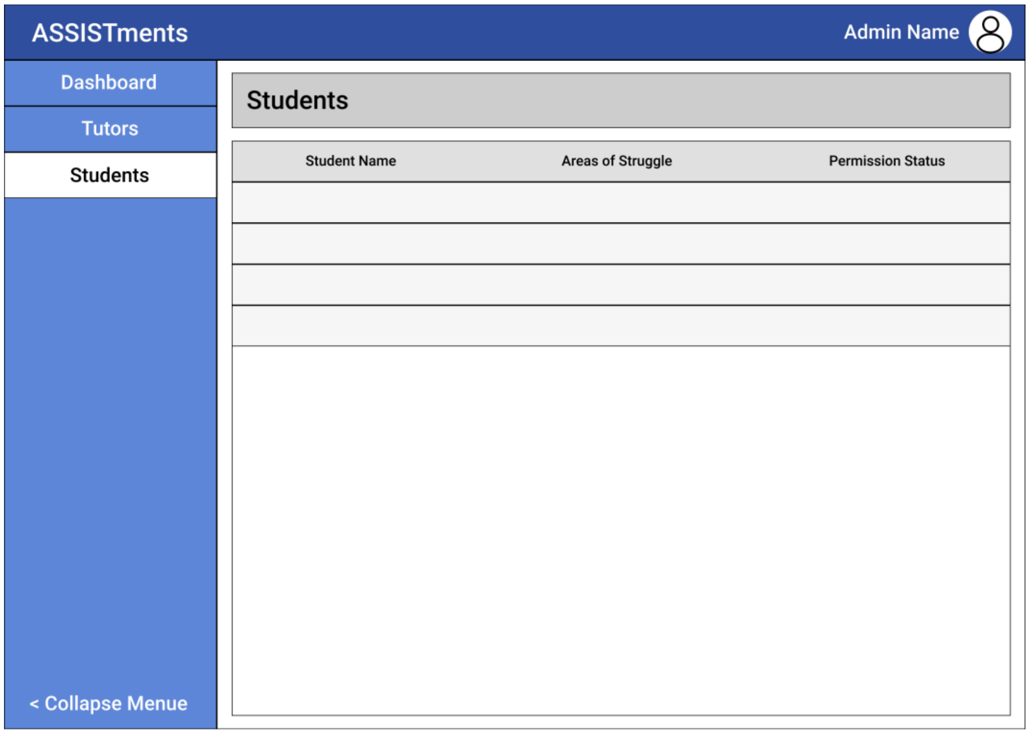
Appendix B: UI Mockup Views

Admin Dashboard



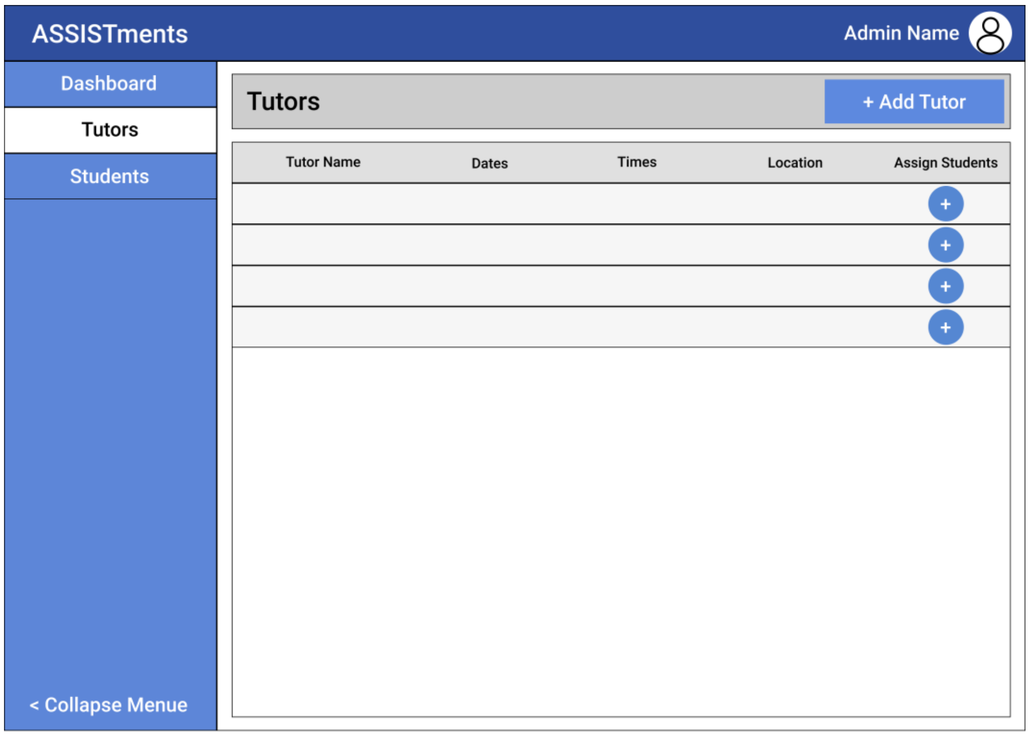
Date	Start Time	End Time	Tutor	Student 1	Student 2	School Supervisor	Student Location	Tutor Location

Admin Students

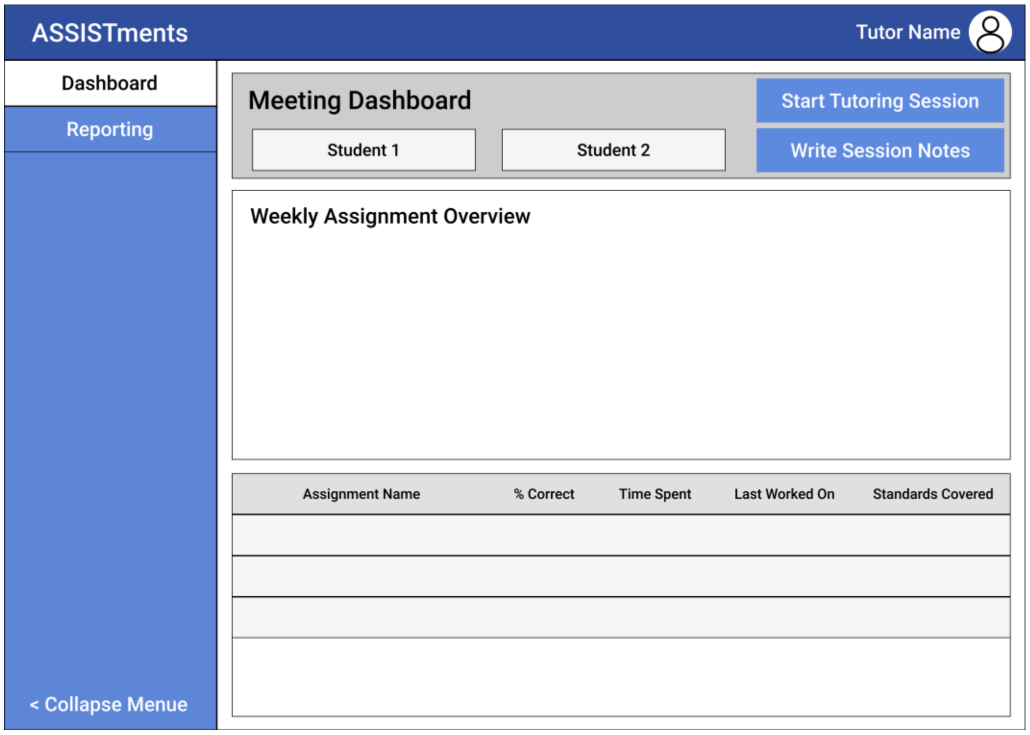


Student Name	Areas of Struggle	Permission Status

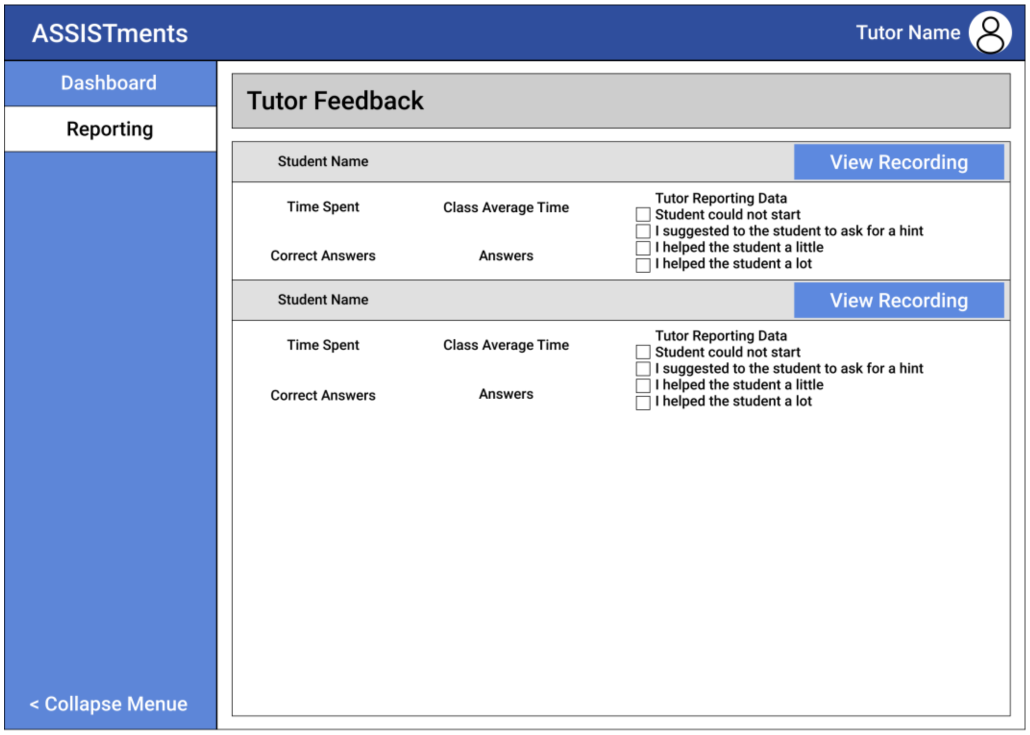
Admin Tutors



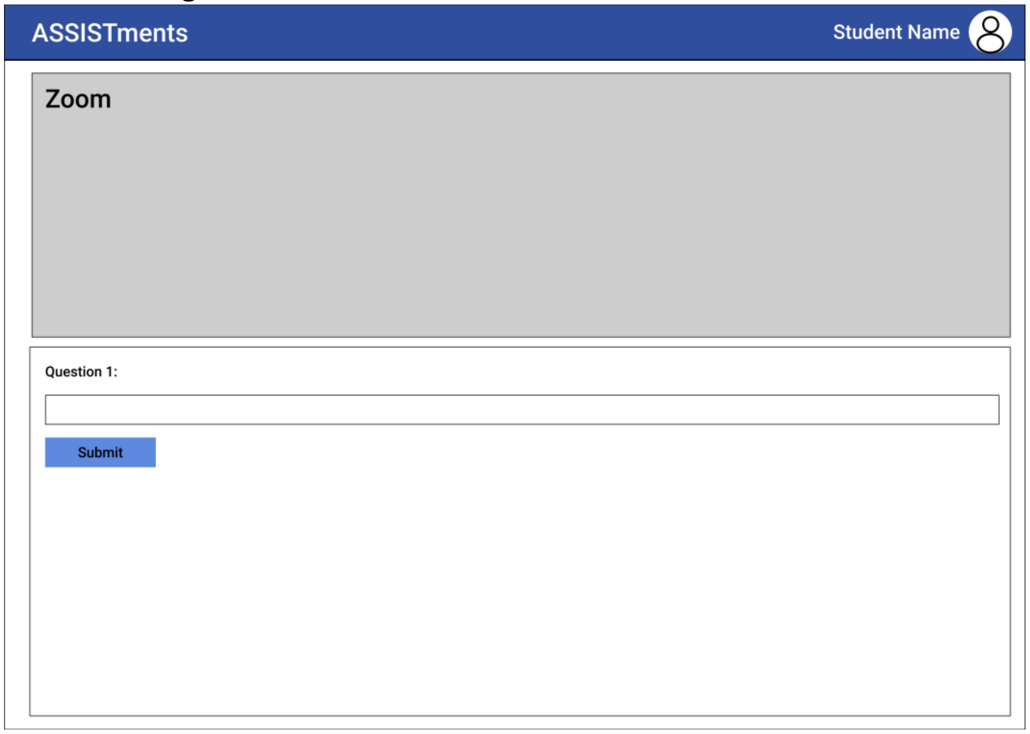
Tutor Dashboard



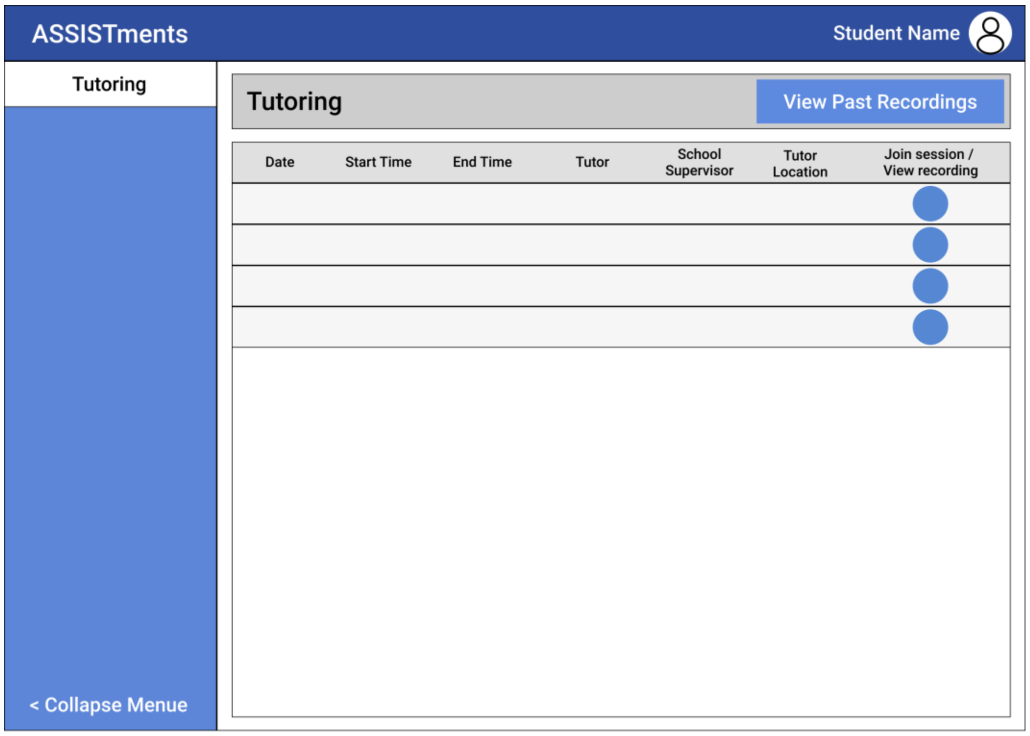
Tutor Feedback



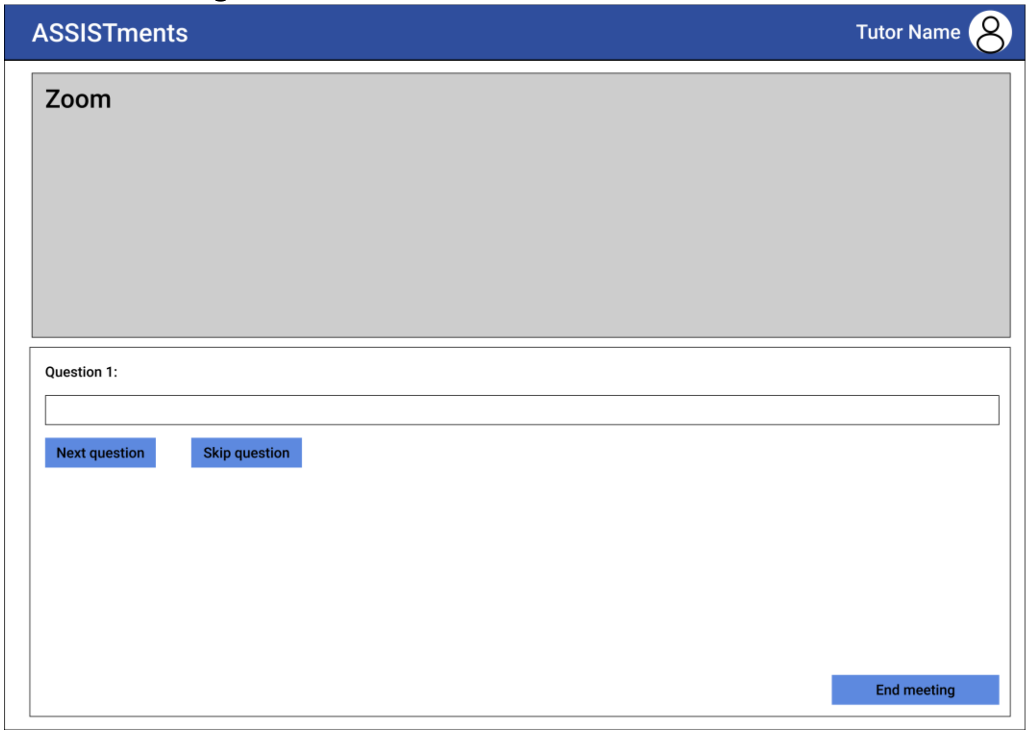
Tutor Meeting



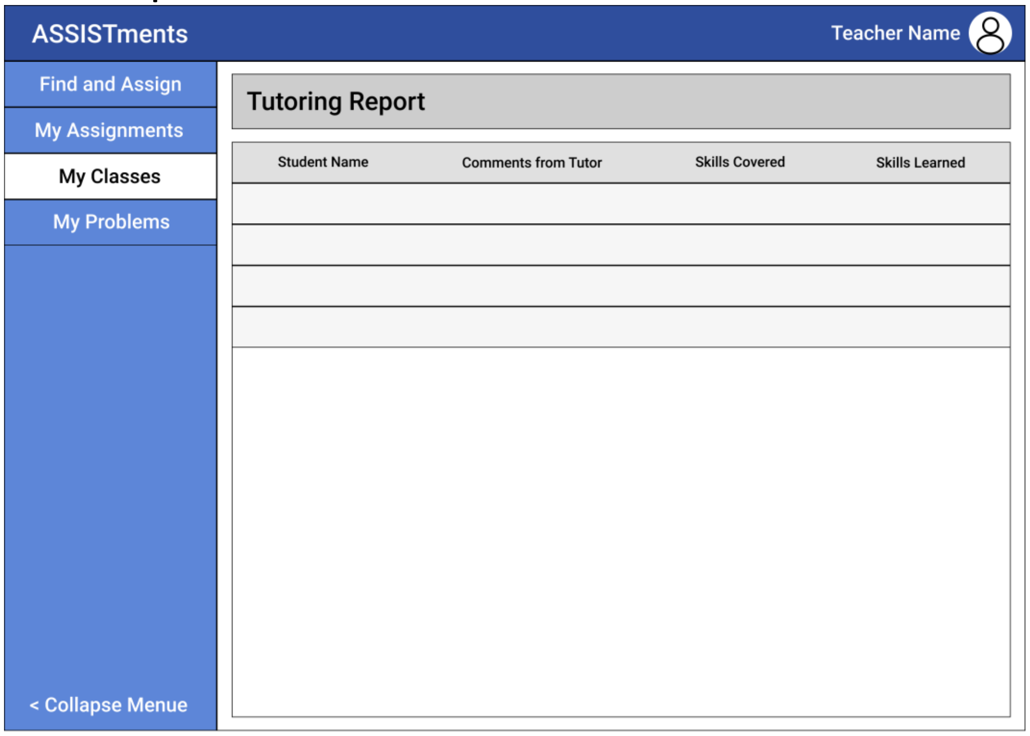
Student Dashboard



Student Meeting



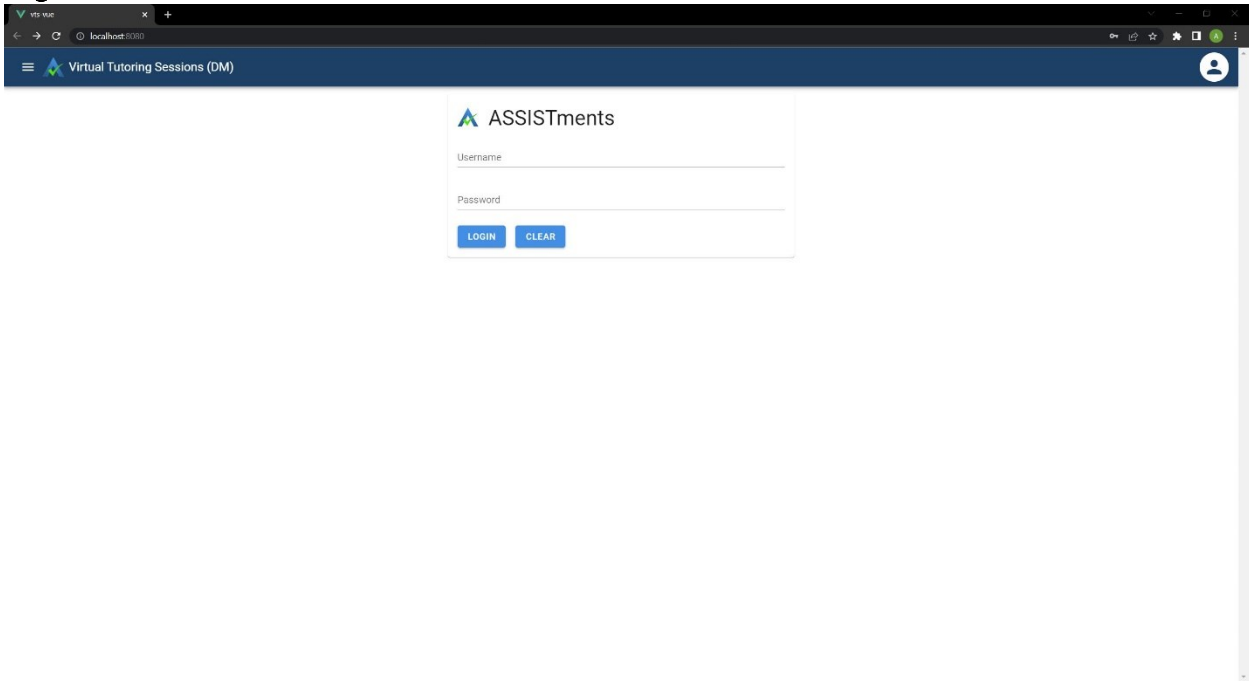
Teacher Report



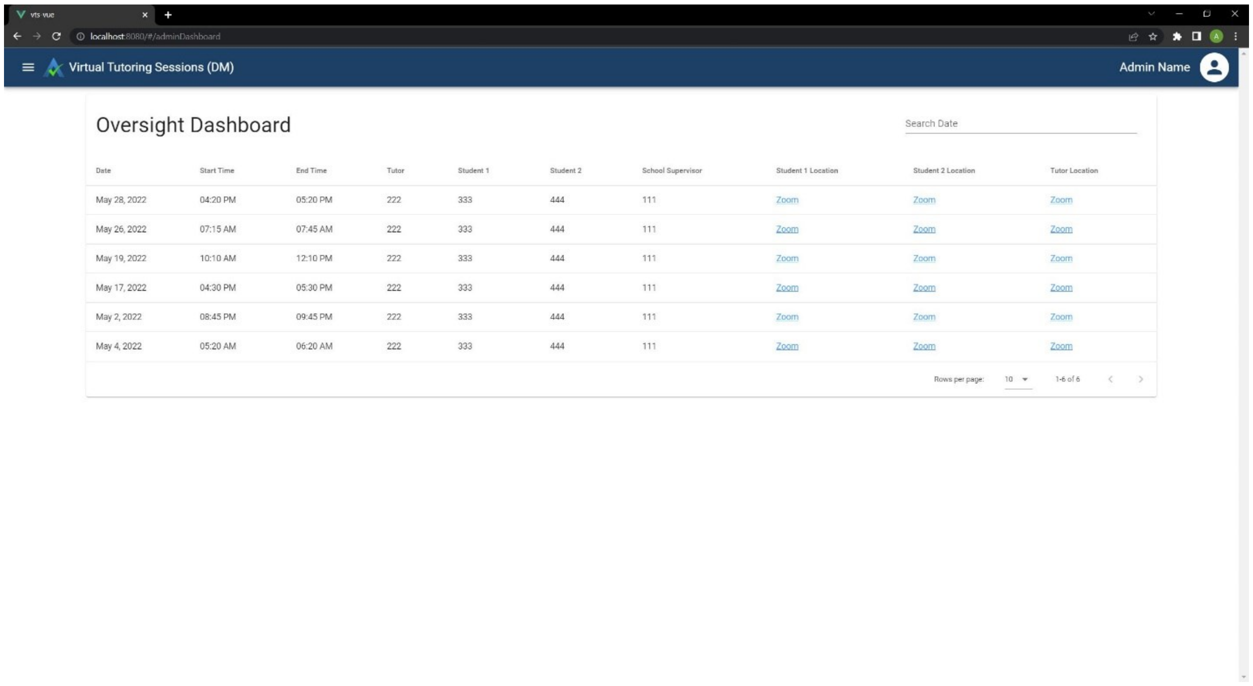
Student Name	Comments from Tutor	Skills Covered	Skills Learned

Appendix C: Implementation Views

Login



Admin Dashboard


vts-vue
localhost:8080/#/adminDashboard
Virtual Tutoring Sessions (DM)
Admin Name

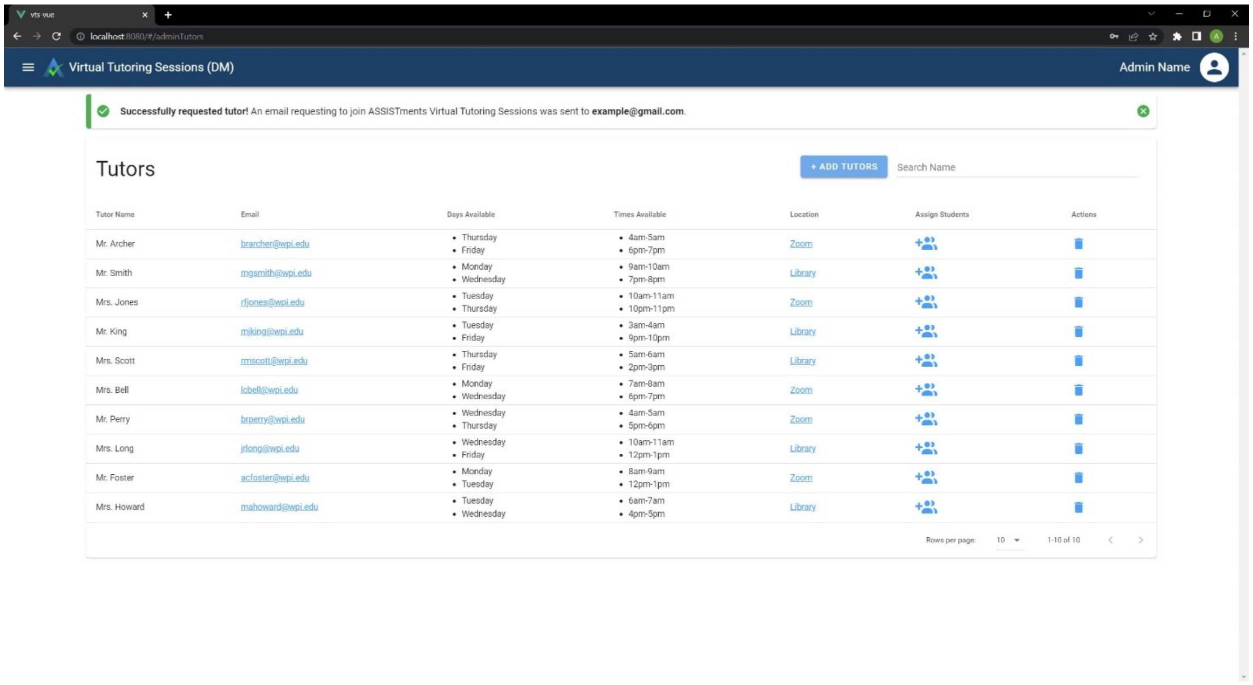
Oversight Dashboard

Search Date

Date	Start Time	End Time	Tutor	Student 1	Student 2	School Supervisor	Student 1 Location	Student 2 Location	Tutor Location
May 28, 2022	04:20 PM	05:20 PM	222	333	444	111	Zoom	Zoom	Zoom
May 26, 2022	07:15 AM	07:45 AM	222	333	444	111	Zoom	Zoom	Zoom
May 19, 2022	10:10 AM	12:10 PM	222	333	444	111	Zoom	Zoom	Zoom
May 17, 2022	04:30 PM	05:30 PM	222	333	444	111	Zoom	Zoom	Zoom
May 2, 2022	08:45 PM	09:45 PM	222	333	444	111	Zoom	Zoom	Zoom
May 4, 2022	05:20 AM	06:20 AM	222	333	444	111	Zoom	Zoom	Zoom

Rows per page: 10 1-6 of 6

Admin Tutors



vts vue

localhost:8080/#/adminTutors

Virtual Tutoring Sessions (DM)

Admin Name

Successfully requested tutor! An email requesting to join ASSISTments Virtual Tutoring Sessions was sent to **example@gmail.com**.

Tutors

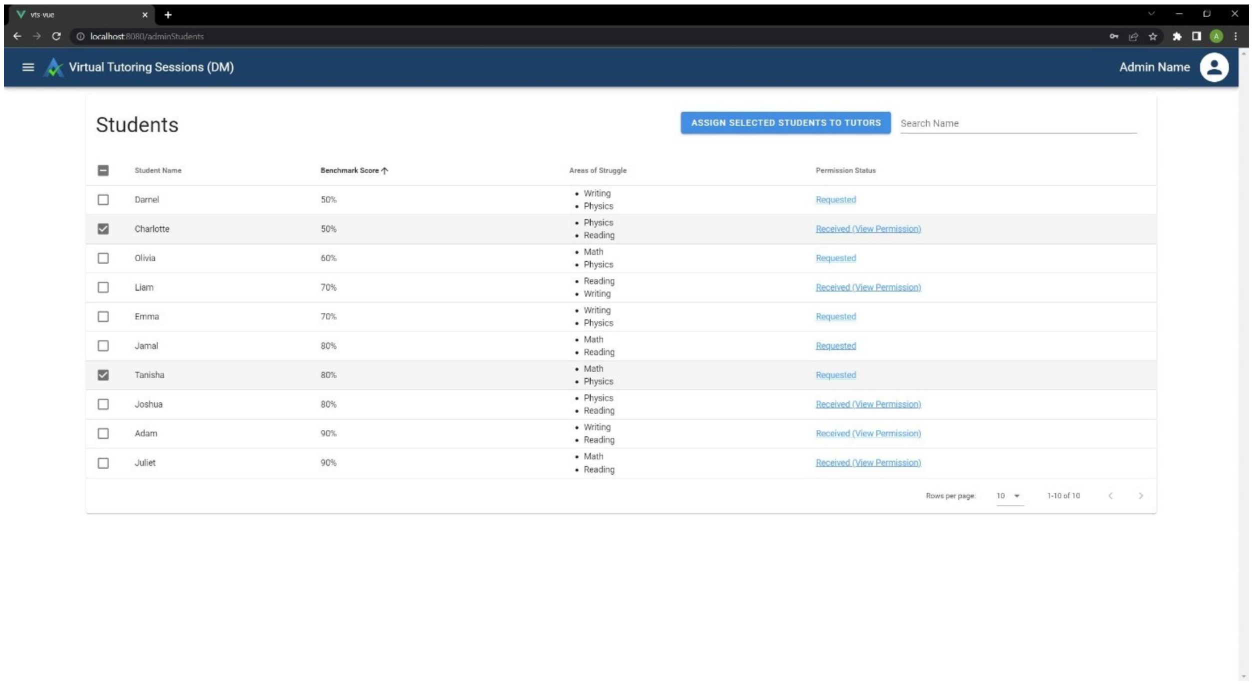
+ ADD TUTORS

Search Name

Tutor Name	Email	Days Available	Times Available	Location	Assign Students	Actions
Mr. Archer	brarcher@wpi.edu	• Thursday • Friday	• 4am-5am • 6pm-7pm	Zoom		
Mr. Smith	mgsmith@wpi.edu	• Monday • Wednesday	• 9am-10am • 7pm-8pm	Library		
Mrs. Jones	rfjones@wpi.edu	• Tuesday • Thursday	• 10am-11am • 10pm-11pm	Zoom		
Mr. King	mjking@wpi.edu	• Tuesday • Friday	• 3am-4am • 9pm-10pm	Library		
Mrs. Scott	rmscott@wpi.edu	• Thursday • Friday	• 5am-6am • 2pm-3pm	Library		
Mrs. Bell	lcbell@wpi.edu	• Monday • Wednesday	• 7am-8am • 6pm-7pm	Zoom		
Mr. Perry	brperry@wpi.edu	• Wednesday • Thursday	• 4am-5am • 5pm-6pm	Zoom		
Mrs. Long	jrlong@wpi.edu	• Wednesday • Friday	• 10am-11am • 12pm-1pm	Library		
Mr. Foster	acfoster@wpi.edu	• Monday • Tuesday	• 8am-9am • 12pm-1pm	Zoom		
Mrs. Howard	mahoward@wpi.edu	• Tuesday • Wednesday	• 6am-7am • 4pm-5pm	Library		

Rows per page: 10 1-10 of 10

Admin Students



vts vue

localhost:8080/adminStudents

Virtual Tutoring Sessions (DM)

Admin Name

Students

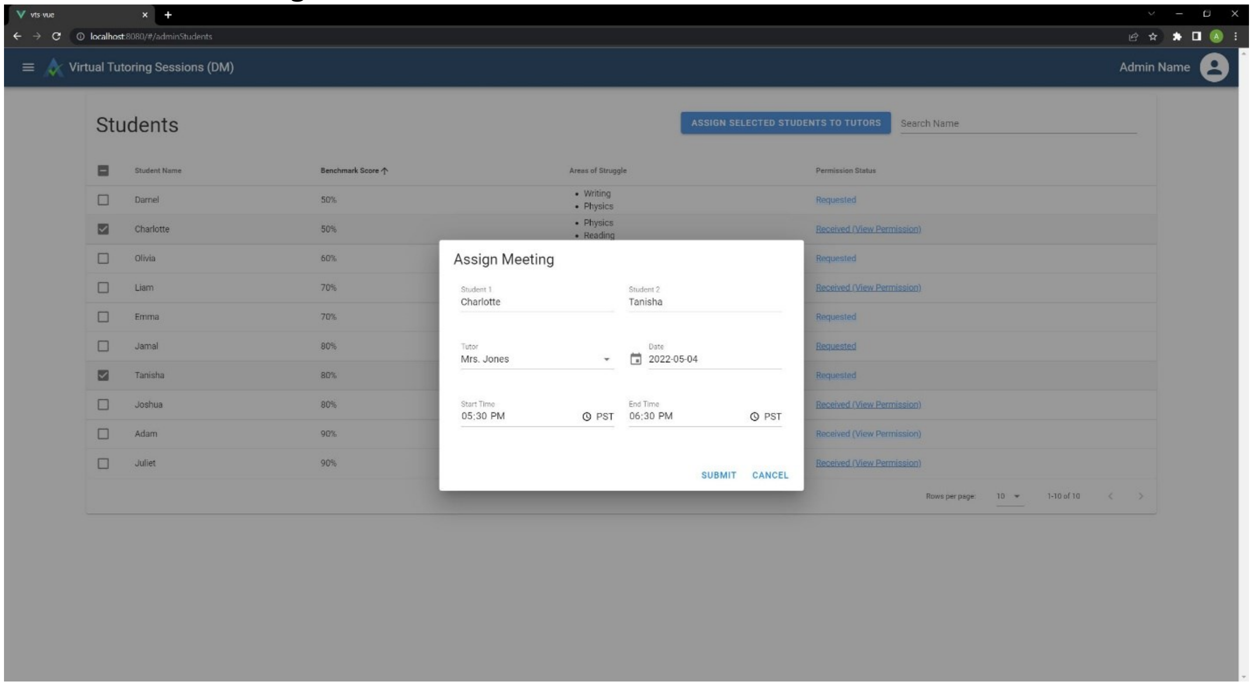
ASSIGN SELECTED STUDENTS TO TUTORS

Search Name

	Student Name	Benchmark Score ↑	Areas of Struggle	Permission Status
☐	Darnel	50%	• Writing • Physics	Requested
☑	Charlotte	50%	• Physics • Reading	Received (View Permission)
☐	Olivia	60%	• Math • Physics	Requested
☐	Liam	70%	• Reading • Writing	Received (View Permission)
☐	Emma	70%	• Writing • Physics	Requested
☐	Jamal	80%	• Math • Reading	Requested
☑	Tanisha	80%	• Math • Physics	Requested
☐	Joshua	80%	• Physics • Reading	Received (View Permission)
☐	Adam	90%	• Writing • Reading	Received (View Permission)
☐	Juliet	90%	• Math • Reading	Received (View Permission)

Rows per page: 10 1-10 of 10

Admin Create Meeting


vts-vue
localhost:8080/#/adminStudents
Virtual Tutoring Sessions (DM)
Admin Name
ASSIGN SELECTED STUDENTS TO TUTORS
Search Name
Rows per page: 10
1-10 of 10

Students

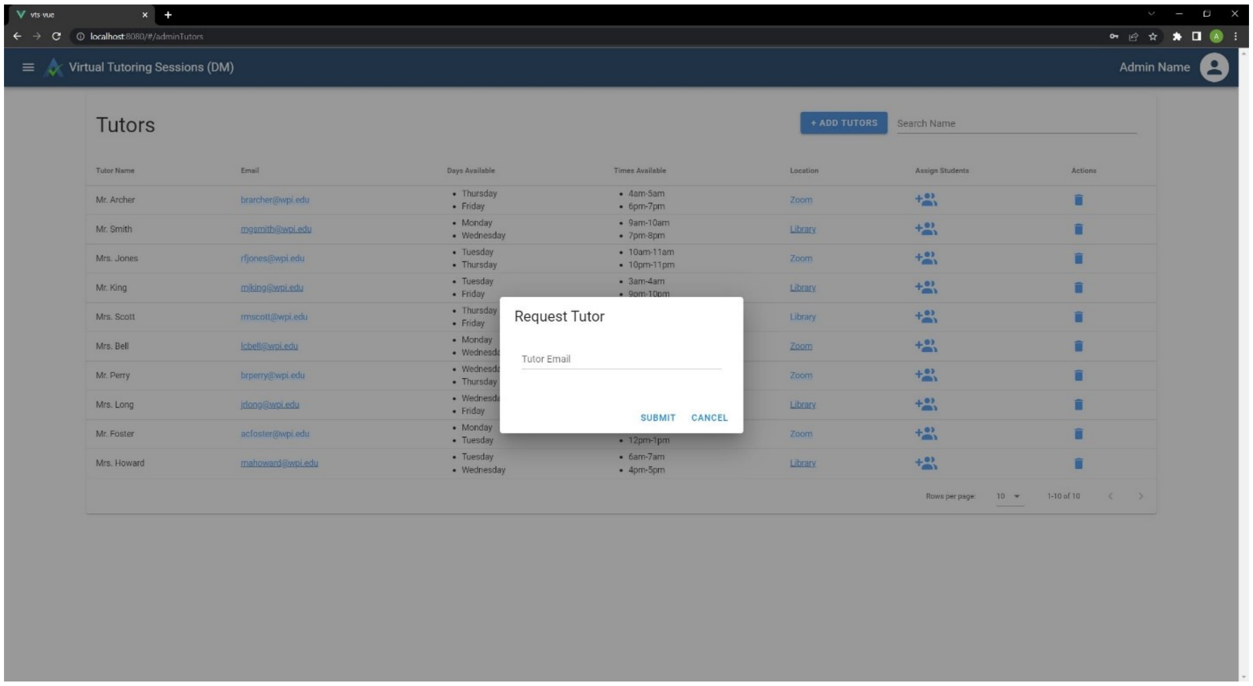
	Student Name	Benchmark Score	Areas of Struggle	Permission Status
☐	Darnel	50%	• Writing • Physics	Requested
☑	Charlotte	50%	• Physics • Reading	Received (View Permission)
☐	Olivia	60%		Requested
☐	Liam	70%		Received (View Permission)
☐	Emma	70%		Requested
☐	Jamal	80%		Requested
☑	Tanisha	80%		Requested
☐	Joshua	80%		Received (View Permission)
☐	Adam	90%		Received (View Permission)
☐	Juliet	90%		Received (View Permission)

Assign Meeting

Student 1: Charlotte
Student 2: Tanisha
Tutor: Mrs. Jones
Date: 2022-05-04
Start Time: 05:30 PM PST
End Time: 06:30 PM PST

SUBMIT CANCEL

Admin Add Tutor



Tutors

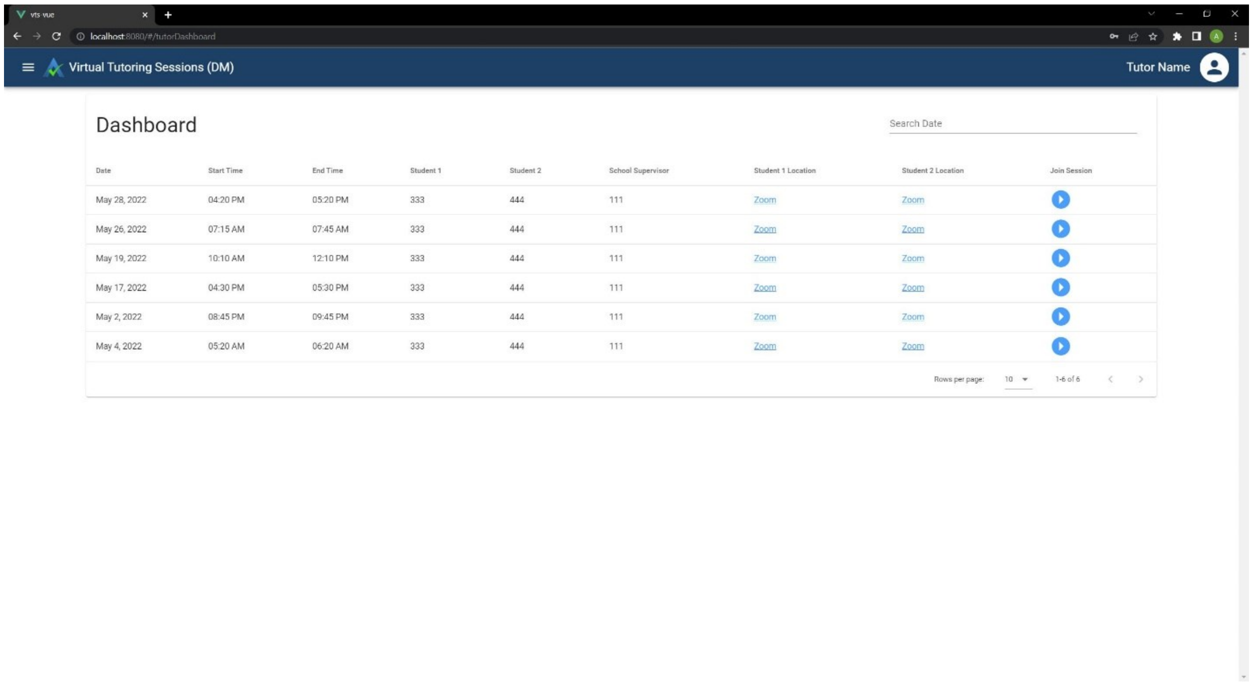
Tutor Name	Email	Days Available	Times Available	Location	Assign Students	Actions
Mr. Archer	brarcher@wpi.edu	• Thursday • Friday	• 4am-5am • 6pm-7pm	Zoom		
Mr. Smith	mgsmith@wpi.edu	• Monday • Wednesday	• 9am-10am • 7pm-8pm	Library		
Mrs. Jones	rfjones@wpi.edu	• Tuesday • Thursday	• 10am-11am • 10pm-11pm	Zoom		
Mr. King	mjking@wpi.edu	• Tuesday • Friday	• 3am-4am • 9pm-10pm	Library		
Mrs. Scott	rmscott@wpi.edu	• Thursday • Friday		Library		
Mrs. Bell	lcbell@wpi.edu	• Monday • Wednesda		Zoom		
Mr. Perry	brperry@wpi.edu	• Wednesda • Thursday		Zoom		
Mrs. Long	jdlong@wpi.edu	• Wednesda • Friday		Library		
Mr. Foster	acfoster@wpi.edu	• Monday • Tuesday	• 12pm-1pm	Zoom		
Mrs. Howard	mahoward@wpi.edu	• Tuesday • Wednesday	• 6am-7am • 4pm-5pm	Library		

Request Tutor

Tutor Email

SUBMIT CANCEL

Tutor Dashboard


vts vue
localhost:8080/#/tutorDashboard
Virtual Tutoring Sessions (DM)
Tutor Name

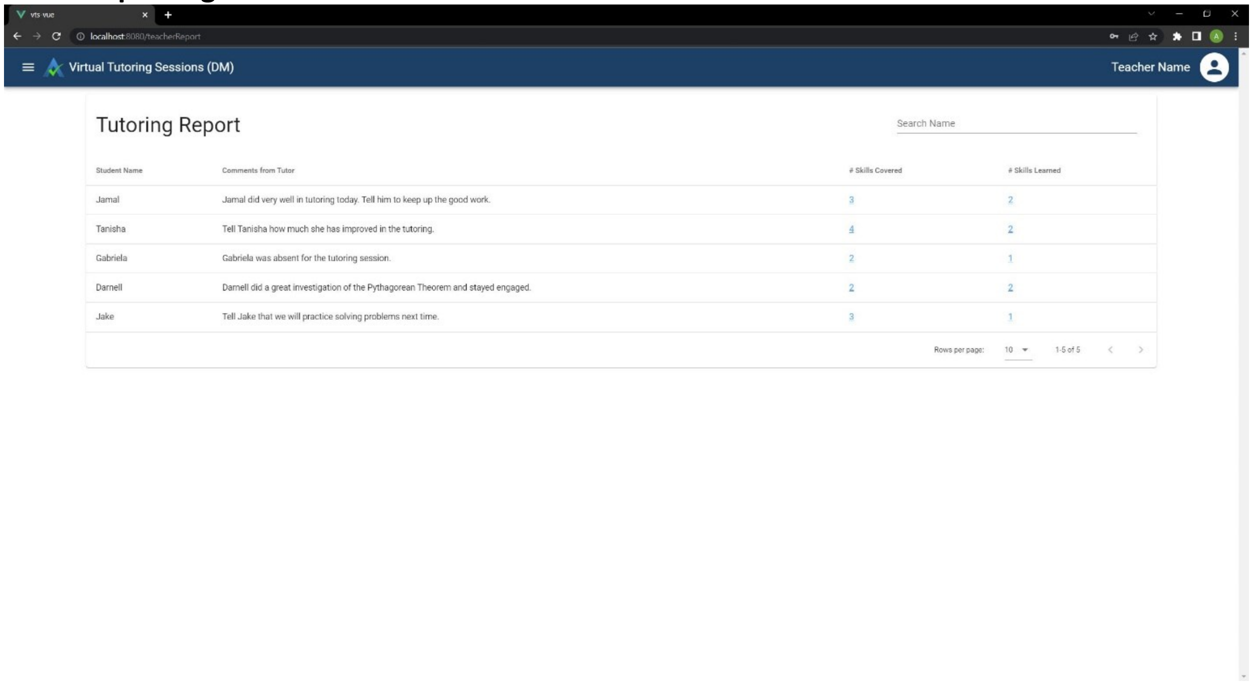
Dashboard

Search Date

Date	Start Time	End Time	Student 1	Student 2	School Supervisor	Student 1 Location	Student 2 Location	Join Session
May 28, 2022	04:20 PM	05:20 PM	333	444	111	Zoom	Zoom	▶
May 26, 2022	07:15 AM	07:45 AM	333	444	111	Zoom	Zoom	▶
May 19, 2022	10:10 AM	12:10 PM	333	444	111	Zoom	Zoom	▶
May 17, 2022	04:30 PM	05:30 PM	333	444	111	Zoom	Zoom	▶
May 2, 2022	08:45 PM	09:45 PM	333	444	111	Zoom	Zoom	▶
May 4, 2022	05:20 AM	06:20 AM	333	444	111	Zoom	Zoom	▶

Rows per page: 10 1-6 of 6

Tutor Reporting



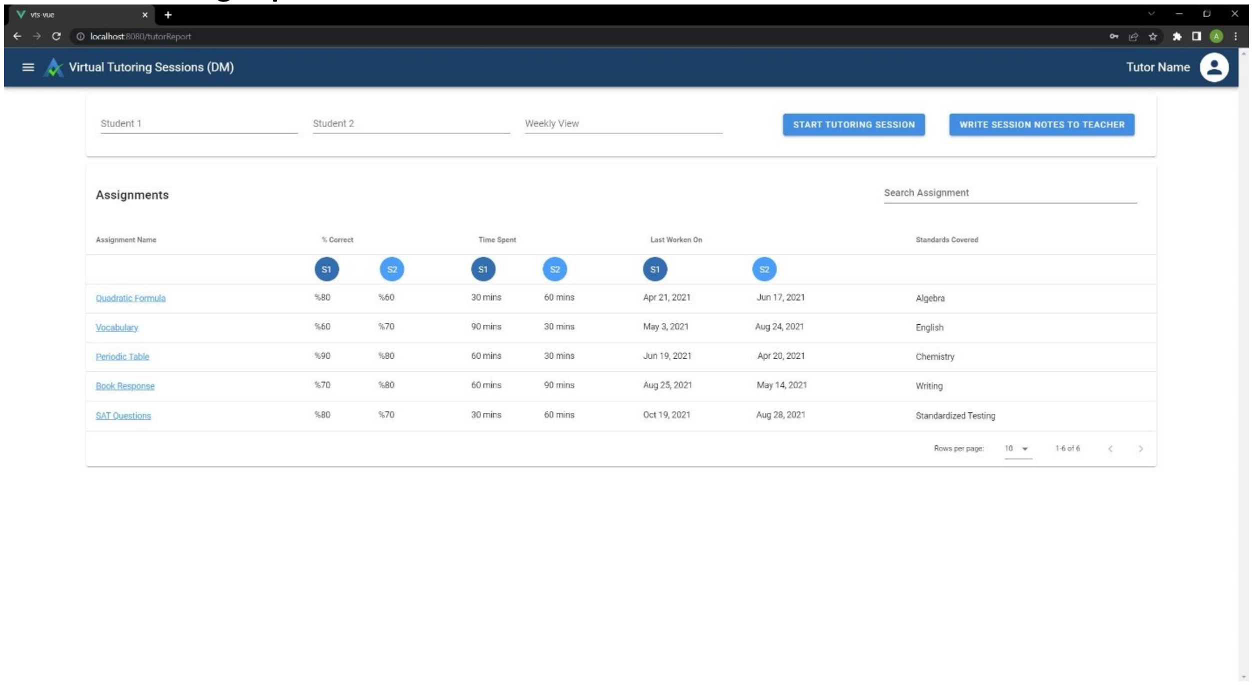
Tutoring Report

Search Name

Student Name	Comments from Tutor	# Skills Covered	# Skills Learned
Jamal	Jamal did very well in tutoring today. Tell him to keep up the good work.	3	2
Tanisha	Tell Tanisha how much she has improved in the tutoring.	4	2
Gabriela	Gabriela was absent for the tutoring session.	2	1
Darnell	Darnell did a great investigation of the Pythagorean Theorem and stayed engaged.	2	2
Jake	Tell Jake that we will practice solving problems next time.	3	1

Rows per page: 10 1-5 of 5

Teacher Tutoring Report



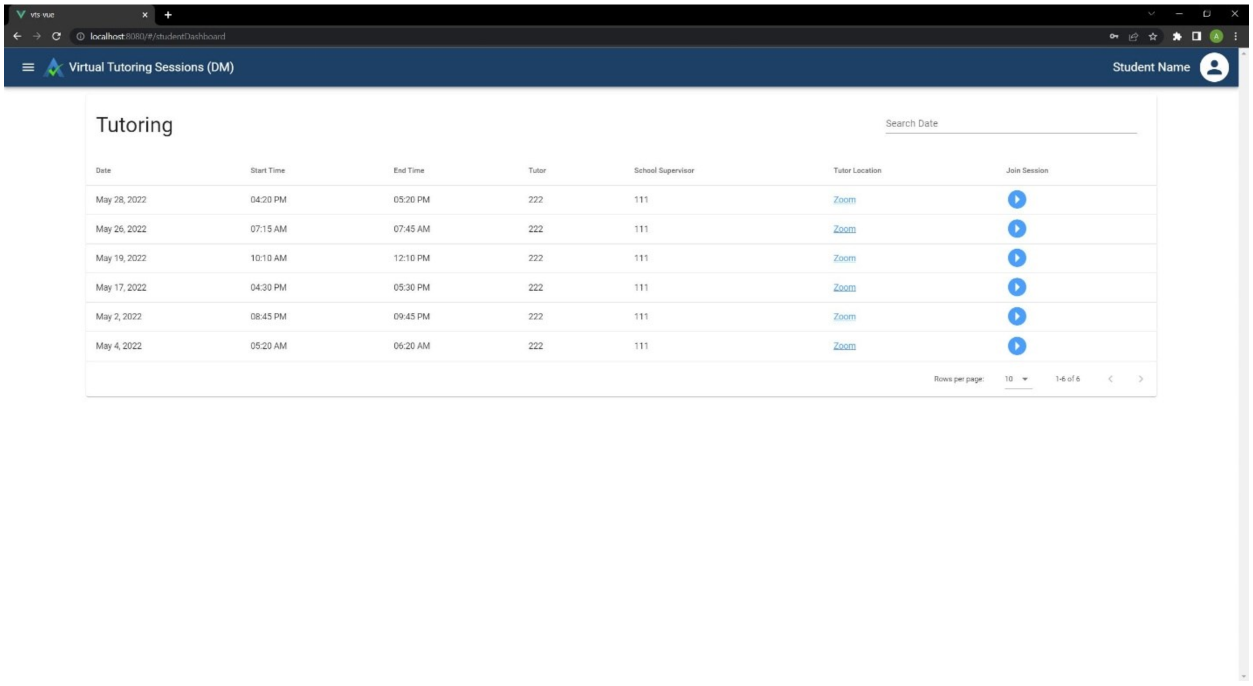
vts-vue
localhost:8080/tutorReport
Virtual Tutoring Sessions (DM)
Tutor Name
Student 1
Student 2
Weekly View
START TUTORING SESSION
WRITE SESSION NOTES TO TEACHER
Search Assignment

Assignments

Assignment Name	% Correct		Time Spent		Last Worken On		Standards Covered
	S1	S2	S1	S2	S1	S2	
Quadratic Formula	%80	%60	30 mins	60 mins	Apr 21, 2021	Jun 17, 2021	Algebra
Vocabulary	%60	%70	90 mins	30 mins	May 3, 2021	Aug 24, 2021	English
Periodic Table	%90	%80	60 mins	30 mins	Jun 19, 2021	Apr 20, 2021	Chemistry
Book Response	%70	%80	60 mins	90 mins	Aug 25, 2021	May 14, 2021	Writing
SAT Questions	%80	%70	30 mins	60 mins	Oct 19, 2021	Aug 28, 2021	Standardized Testing

Rows per page: 10 1-6 of 6

Student Dashboard



vts-vue
localhost:8080/#/studentDashboard
Virtual Tutoring Sessions (DM)
Student Name
Search Date

Tutoring

Date	Start Time	End Time	Tutor	School Supervisor	Tutor Location	Join Session
May 28, 2022	04:20 PM	05:20 PM	222	111	Zoom	▶
May 26, 2022	07:15 AM	07:45 AM	222	111	Zoom	▶
May 19, 2022	10:10 AM	12:10 PM	222	111	Zoom	▶
May 17, 2022	04:30 PM	05:30 PM	222	111	Zoom	▶
May 2, 2022	08:45 PM	09:45 PM	222	111	Zoom	▶
May 4, 2022	05:20 AM	06:20 AM	222	111	Zoom	▶

Rows per page: 10 1-6 of 6