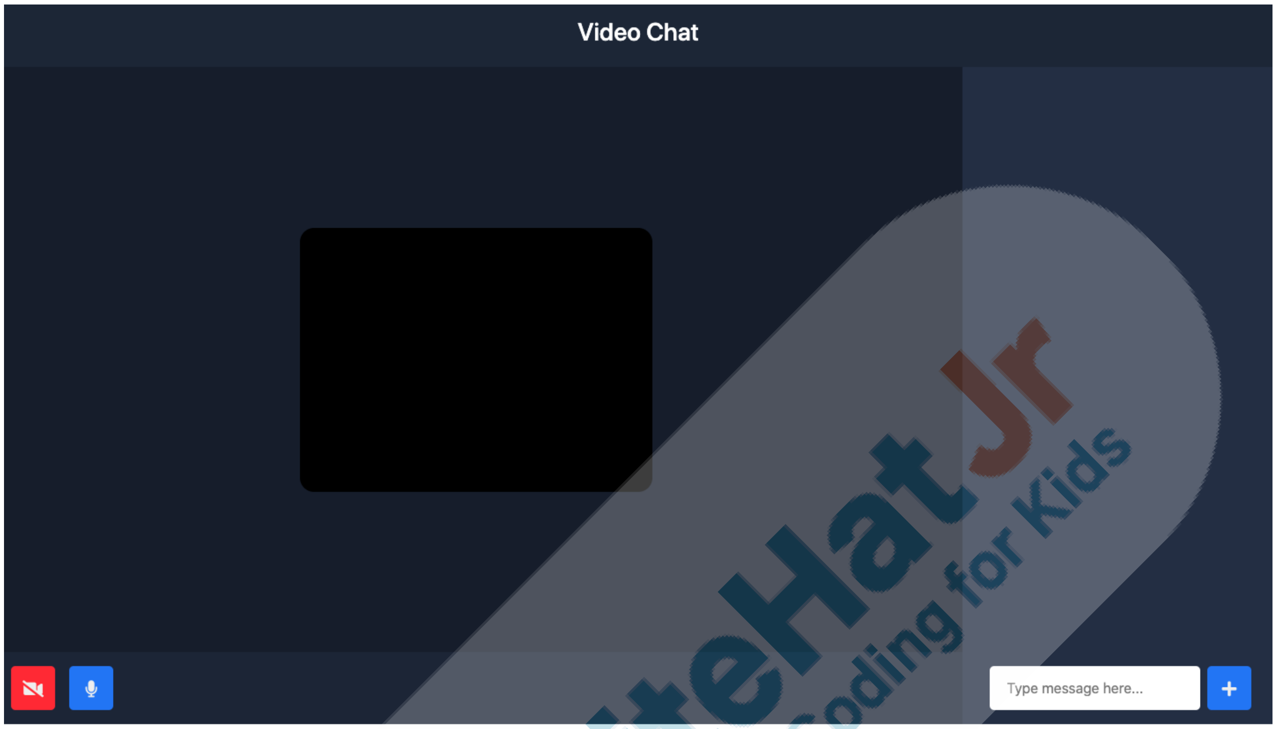
Topic	Video Chat App - Mailer	
Class Description	**Student will be integrating the mailer API with the client and testing the functionality**	
Class	**C-222**	
Class time	**45 mins**	
Goal	● Integrating the mailer API with the client ● End to end testing of the application	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code	
Class structure	**Warm-Up** **Student - led Activity 1** **Wrap-Up**	**10 mins** **30 mins** **5 mins**

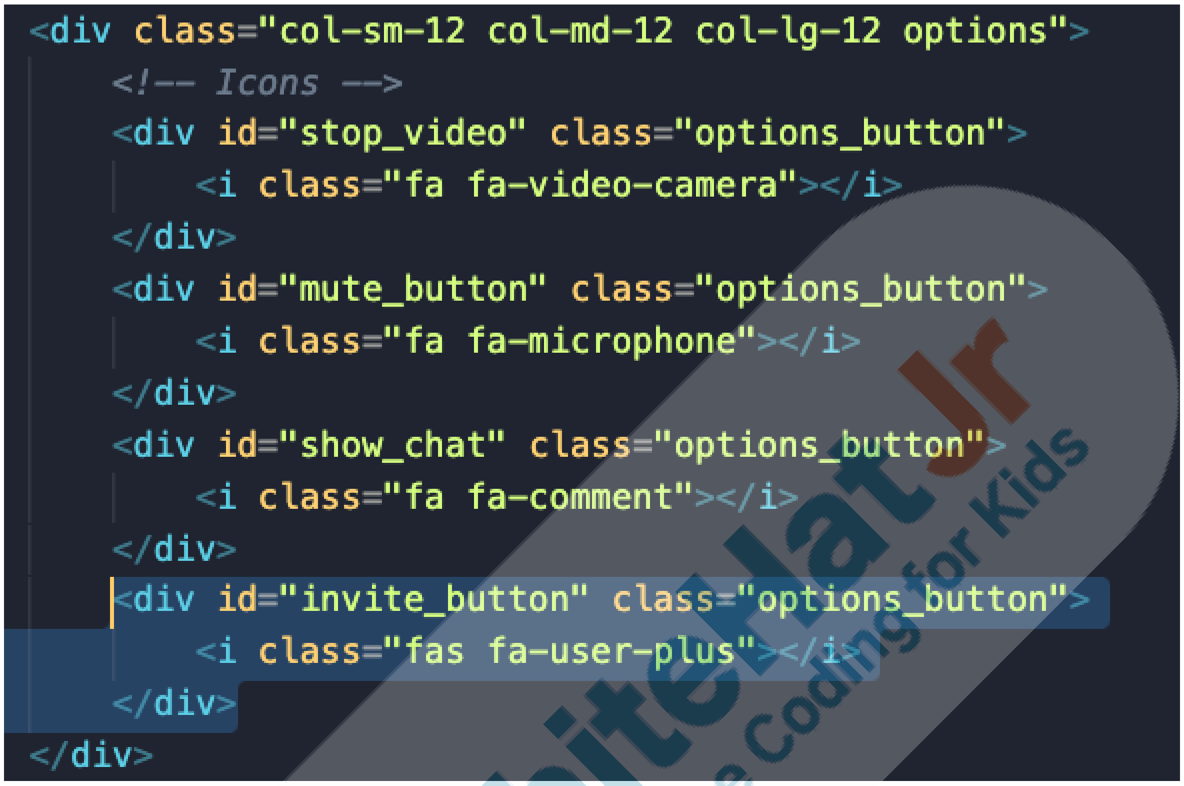
WARM UP SESSION - 10mins	
Teacher Action	**Student Action**
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?	**ESR**: Hi, thanks, yes, I am excited about it!

Q&A Session	
Question	**Answer**
What is the purpose of JSON? A. For transmit data B. Database C. Data Management D. None of the above	**A**
What URL stands for? A. Uniform Researh Locator B. Uniform Resource Locator C. Uniform Resource Location D. None of the above	**A**
STUDENT-LED ACTIVITY - 30mins	
Student Initiates Screen Share	
<u>ACTIVITY</u> • **Understanding about WebRTC and it's functions** • **Fetching the audio and the video for the chat app from the user's browser**	
Teacher Action	**Student Action**
This is a student-led class where all activities should only be performed by the student on the repository that was	

updated on Render. The teacher is expected to guide the student on the code, explanations and steps.	
In the last class, we started building a mailer using the ***nodemailer*** library offered by NodeJS. We created our own App Password to use gmail services and learnt about SMTP. We also created a transporter that could transport our mails through the SMTP server, and we finally created a ***POST*** api, that takes - 1. URL - the url of the room for the receiver to join 2. To - the email ID of the receiver Everything looks good so far! In today's class, we will be integrating this API with our client so that anyone can invite their friends and family to video chat with them. Let's get started then!	
Now, in order to integrate the mailer API that we have just created, we will need a button on our UI client side, and it's click event. Currently, our UI looks like this -	

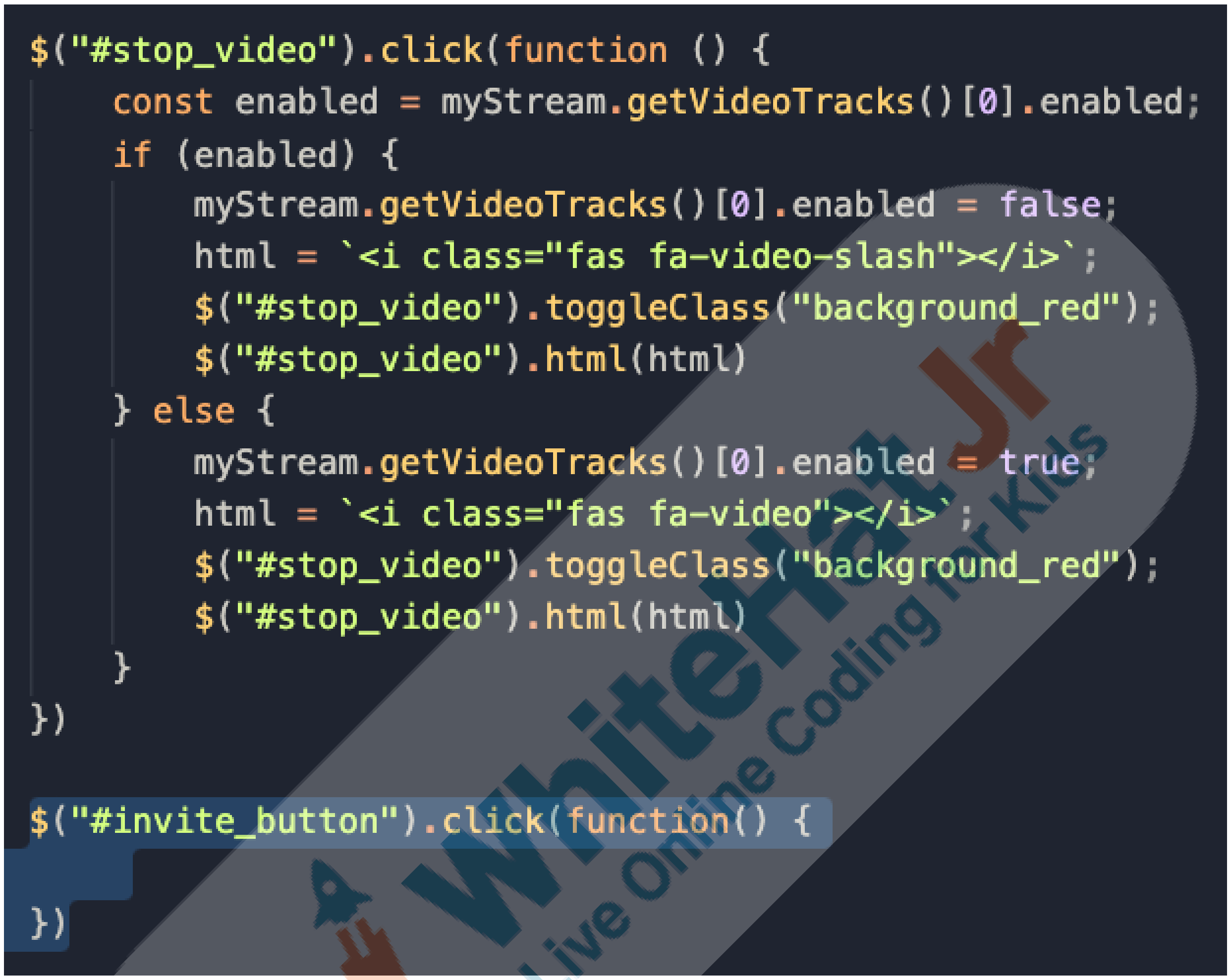


Below, we can see that we have buttons to close our audio and videos! Let's add one more button there, that will be to invite someone to our room. We will make the changes in ***index.ejs*** file - *Teacher helps the student in writing the code*	*Student writes the code*



```
<div class="col-sm-12 col-md-12 col-lg-12 options">
    <!-- Icons -->
    <div id="stop_video" class="options_button">
        <i class="fa fa-video-camera"></i>
    </div>
    <div id="mute_button" class="options_button">
        <i class="fa fa-microphone"></i>
    </div>
    <div id="show_chat" class="options_button">
        <i class="fa fa-comment"></i>
    </div>
    <div id="invite_button" class="options_button">
        <i class="fas fa-user-plus"></i>
    </div>
</div>
```

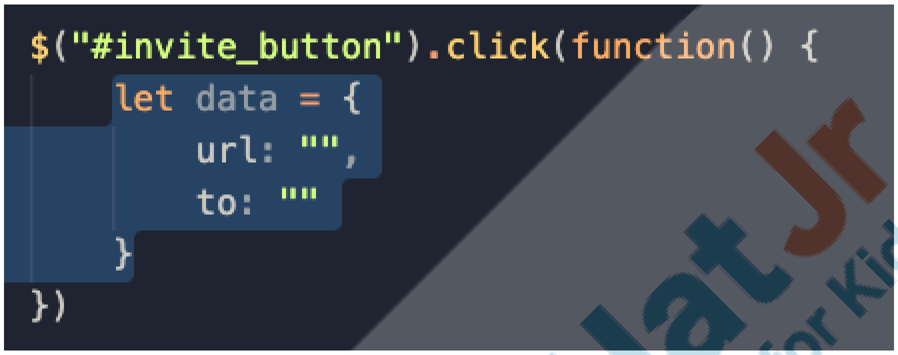
With all the buttons that we already had, we have now added one more button with ***id*** as ***invite_button***, and have given it the same class called ***options_button*** so it looks similar to the other buttons. We have also used the class ***fa-user-plus*** which would be appropriate for our invite button! Next, we will need to create a ***click*** event handler on this button that we just created in our ***script.js*** Let's do that - *Teacher helps the student in writing the code*	*Student writes the code*



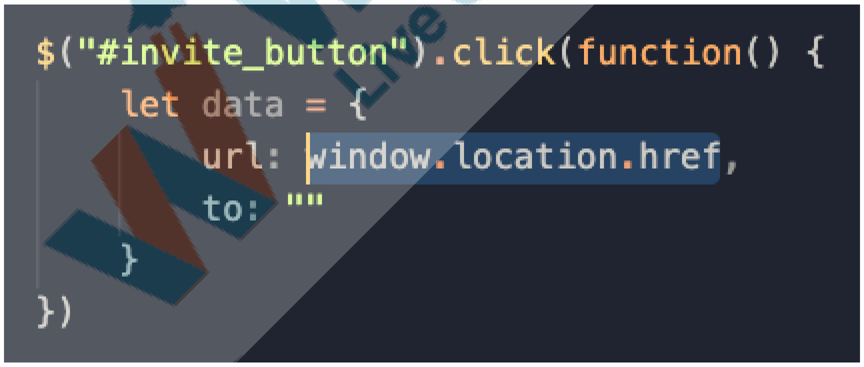
```
$("#stop_video").click(function () {
    const enabled = myStream.getVideoTracks()[0].enabled;
    if (enabled) {
        myStream.getVideoTracks()[0].enabled = false;
        html = `<i class="fas fa-video-slash"></i>`;
        $("#stop_video").toggleClass("background_red");
        $("#stop_video").html(html)
    } else {
        myStream.getVideoTracks()[0].enabled = true;
        html = `<i class="fas fa-video"></i>`;
        $("#stop_video").toggleClass("background_red");
        $("#stop_video").html(html)
    }
})

$("#invite_button").click(function() {

})
```

Inside the dollar function ***$(function(){})*** where we have event handlers for all our buttons, we are creating an event handler for clicks on the ***invite_button*** (based on the ID of the button). Okay, now can you recall the 2 things our email api requires from the client?	**ESR:** 1. URL of the room 2. Email ID of the receiver

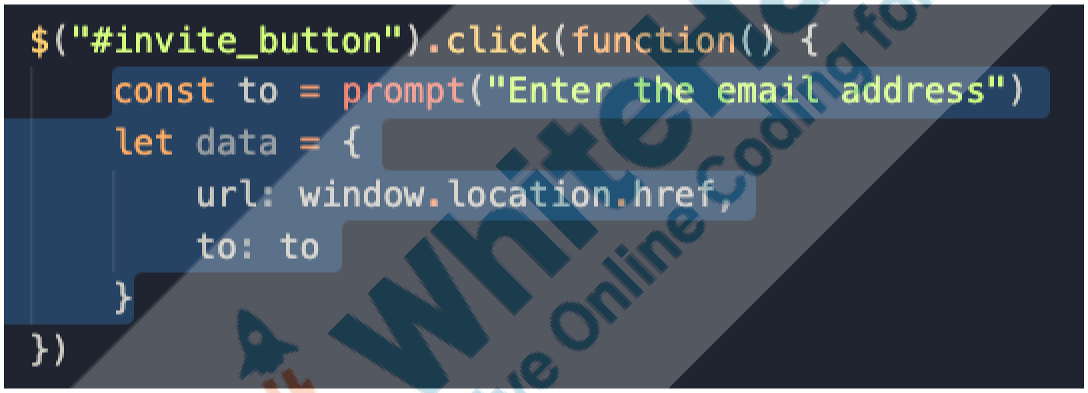
Great! So, these are the 2 things that we need to get! Let's create an object where we can save these 2 things - *Teacher helps the student in writing the code*	*Student writes the code*


`$("#invite_button").click(function() {`
`    let data = {`
`        url: "",`
`        to: ""`
`    }`
`})`	

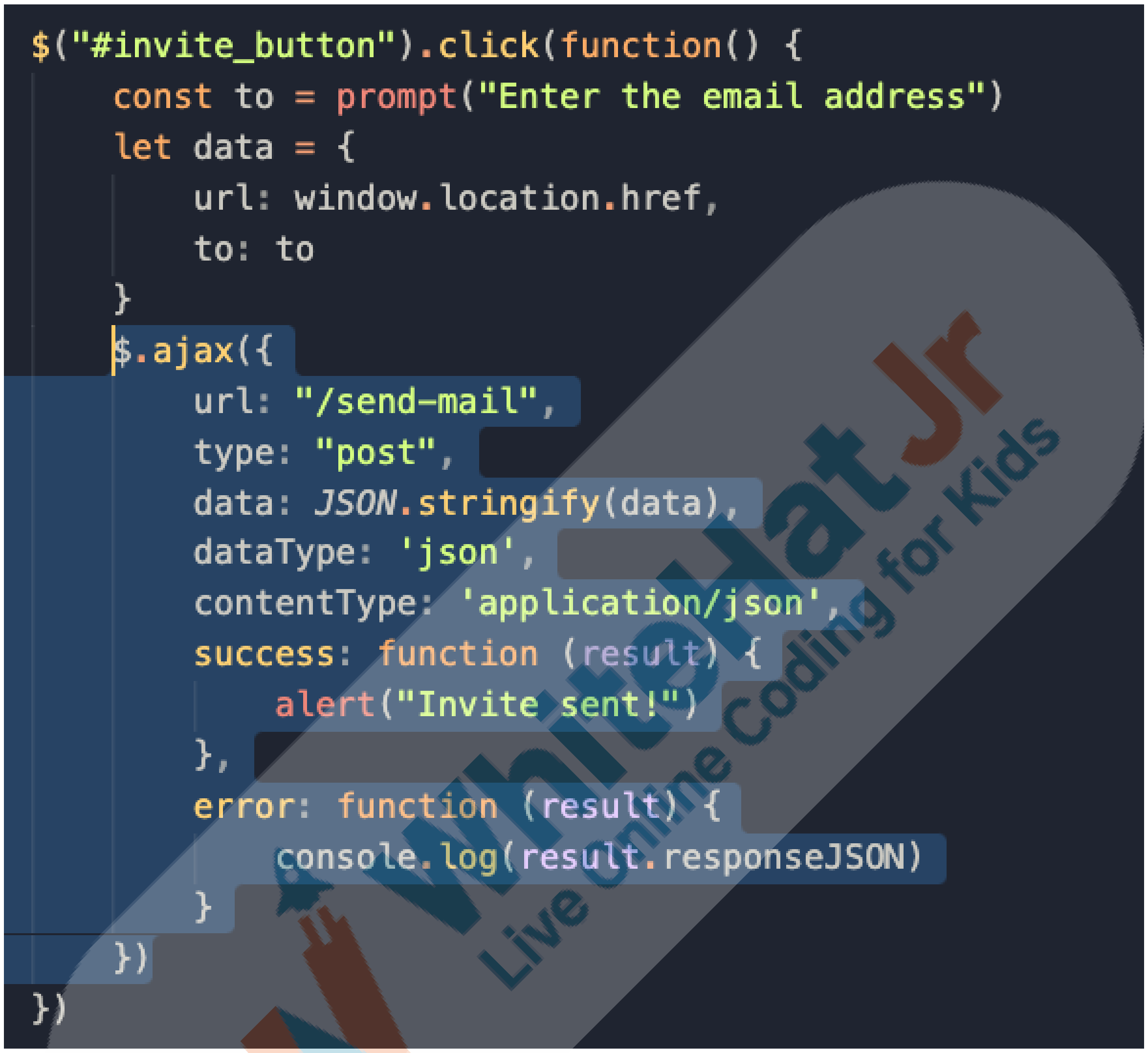
Inside this, let's first write the code to fetch the current URL of the page. This is the URL with the unique ID of the room, and if our friend joins this URL, then they can enter our room so this will be the URL we would want to share with them to invite them - *Teacher helps the student in writing the code*	*Student opens the file*


`$("#invite_button").click(function() {`
`    let data = {`
`        url: window.location.href,`
`        to: ""`
`    }`
`})`	

Just how we use ***window.location.href*** to change the URL of the page in JavaScript, it contains the URL of the current page. It's value would be our URL that we want to use!	

Next, we want to retrieve the email ID of the person to whom we want to send the invite! Do you have any ideas on how we can do that? That's right! It's just how we take the name of the user when we open the page. *Teacher helps the student in writing the code*	**ESR:** We can use the ***prompt()*** function of JavaScript! *Student writes the code*
	
Now all that's left to do is to make a ***post*** request to the server on ***/send-mail*** API! For that, we can use the ***AJAX*** request! Let's do that - *Teacher helps the student in writing the code*	*Student writes the code*

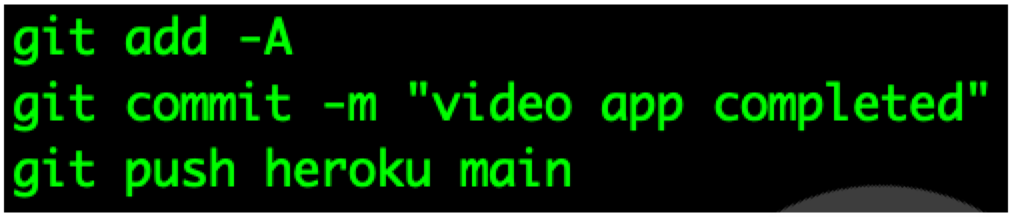
```
$("#invite_button").click(function() {
    const to = prompt("Enter the email address")
    let data = {
        url: window.location.href,
        to: to
    }
})
```



```
$("#invite_button").click(function() {
    const to = prompt("Enter the email address")
    let data = {
        url: window.location.href,
        to: to
    }
    $.ajax({
        url: "/send-mail",
        type: "post",
        data: JSON.stringify(data),
        dataType: 'json',
        contentType: 'application/json',
        success: function (result) {
            alert("Invite sent!")
        },
        error: function (result) {
            console.log(result.responseJSON)
        }
    })
})
```

Here is how the Ajax requests work! The ***$.ajax()*** function takes an object with all the details about the request. We first tell it about the ***URL*** to which we want to send the request, which in our case, it ***/send-mail***. Next, we tell it about the ***type*** of request that we are trying to make, which is ***post*** for us.	

Next, we define if we want to send some ***data*** with the request. Now make sure that ***AJAX*** only sends the data in the form of a string, so we are using ***JSON.stringify()*** function to convert our ***data*** object to string. Next, we tell it about the ***dataType***, which is JSON so that it can tell the server to interpret the data as a ***JSON*** when the request is made. We also tell it about the ***contentType***, to define what kind of content we are sending. It's value is ***application/json*** since it's a ***JSON*** coming out of an application. Finally, we have the ***success*** and ***error*** handlers of our AJAX request, which are just simple functions. In case of ***success***, we are using the ***alert()*** function to tell the user that the invite has been sent successfully. In case of ***error***, we are just logging the ***responseJSON*** from the error. With this, the functionality of our app is completed! Let's deploy our App on Renderw with the following commands on the project repository - ***git add .*** ***git commit -m "video app complete"*** ***git push***	
Teacher helps the student in opening the command prompt/terminal, navigating to the project directory that is deployed on Render and run the following commands	*Student opens the command prompt/terminal, navigate to project directory that is deployed on Render and runs the commands*



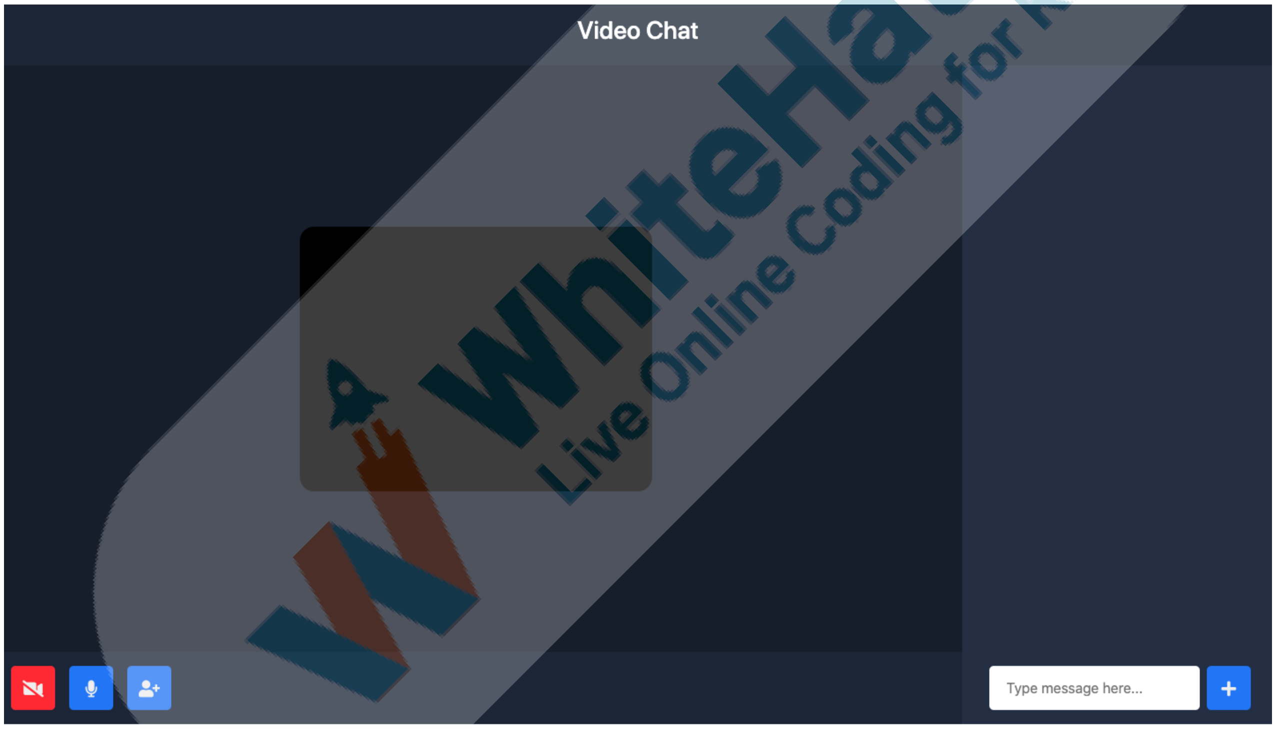
```
git add -A
git commit -m "video app completed"
git push heroku main
```

Now let's test the functionality!

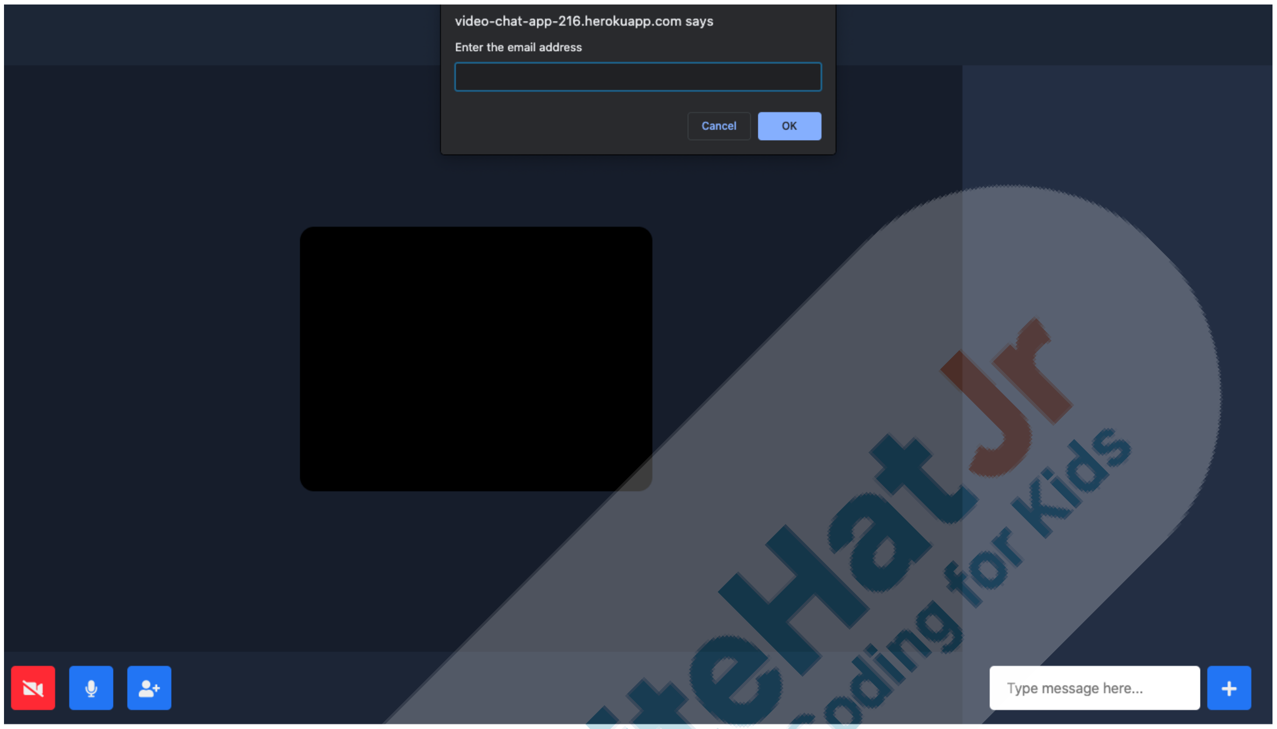
Open your Render app on the browser -

Teacher guides the student in opening the app on the browser

Student follows the instructions

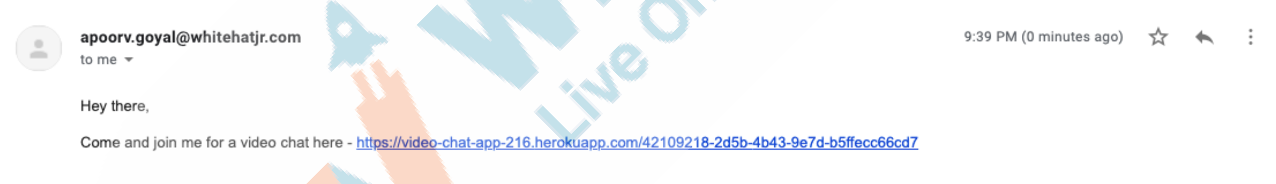


Click on the invite button so it asks you the email ID of the receiver -


video-chat-app-216.herokuapp.com says

Enter the email address

Ask the student to enter your email ID, and check if you received the email.	


apoorv.goyal@whitehatjr.com
to me

9:39 PM (0 minutes ago)

Hey there,

Come and join me for a video chat here - https://video-chat-app-216.herokuapp.com/42109218-2d5b-4b43-9e7d-b5ffecc66cd7

Click on the link and verify if you both have joined the same room!	
With this, we have completed all the functionalities of our video chat application! In this application, we also learnt how we can deploy the app on a remote server and learnt how we can deploy it with b.	

We also learnt about WebRTC, PeerJS and built this app using NodeJS! I hope you got to learn a lot while building this application, and had fun. You just successfully built your first full fledged networking application! In the next class, we will be looking into cyber security and deep dive into what vulnerabilities are in applications!	
Teacher Guides Student to Stop Screen Share	
WRAP UP SESSION - 5 Mins	
Quiz time - Click on in-class quiz	
Question	**Answer**
What is nodemailer? A. For email sending B. Import module C. For data D. None of the above	**A**
What does href mean? A. Specifies the Url B. Generate Url C. Find the Url D. None of the above	**A**

What does AJAX stand for? A. Asynchronous JqueryScript And XML B. Asynchronous JavaScript And XML C. Asymmetric JavaScript And XML D. None of the above	**B**
End the quiz panel	
<u>FEEDBACK</u> ● **Appreciate the students for their efforts in the class.** ● **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**	
Teacher Action	**Student Action**
You get Hats off for your excellent work! In the next class	*Make sure you have given at least 2 Hats Off during the class for:* Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
Project Discussion	
Teacher Clicks ✖ End Class	

ADDITIONAL ACTIVITIES	
Additional Activities *Encourage the student to write reflection notes in their reflection journal using markdown.* Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	*The student uses the markdown editor to write her/his reflections in the reflection journal.*

ACTIVITY LINKS		
Activity Name	**Description**	**Link**
Teacher Activity 1	Previous Class Code	https://github.com/pro-whitehatjr/PRO-C221-Reference-Code
Teacher Activity 2	Reference Code	https://github.com/pro-whitehatjr/PRO-C222-Reference-Code
Student Activity 1	Previous Class Code	https://github.com/pro-whitehatjr/PRO-C221-Referen

		ce-Code