
Personal Report

Students:

Jiahe Zhang 20372118
UCD 20205722

December 8, 2022



1 General Overview

1.1 Team role

Core team manager, head of the main program development part of the team. 2+2+1 rotational system in the role of the "1" resident in the front line of development and management. In sprint1 (09.07-09.18), sprint2 (09.19-10.02), sprint3 (10.03-10.16), sprint4 (10.17-10.30), sprint5 (10.31-11.13), sprint6 (11.14-11.27), act as The team progress coordinator, acting as team leader at the start of the project in sprint 1 and in a supporting role to the rotating team leader for all subsequent sprints.

1.2 Main contributions

1.2.1 Development

- Complete the whole process of developing the main part of the video conference alone.
- Almost complete the whole process of developing the chat box function alone.
- Completed the project framework almost alone.
- Participated in the development of the whiteboard function.
- Configuring docker.

1.2.2 Management

- Act as a team leader or top management role in multiple sprints, organising meetings and supervising the smooth running of phases.
- Made several decisions that were beneficial to the team's progress, such as establishing a Microsoft Team-based online collaboration model in the first week, and finalising decisions on the use of Agora, Pubnub and Firebase during the development process.
- Leading the development of the project plan and the project process model in the first sprint and pushing it through at a later stage.
- Integration of a group REPORT done almost alone.
- Propose project process models and in-iteration planning.
- Proposes a 2+2+1 division of labour model.
- Risk assessment.
- Each sprint will collate relevant content for categorisation, progress reports, Sunday meeting minutes and Thursday for collaborative development, group charter writing, and team activities at specific times or stages after completion of tasks.
- Needs analysis, especially for the main body of the video conference.
- The prototype is drawn.

2 Development log

2.1 Brief description

I have undertaken most of the development work in this group, including the building of the flask project, the full stack development of the main part of the Agora webRTC based audio and video conferencing, the full stack development of the Agora RTM based chat room, and the development of the real-time interactive whiteboard based on Firebase and Pubnub. Throughout the process, the knowledge learnt in the classroom was put into practice, and the development work was carried out in strict accordance with the full process of dynamically determining requirements, developing, testing and modifying feature points in a single cycle, and successfully completing the main video conferencing service and the whiteboard collaboration service. I have attached below a simplified development log for the audio/video service and the chat room section, which describes my general development experience and the problems I encountered and their solutions.

2.2 sprint1(09.07-09.18)

2.2.1 09.11 (end of group meeting on Sunday)

Combined with the group meeting for the overall project requirements brainstorming, importance analysis, etc., combined with the actual situation and project requirements, the initial establishment of the main video conferencing function by my responsibility, proceeded to carry out further requirements analysis and integration.

2.2.2 09.12

Defining a two-step development direction.

- Phase 1: Testing feasibility mainly, with the ultimate goal of a simple peer to peer video calling application. Have a lobby to access a designated room and invite others to the same room with a room ID. The room can be used for video calls, microphone and camera control and exit.
- Phase 2: A sophisticated, group video conference that can support multiple people online. Has all the content of phase one. A new shared screen, online member display, live chat box. And to provide comfort for users, it allows them to adjust the main view at will (inspired by the Twitter space). When a user joins a room, a live image of all the online users appears in a small circle and the user can switch the main view with a click.

2.2.3 09.15 (centralised development)

After the technical certification, the initial decision was made to use webrtc for the implementation of audio and video calls and chats, which I had not been exposed to before, and further study of webrtc was carried out. It allows us to establish a peer-to-peer connection between two browsers in order to exchange data such as audio and video in real time, so this is real-time communication. The connection is therefore made between two browsers and the data transferred between the browsers never actually reaches the server, so this does not mean that we do not need the involvement of the server at all, assuming we have two clients, once the connection is established the data is transferred directly without having to reach the server, so this makes webrtc ideal for exchanging audio and video, as any increased latency due to having to access the server first would actually result in a slight delay.

2.3 sprint2(09.19-10.02)

2.3.1 09.22 (centralised development)

- Overturning the development of webrtc-based audio and video conferencing established in sprint1: first of all webrtc uses udp, udp is not a reliable protocol for transmitting important data, the way udp works is that it sends data very quickly but it never verifies that the data has been received, so if we are sending something like audio or video data, if we lose a few frames it's not a big deal. If we're sending a file to someone, if we lose a few bytes of data the whole file will be corrupted, so because udp is unreliable.
- It was decided to develop it using a combination of webrtc + websockets: a message is sent between two peers, a connection is established and then webrtc takes over. Two peers, if they want to communicate and one peer initiates a connection, they have to send this message to peer 2. So they will send a message, which can be done by signalling, that will contain some information about this peer, so they have to send their network information to peer 2, now if peer 2 accepts this offer they will go on to send some information back to peer 1, the way they communicate now is not relevant, once the information has been exchanged the data can start to flow. The two peers do not need the server to be involved, they can start transferring data, so this can be audio and video data and the connection is already established.
- Feasibility check: Using Google's stunt server, two side-by-side tabs were opened and a connection was successfully created between the two peers. First create an offer, this is what an offer looks like, copy the whole offer, I'm going to paste this into peer 2. manually, won't do this in the real application, but this way you can see the exchange process, so peer 2 will get the offer and then we will create an answer, so whenever we add an offer from peer 1, let's go ahead and create from peer 2 an answer. So now we have an answer, you can see the object, one for the offer and one for the answer, copy the whole answer and send it to peer 2.
- **Test result: success.**

2.3.2 09.25

- Project 1 Requirement Analysis: a room that has my console with full control of my camera, I can toggle this switch with the microphone, leave this room, create a room, add any value, give it a name, assume I want to share this URL with someone, assume they try to join this room but there are no additional parameters here so there is no room name, it will redirect them to the lobby. Both peers can see each other, mute peer 1's camera and the whole camera screen will be hidden. The whole camera screen will be hidden here so that the user I am now in is always in the top left corner and peer 2, the person who is talking, is always in the big picture, and if this peer's camera is muted, nothing will be seen here. If you unmute it, you will see the video frame so that it goes both ways and both parties have full control here, so if I am in this room, or if peer 1 is in their app and peer 2 leaves, the video frame will automatically go to full screen and peer 1 will leave and can join a room and leave it.
- Initialisation of index, including the overall architecture of index, the video framework. Where the video framework without group video calls, because the test part of project one for peer to peer meeting, the framework will be modified in project two. Create the container.

- Initialize css.
- Get the camera audio and video streams (local stream variables are created (local microphone audio and camera video data streams)) and the remote stream, once we connect to another user, will be the remote user's camera and audio data.
- Initialize the init function (asynchronous): when loading our page, request permission to access our camera video and audio, add this to the dom, for this we are going to call the local stream, then we are going to set this value to wait for the navigator mediadevices.getUserMedia.
- **Test result: success.** Access to the site shows a prompt for video permissions.

2.3.3 09.29 (centralised development)

- createOffer function (asynchronous): the first time we load our page and another peer joins, we have to create an offer and send it to them. Set the peer connection variable, essentially this is an interface that stores all the information between us and the remote peer.
- Get my local streams and add them to the peer connection: Loop through all the audio and video streams and actually add them to the connection so that my remote peers can actually get them. To get the tracks, we have to loop through each track (4) and go to the peer connection point to add the track, which is an object method. Listening to the peer's tracks (event listener), looping through each track of our remote peer and setting it up as a remote stream, passing it to the video container, peer connection.
- **Test result: success.** Set up a media stream inside this video player, inside the video tab, set up the media data or the actual media stream, then add data to it, call it and refresh the page in the background to see the offer of the string displayed normally in the peer connection (passed to the rtc peer connection with the help of Google stunt server)).

2.3.4 10.02

- Initialise the Agora room: download the files for the audio/video SDK and add the access js, enter the ID and token to open the service. Need a UID for each user, when joining a channel, this is how each user in the application identifies who is who, so that users can send messages to each other. Generate a random number, I will make sure it is a large number so we don't have some sort of chance of duplication. Create a channel, join that channel, send a message to that channel, set up our application ID, token UID, client and channel, and start triggering.
- Login: uid and token, sets the id of the application, throws our token to create the client, calls await, and then executes the client.
- **Test result: success.** Typing in the console you can see that the AgoraRtc service is imported normally and the user connects to the default channel successfully.

2.4 sprint3(10.03-10.16)

2.4.1 10.06 (centralised development)

- Create channel: name by which the function will find a channel, or it will create it (successful test, given a fixed roomId: main) later this will be some kind of dynamic value, such as the room ID.

- Listening for users to join the channel: member join event listener (asynchronous), getting the member's UID.
- Respond to a user joining a channel: handle message from (asynchronously), get the member's ID, get the message point text value.
- **Test result: fail.** The console responds in real time to the actual creation of the offer once a user joins. but encounters a bug.
- **bug(debug 10.07)** For some reason, if I refresh the page too quickly, the local stream doesn't get created immediately, so basically we're calling localstream. gettracks, and every so often, when the page refreshes, that's basically a null value, and then we can get the track, and then it throws some sort of error.

2.4.2 10.07

- **debug** If it's not a local stream, so we're using the not operator, let's quickly create a local stream and attach it to dom if we don't have it when we create the quote, so it's a bit of an extra check if we happen to not have it. I'm sure there are different ways to do this, but this should be a quick fix.

2.4.3 10.09

- createanswer (asynchronous) function: pass in the member ID, on the peer two side we still need to create the pure connection, we still need to set up the remote stream, which will be user two of peer two, so technically on the receiving end, the user sending the offer will be user two, call await create peer connection and pass in the member ID. do pureconnection.set remote description, which needs to send back their SDP answer, sends a message to the peer, the type will be answer, and then we will take the actual answer.
- **Test result: fail.** Got the quote, we created the answer, we got the answer, we added the answer and then when we got a candidate, we went ahead and set that candidate up, reporting an error.
- **bug(debug 10.09)** The likely cause is presumed to be that in the Create Peer Connection function, a track was forgotten to be added to the peer connection track, so this track was brought into the remote stream.
- **Test result: success.**
- **bug(debug 10.13)** On the 2 side of the peer, 1 will be frozen when it leaves.

2.4.4 10.13 (centralised development)

- **debug** css hide away from the user. Grab the second video container or video tag and hide it. Once the user actually joins the stream, display it as a block element. Both local and remote users can do this, so both peers are triggered. handle user left function, takes the id of the member, passes this id, grabs the display attribute of the user's corresponding block set to invisible.
- **Test result: success.**
- **bug(debug 10.16)** We don't actually leave the channel with the user when they try to leave here.

2.4.5 10.16

- **debug** Leaving the channel: leave channel (asynchronous) function, create or call await channel. leave, leave channel, user logout, client.logout.
- **Test result: success.** Perfect exit.
- **bug(debug 10.16)** The user triggers off and we should still be in our conversation, but because another user has just left the channel, that element is now deleted, so now if I do go and delete this or comment on this or delete that comment, now the user will be deleted as soon as we tell them.
- **debug** Create dynamic rooms: by attaching the name of the room to the URL, you can actually create multiple rooms and let people join them, so what we're going to do is we're going to create another page, which will be a lobby.
- lobby initialize lobby front end: create form for entering room number and user ID. Add listener to form. add event listener, grab invite code, pass in url parameter to enter room (previously written dead as main room).
- **Test result: success.** Peer-to-peer users can join the room and launch.

2.5 sprint4(10.17-10.30)

2.5.1 10.20 (centralised development)

- Control container functionality implementation (camera, microphone): front-end html page initialisation of control wrapping containers, button design and associated click animation js and css files.
- Turn the camera on and off (asynchronously): look for all tracks in the local stream, find the video track, loop to find that track, go to the local stream, find the video track, go to the local stream, find the video track, meet the condition track dot kind value is equal to video, turn off. Close with open.
- Turning the microphone on and off (asynchronous): similar.
- **Test result: success.** The switch can be toggled.

2.5.2 10.23

- Adjust the main view display: for the user itself, keep it as it is, the remote user's style takes up the big screen. css and js styling. Grab the current user, now get user one, copy and paste this here, let create some space to change this user one, this is, just to add a class. To do the classlist dot add, to go ahead and add the class for the small frame, which will change the frame, and then after that, also make sure to delete it when the user leaves, in place of handle user left, let go ahead and update this place, handle user left to go ahead and grab the same element.
- **Test result: success.** Add a new user, when this user joins this video frame is at the top of this section, if I leave now the remote user is in the larger frame and the video frame currently jumps to the top left here.
- Project one completed.

- To summarise with the official version of the project two forward: the application only deals with two users, it's just peer to peer and group video calls are a completely different topic, there are multiple ways to do this in webrtc, not going to build anything but I did want to give you a high level overview of how to deal with it before moving on to the next project in this video. The first method is a mesh network, through which a network of peers is created, these peers maintain peer connections to other peers in a group video call, so assuming there is a call with five peers, peer one must create a peer connection to peer two and then peer three. Peer two connects to peer two and then to peer three, peer four and peer five. All the other points in the network must create the same connection to the other points in that network, it connects them together, this happens to the other points that join thereafter, each new point that joins must create this connection, all the other points must connect to this point and each time a new point joins, a new connection must now be maintained. This does technically meet the requirements, but there is a huge downside to this approach, it simply does not scale, the problem is that each time a new peer joins, a stream of data must be uploaded to maintain that connection, this requires a lot of upload bandwidth, once you get beyond four or five peers you start to see a lot of problems, this approach is not recommended if you want to scale. architecture, but more of a peer-to-peer server architecture, which is the MCU approach and stands for Multi Conference Unit. The way this approach works is that each peer is now connected to a server and unlike a mesh network, the data streams that are uploaded to the server, the server receives all the incoming data streams, processes them they mix them together and then they send them back to the peer, the advantage of this is that it scales much more and you only have to upload the data streams once, the disadvantage is that the server you are sending the data to has to be very powerful and have to handle all this data . So it can become very expensive to actually deal with this, it scales better but the downside is that it can be more expensive. The third and last method, there is the sfu method, which stands for selective forwarding unit, which is very similar to the MCU method, i.e. there are still present in the server architecture and still only one upload of streams to the server. The difference is that no longer are these streams mixed together for processing, the server now acts as a router and simply makes sure that it gets all the peer streams, all the peer streams get the streams, it simply directs the traffic and that's all it does, the advantage of this is that because no longer are all the streams mixed and processed, there is now no need for such a powerful server, it would be a lot though there are a lot of benefits to this method, the thing to note is that there is still only one stream to upload but now you have to download multiple streams, if you start having a group video call, there could be 20 30 40 people and you have to be aware of how fast people are downloading, now most people would probably be fine with this but it just needs to be considered the fact that you do need to download The benefit of the MCU method is that users only have to upload once and download once whereas the Sfu approach is to upload once and then download multiple times and in the second part of this video, creating a group video called an application but instead of using one of the three approaches that I talked about, going the other way and just going to use a third party So whenever you're going to build an application like this, you have to decide do I want to build all of this myself, do I want to build infrastructure to handle the scaling of this application or is there a third party service that can make my life easier.

2.6 sprint5(10.31-11.13)

2.6.1 11.03 (centralised development)

- Project 2 Requirements Analysis: When joining this site, you will see or when joining a room, you will see a bot sending a message to and it will always welcome to the room. I can say "Hi" and then "Hi" and this is all in real time and any other user in this room will see this and I will see my name and now if I really want to share my screen I can go ahead and switch this and select the screen to share here. If another user wants to join the room, all they have to do is copy and paste this link here, and if they don't have a name, it will immediately redirect them to the lobby. If Paul wants to put Dennis' video in focus, all he has to do is click on this so that it expands to focus and can switch it at any time, just keep changing it and it will adjust the actual video frame, basically changing the focus. If you go back to this room, Dennis will see that Paul has joined the room, and if Paul does want to join the room, he won't have to deal with the invitation. Maybe this will be added in the live demo, but now anyone can join the room, and if Paul joins, this is now Paul's stream, and if Paul wants to show his face here, he can switch back and forth here, and if it goes back to Dennis's screen, Dennis can switch back and forth as well, and Dennis can now see Paul. That's the core functionality, there's the participants, there's the full functionality of switching the microphone, switching the screen share, changing the video frame, and when you actually go from the screen share back to the presentation, I make sure your microphone is muted. I make sure that your microphone is muted and your webcam is muted, if you're sharing your screen but maybe you have someone in the room and then you go back and just turn off your screen sharing and your webcam is on again, it can be a bit awkward and you actually have to switch your webcam.
- Initialising the front end.
- Initialize Agora webRTC service (core content): create the mesh network, it has many drawbacks, very limited, also sfu method and mcu method, what to do now is to use the third party service agora, instead of entering all these, the whole network created. Import the webRTC (real time communication) and webRTM (real time messaging) SDKs into the project, initialise the UIDs (random numbers) needed in the channels, token binding, room url logic modification.
- **Test result: success.** The environment configuration is tested successfully and the console shows that the link to the agora service was successful.
- Modify the initialisation of the client: the createclient has the mode, which can be Live or rtc, which is just the optimisation algorithm to be used, and then there is the codec, which is just the method used by the browser to encode it. The call waits and the client is called using this client object. Throw in the uid.
- **Test result: success.** Visible.

2.6.2 11.06

- Audio and video track merge: create a function to output the video stream, get the camera feed or camera and audio and then its displayed on the main screen. It is called when the room is initially joined and later joined via a button here and the actual click of a button to join the actual room. The call waits for the Agora RTC point to create a microphone, creates an actual video player, grabs one of these video containers, stores it and to add this video player to this streaming container.

- **Test result: success.** You can see it for yourself, but others cannot.
- Others handle events for streaming: create asynchronous functions, create an event listener, get the user value from the user, the actual track, then the media type, call user dash published, any time a user publishes, listen, call handle user published. create a player for this user, this player is the same as the personal player audio tracks are merged in the same way.
- **Test result: success.** Visible.
- The actual playback of the merged video and audio: check the media type, take the parameters of the user object to access the video track, the video track, and then just call the play method here. Whenever a user joins the room, call join stream, then you have the local audio track, the last thing you need to do is to actually publish the track, just keep calling await and then do client dot publish, you need to specify the track you want to publish here.
- **Test result: success.** Three users enter the same room at the same time and their respective audio and video services are normal and synchronised.
- **Test result: fail.**
- **bug(debug 11.10)** Add another user, give it a second to load that user, to add a second and then a third, delete that element, let go ahead and save it, add another user here, if I click on x, I can now publish a track and also delete that track, if I'm with another user it will obviously do the same thing The next thing I want to do is focus on actually rendering Right now it's hidden, but at any point, say a user is sharing their screen, or I just want to focus on say user number two who is talking, I want to be able to click on that user, add them to this big frame, and then switch users in that frame and also delete him.

2.6.3 11.10 (centralised development)

- **debug** Extending the video framework. Get all video containers. Set one When clicking on an element, I want to know who the user is inside this display frame. Leave it blank and expand a video frame. When a new video frame is added, add another user that can switch between the two.
- **Test result: success.** There are changes.
- **bug(debug 11.10)** Now this is the remote user, this is the user in this tab, if I click on this user they are displaying their stream, if I close this user they leave the stream. If they are the user that is currently showing their video stream, just want to resize all of these to the original height and width, make sure that this is fixed down, what to do is, if the current user ID is in the display box, to get that ID, to check if it is this current user, to get that full ID, to put it in. But also want to know if this host needs to be hidden, and if so, if they are the person in there, let go ahead and do the display frame point style point display, just to set this to empty, to remove any styles.
- **debug** Redesign of css.

2.6.4 11.13

- Screen sharing: go to the RTC room, local screen track, set a value called shared screen, set the switch screen asynchronous function. Event listener. Call agora rt to create a screen video track, purpose is to toggle again asking what screen you want to share, if agreed, just like the first time you use the camera, now once this is done, let go ahead and delete the current video track and only share the video or camera track. Grab the current uid and throw in the uid, the current video track. Create a video player, set it to a variable Will grab the item, then will do something to add an event listener, will listen for a click event, then will go ahead and add the extended video frame, each new item added make sure it has that, in this case just add it to the new element just created. Grab all or create the track and then call the local track to play, it goes on to create a video tag and its thrown into this frame, do the same thing here, once the user is set and the frame is displayed, just At this point there is only one track and nothing has to be done.
- **Test result: success.**
- But there are problems: Issue 1: The screen track is not published. Issue 2: Turning off sharing automatically turns on the camera.
- Stop sharing: Unpublish and republish video tracks, now set the sharing screen to false if it is streaming.
- Sharing optimised: continue playing the local video track, to take this out of the join stream, take this to the switch to the camera function, the audio track that has been published, switch my camera again. Share this screen, deactivate the microphone and camera.
- **Test result: success.** The second page also allows you to join the room and see everyone's video stream.

2.7 sprint6(11.14-11.27)

2.7.1 11.17 (centralised development)

- Users are able to create new rooms: ensure that users can actually enter the created room and specify their name. Add event listeners to react to lobby lobby input (asynchronously). The form gets room.value, gets the invite code, but if the invite code doesn't exist, use the not operator, go ahead and say, hey, if there is no invite code, let go ahead and create a new one, for this, just go ahead and do the string, generating a random number between 1 and 10 000. window.location, just to redirect the user, once the form is submitted, to redirect the user to the room.
- **Test result: success.**
- Real time feature added - see the participants in the room in real time on the left side: initialize Agora rtm to be able to use this SDK to create this feature. modify the configuration of a client object (rtm client), create the room channel, when joining a video call, in the initialization of joining the room, also use the rtm sdk to join the channel, let move this here, basically join two things at once, get the rtm client, at this point you can call wait agora rtm, previously with rtc, now here with rtm. continue call await, here perform rtm client point create channel, if this channel exists, join it, if not, create it, take the same room id, room id created here for rtc client, join the same thing, create or join Once this channel is created, you need to join this channel.

- **Test result: success.**Initialisation successful, console link rtm successful.
- Room participant import: handle member joined (asynchronous) function, executes member ID.
- **Test result: success.**Type tim and the console hits the list of members containing tim.
- Room participant exit: when a user leaves, delete the stream within their channel, front-end remove wrapper, need to be able to query to delete it.
- **Test result: fail.**Actually leaving the channel, if you just close the browser, the member remains there.
- **bug(debug 11.17)**agora by default if it senses an activity it will actually But expect this function to happen immediately when the user closes their room, leaves the channel, shuts down their computer or whatever happens, expect this function to be called, it is now possible to call this function when some sort of back button or exit button is clicked but this is not always practical as some people just leave the room by closing their laptop or shutting down their computer, not everyone will click a button .
- **debug**Calling it before unloading or uninstalling the event listener first creates a function which is called leave channel and continues to actually call this when the user leaves a channel, leaving the channel continues with the execution of await channel . leave and then logging in with the rtm client just executes await rtm client.
- **Test result: success.**When a member leaves, see that it is removed from the domain.

2.7.2 11.20

- **bug(debug 11.20)**A new user joins and wants to see myself as well as the users who were there before me, but now only new members are visible.
- **debug**Underneath removing members from dom, let's call it get members,, call all members in the channel on the first load, to get members, create a variable members and the channel has a function, I actually want to make sure I add a weight here, there's a method called get members, it goes into this channel and gives every member currently in this channel, it is just going to return an array of their IDs. Once you get this, create a for loop that loops through the members of each channel, then they are added to dom, let continue doing let i equal 0, then do members . length, if it is greater than i, continue adding i, then continue adding this function here, do add member to dom, at this point, because of the iteration of this loop, actually get the ids of the members, do members and Then get the ID of the current member, to continue calling add member to dom, then render them here, in order to get the members on the first load, let continue grab this, bring it into the underscore rtc room, I put it underneath the channel function, let continue to call this, continue to call get members and hopefully have time to get everything loaded and load all the channel members here.
- **Test result: success.**Display is normal.

2.7.3 11.24 (centralised development)

- Adding a chat message: send message (asynchronous) function. Send a message to the whole channel, anyone who belongs to this channel can have an event listener that can

listen to this, it is a channel message. Add an event listener, get the element message form by ID without adding an event listener, the event to listen for is submit or when submitted, then just to call send the message .

- **Test result: success.**The form is entered and displayed in the console.
- Receiver message response: add channel, listen for channel messages. Calling this handle channel message always makes them an asynchronous function, get the message data here, also want to know who the message came from, get the member ID, both values are immediately accessible, then continue with this function. Create a variable called data, parse it to a text value and get everything inside this object.
- **Test result: success.**A sends the message and the console at B can see the message hit.
- Display text: front-end content, get message, handle channel message update on screen. Really get the message and attach it to the dom.
- **Test result: success.**
- Welcome text is displayed in the chat box: the previous join member handler section is called and the text (based on ID, handled in the same way as above) is entered into the chat channel.
- No more messages are sent to the user after leaving the room: grab it from this member wrapper and remove this element from the dom. (The console no longer receives messages after the user exits the room and enters the lobby.)

2.7.4 11.27

- Split measure between entering the stream and entering the room: the user enters the room and can choose whether to enter the stream or not. Add button, getelementbyid, create an event listener for join btn, for join button, do add event listener, let continue to listen for click events, do click, let continue to call join stream, now only after this button is clicked, to go back to the join stream function, let change something here, before really before actually getting the audio and video tracks, let change some values and grab the join stream button.
- **Test result: success.**
- Open Stream: To add an event listener to this button that allows the user to leave, unpublish their video, but still stay in the chat room. Asynchronous function, to close the room rtm, add an event listener for leave stream. Create a loop that is going to loop through each of the local tracks, to stop and then close each track.
- **Test result: success.**The console agorartm sent the message successfully and the user has left stream.

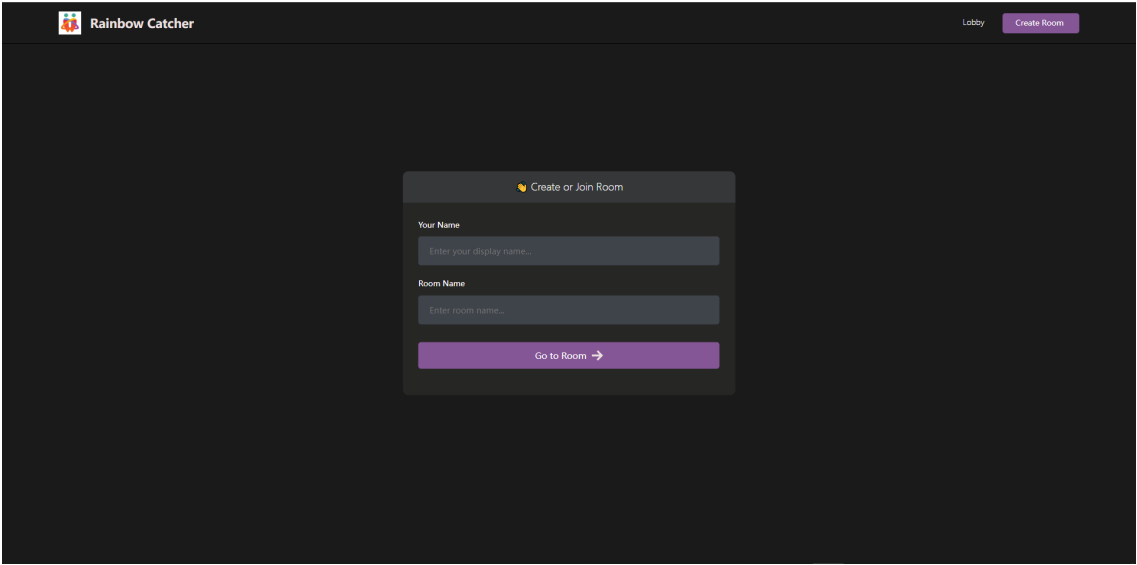


Figure 1: Output-lobby

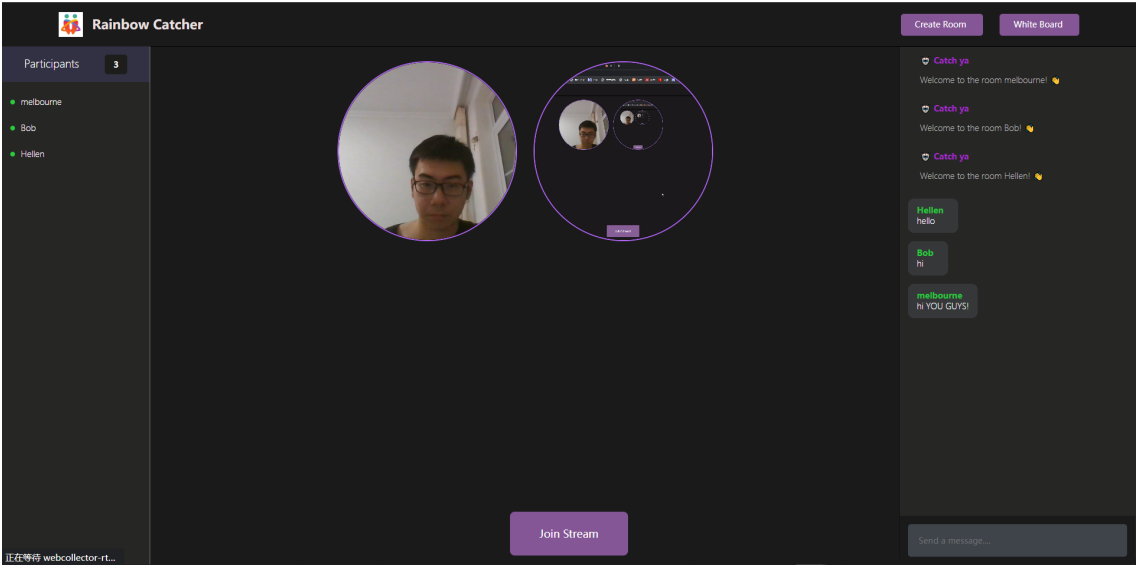


Figure 2: Output-join stream

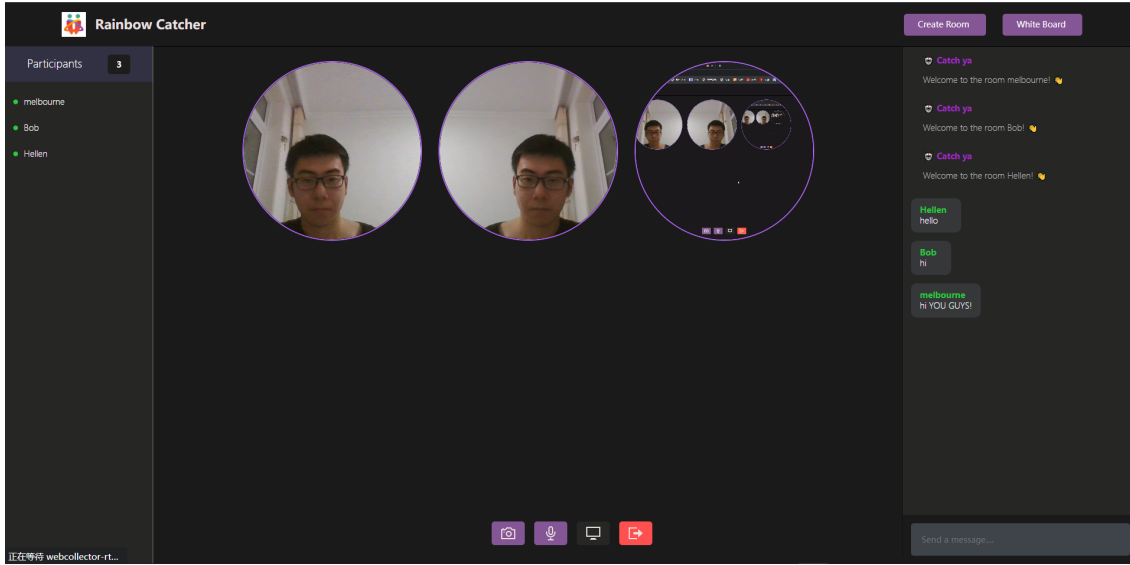


Figure 3: Output-unfocus

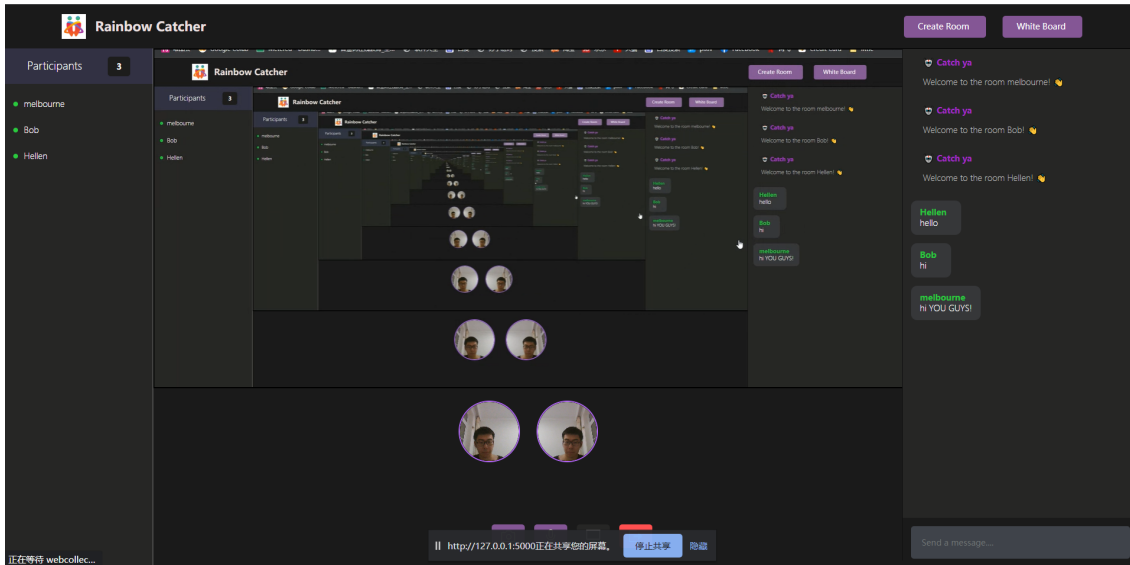


Figure 4: Output-focus

3 Personal application of course knowledge as the project progresses

3.1 General understanding

Software engineering methodology consists of five basic areas: 1. structured theory and methods; 2. modular techniques and data abstraction; 3. software testing and proof of program correctness; 4. software analysis and design methods, tools and environments; and 5. software engineering management and quality evaluation. I believe that software engineering methodology provides us with technical guidance on "how to do it" when developing software. For example, in the process of learning how to develop, we follow the methods it gives us, which can be understood as a learning model, or a learning framework. And it is these methods given to us that have helped many of us newcomers when we were first learning. It can be said that

the software engineering methodology is the experience, the learning method, that has been summarised by many predecessors. In this methodology, we are told what to do and how to do it, so that we can avoid a lot of detours and enable us to improve the speed as well as the quality of our development in software development. It has a unique role for us to learn software development, especially for beginners. Based on the software engineering methodology, we have a general learning direction and development framework, so that beginners do not feel at a loss and have no way to start. It is both a framework for getting started quickly and a method for later learners to learn from the experience of those who have gone before, a tool on which later learners can draw lessons, expand their thinking and add their own ideas to make it better.

3.2 Project Management

- People management: This course collaboration was the first time that I led a group in a project as project leader and main manager. First of all during the selection of the group members I combined the relevant knowledge, for a change from the high priority I used to give to hands-on programming skills. This time I profiled the group members and classmates in several dimensions, including application domain experience, programming ability and communication skills, and assigned responsibilities in relation to their respective strengths.
- I took the lead in proposing a 2+2+1 rotation model that suited the characteristics of our group, allowing students who were good at writing and coding to do what they were good at. At the same time, in order to ensure coordination within the project, we increased the number of rotating roles and ensured that everyone had access to the development content and had some paperwork skills at the same time, greatly reducing the risk of staff changes and increasing efficiency.
- For the software development aspect, the overall development process was established and then responsibilities were assigned.
- Risk assessment: This was the first time I had carried out a risk assessment during this project and certain risks did arise as the project progressed, but it was good to have estimated them in advance and allowed sufficient buffer time.
- For the estimation of the workload, the learning was calculated using COCOMO.
- Project charter: led the writing of the team charter.
- Overseeing the conduct of the organizational meetings on Sundays, taking the relevant minutes, and the conduct of the co-programming on Thursdays.
- Hosted the project kick-off meeting and several mid-stage events.
- Dealing with the resolution of various conflicts between members or during the progress of the project, especially during the docking process.
- Led the establishment of a team collaboration process with microsoft Team as the core. The team functioned to its full potential, regularly updating documents, posting tasks, meetings, recording communications etc.

3.3 Software Process

- A detailed understanding of the software development model and the practical implementation of a requirements-based, agile development framework with scrum as the main management model in the entire process.
- The prototype model was fully utilised at the beginning of the project and a paper prototype image of the main body of the video conference and the whiteboard section was drawn independently.

3.4 Agile Development

- The core body of our work is agile development, but at my suggestion, learning from past experience and in order to prevent problems such as the large span of development sections between people in last term's team project, lack of reference, difficulty in splicing and serious communication dilemmas, a small documentation + logging + agile development model was adopted for the development work. The importance of documentation in the alignment process is emphasised.
- During the initial meeting of each sprint as the project progresses, I will act as an ad hoc 'customer', suggesting requirements and changes to the project delivered within the current schedule, reflecting the customer value focus of agile development.
- In the course of the project, the team was fully driven to incorporate the scrum management model to assist the agile development process. For example, meetings will require everyone to produce product backlogs, relevant content will be displayed using the Microsoft team's Kanban board content, to urge everyone to complete the project, continuous and uninterrupted iterative delivery, and will regularly update the content and progress on the Kanban board.
- The use of the planning poker was only introduced in the first two sprint sessions in order to prioritise and prioritise the needs of the individual courses, and was removed at a later stage after the sessions had been streamlined due to overload and pressure from the individual courses.
- Test Driven: This project was the first time I wrote functional and unit tests and was involved in collaborative and integration testing. The tests, and the fact that they are reflected in the logs, are very rewarding.

3.5 Requirements Engineering

- In previous courses and designs, the specific requirements and details of the teacher or assignment project, or even the marking criteria, were often used as the development criteria. Therefore the extraction, refinement, categorisation and analysis of user requirements was almost never considered. In this course design, the importance of customer requirements engineering was fully realised and the activities involved were actively planned, discussed efficiently with the group and requirements details were established before the design started. Have also participated in the writing of the requirements myself.
- The requirements extraction process was actively used in the form of brainstorming and the prototype was written on paper in several meetings, and the requirements were constantly updated after the discussions.
- Use case: independently completed the relevant elements of the use case diagram.

3.6 User Story

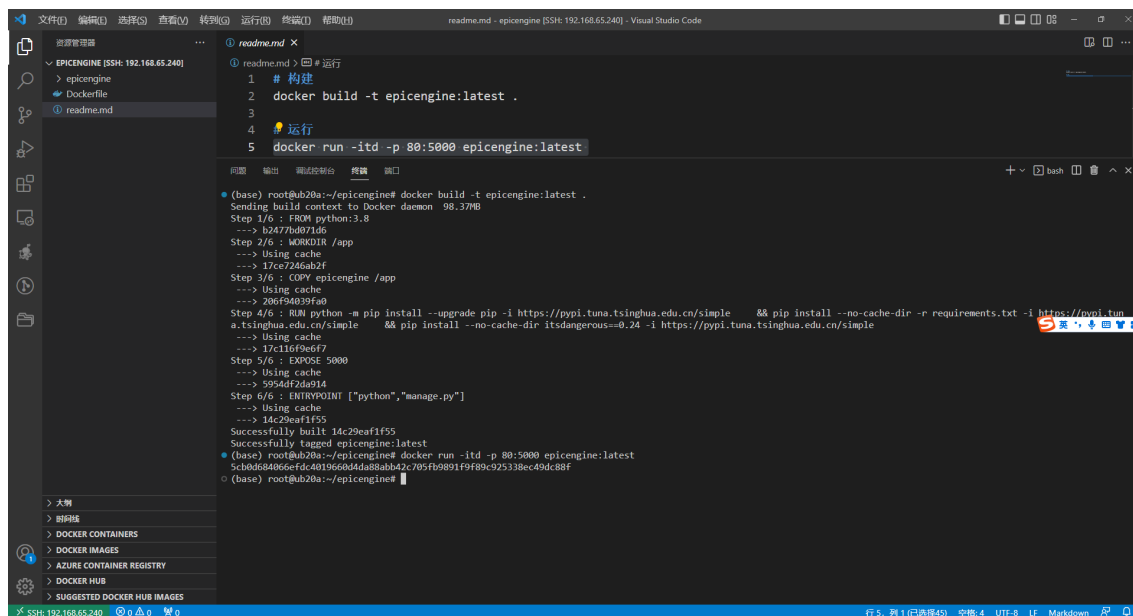
- The module on user stories was actually introduced last semester. This project still places a major emphasis on it in the group meetings of each sprint.
- Actively participated in the presentation of the userstory and the writing of the log.

3.7 System Modeling

- Learned a lot about UML diagrams. Participated in the drawing of the Activity diagram and the sequence diagram.
- Has a deep knowledge of OOD and follows a strict sequence of requirements extraction, requirements analysis, design and practice. Participated in drawing various diagrams such as relationship diagrams, use case diagrams, use case detail description architecture diagrams and sequence diagrams. And in the practice phase.

3.8 Software Testing

- Unit tests, see log for details.



```
readme.md - epicengine [SSH: 192.168.65.240] - Visual Studio Code

1 # 构建
2 docker build -t epicengine:latest .
3
4 运行
5 docker run -itd -p 80:5000 epicengine:latest

(base) root@ub20a:~/epicengine# docker build -t epicengine:latest .
Sending build context to Docker daemon  98.37MB
Step 1/6 : FROM python:3.8
--> b2d77b0971d0
Step 2/6 : WORKDIR /app
--> Using cache
--> 17ce72d6ab2f
Step 3/6 : COPY epicengine /app
--> Using cache
--> 206f9d039fa0
Step 4/6 : RUN python -m pip install --upgrade pip -i https://pypi.tuna.tsinghua.edu.cn/simple && pip install --no-cache-dir -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple
--> Using cache
--> 17c116f9e6f7
Step 5/6 : EXPOSE 5000
--> Using cache
--> 5954df2da914
Step 6/6 : ENTRYPOINT ["python", "manage.py"]
--> Using cache
--> 14c29eaf1f55
Successfully built 14c29eaf1f55
Successfully tagged epicengine:latest
(base) root@ub20a:~/epicengine# docker run -itd -p 80:5000 epicengine:latest
5eb0e08eef6c40196ebdd48a8bbd2705f09b91f9f89c325338ec43dc8df
(base) root@ub20a:~/epicengine#
```

Figure 5: Docker Construct and Start