Introduction

1.1 The overall project vision, goal, purpose or objective

Within recent years, remote working has been increasingly embraced by the global population (Welson-Rossman, 2020). However, the recent pandemic and the consequential lockdowns have caused the notion of remote working to become ubiquitous in far more people's lives (Welson-Rossman, 2020). With no sign of this trend reversing in the long run, remote working has been deemed 'here to stay' by many, with a Gartner survey finding that 74% of CFOs have already reported they intend to shift some of their employees to remote work on a permanent basis ("Gartner CFO Survey", 2020).

Paralleling this increasing prevalence of working remotely, we are also developing an ever increasing reliance on video-based conferencing tools such as Zoom, Google Meets and Microsoft Teams [(Wadhwani & Gankar, 2020)]. In fact, the video conferencing market size is predicted to exceed USD 50 billion by 2026 (Wadhwani & Gankar, 2020). As with the increase we saw in remote working as the COVID-19 pandemic commenced, the demand for videoconferencing also grew exponentially as the pandemic set in. The number video conference mobile app downloads reached 62 million between March 14 and 21, 2020, 90% more than the pre-COVID-19 weekly download average (Trueman, 2020). Ultimately, these video conference technologies are helping us to achieve some sort of workplace normality whilst not physical in the office. With the help of these video conference technologies, the modern day workplace has evolved and close physical proximity is often no longer a necessity amongst coworkers (Trueman, 2020).

Despite the promises of encouraging collaboration and productivity among geographically dispersed coworkers, the current video-based conferencing tools still don't offer the same experience as meeting with a colleague in person. Within a conference room setting, we are able to break into groups whilst still being able to glance around and maintain visualisation of what other conversations are going on around the room. That is, we can see who is talking to who, and can easily flit around to these different groups, popping in and sampling the conversations to see what is going on.

Video conferencing tools do not yet offer us these same liberties. Rather, once in a meeting, we can only see who is in our own meeting. We have no visualisation over other conversations going on in separate meetings, who is a part of these meetings, and have no way to easily move between said meetings. We lose the effortlessness of moving around between conversations that we would normally get in person in social settings, conference rooms, and classrooms. These siloed meeting rooms represent an extreme limitation of the current video-conferencing platforms.

1.2 What the project will achieve

This project sets out to combat the pain-points currently experienced in the siloed meetings of current video conferencing platforms. It aims to exploit a web-platform API to create a web-based, video conferencing platform with the look and feel of a real conference floor. This tool should allow users to effortlessly move between conversations occurring on the platform and be in a conversation whilst maintaining visualisation over what other conversations are occurring around them.

More specifically, the web conferencing platform will be able to support, at a minimum, 10 simultaneous video calls, each able to support at least 5 participants within them. When within one of these video calls, a participant maintains visualisation over who is in each other meeting that is happening at the same time. Each other meeting occurring is listed, along with a visually appealing, low resolution snapshot representing members of that meeting, on the meeting screen of a participant.

Meeting certain benchmarks in terms of performance and usability is also part of the overall goal of the project. For instance, transitioning between meetings effortlessly in less than 3 seconds, with at most 2 clicks is an example of a usability metric which is being considered.

1.3 The key stakeholders, what do they do, and how they interact

We have categorised our major stakeholders into two categories, external or internal, and outlined their differing levels of power over, and interest in, our project. The primary external stakeholder for this project is Ben Sand, the project manager of our client. Due to the high level of power and interest Ben has in our project, and based on our stakeholder analysis using a power-interest grid we realised that we need to manage his expectations closely. He has the biggest impact on the success of our project and is the major decision maker (Bdaiwi, 2017). To keep ongoing communications with Ben, we meet with him twice a week, on Mondays and Fridays at 4:10pm and regularly get updates on his most recent requirements for the project. For each meeting, we have an agreed deliverable due and must create a 1 minute progress video to present to him in order to maximise the efficiency of each meeting.

Other external stakeholders identified who have low power and high interest in our project include the Jitsi developers team, our tutor and our end-users as our external stakeholders. Our team team members are also considered high interest, low power internal stakeholders. All these stakeholders need to be considered and informed adequately as they can provide important information and support to the project. An overview of all major stakeholders identified, along with their expectations for the project, has been provided below in form of a stakeholder register.

Stakeholder name	Title	Role in Project	Type of Stakeholder	Power (H/L)	Interest(H/L)	Type of Communication	Expectations
Ben Sand	Honorary Adjunct Professor	Client PM	External	H	H	Video call - twice a week	On-time, good quality deliverable based on specified requirements of the week
Jitsi Team	Jitsi Developers Team	Supplier/Supporter	External	H	L	Jitsi Community Forum - depending on when we need their assistance throughout project implementation	Clear identification of issues and questions
Adhish Panta	Tutor	Supporter	External	L	H	Tutorial once a week through Zoom call	Clear, concise report of our weekly progress

Stakeholder name	Title	Role in Project	Type of Stakeholder	Power (H/L)	Interest(H/L)	Type of Communication	Expectations
User	Anyone with interest in video calling in a conference room environment	End-User	External	L	H	Usability test / upon request	Being able to video call with others with user-friendly UI, having access to gallery view, switching between sessions and having access to snapshots of what is going on in other meeting rooms
Jamie Ramjan	Student	Client Point of Contact/Programmer	Internal	L	H	Weekly group meetings via video call	**Client point of contact**: Predefine the problems and issues that the team is facing, document and then send them to the client. Being responsible for communicating with the client outside of the regular meeting time to ensure that both the team and client can be timely updated to the progress and changes; **Programmer**: Estimates, generate, and implement user stories, split each story into achievable, measurable, and traceable Engineering Tasks. Being able to contribute to the development of the project technically.
Holly Craig	Student	Manager/Programmer	Internal	L	H	Weekly group meetings via video call	**Manager**: Schedule and prepare agenda for the meetings, guide the conversation and make sure that the current progress, issues, and challenges are all covered during the session, and record the decisions, actions, and the future plan in Meeting Minutes. Also in charge of assigning team members proper works corresponding to their project role and skills; **Programmer**: Estimates, generate, and implement user stories, split each story into achievable, measurable, and traceable Engineering Tasks. Being able to contribute to the development of the project technically.
Jiahe Zhou	Student	Head Programmer	Internal	L	H	Weekly group meetings via video call	**Head Programmer**: Overseeing the project and work, being clear to the constraints and requirements, and ensure the project deliverables meet these requirements in a technical sense. Estimates and generates user stories that can be divided into achievable, measurable, and traceable Engineering Tasks. Being able to translate tasks into coding goals based on required skills and programming style, etc. Have the ability to conduct code reviews, provide technical advice and support during the developing process. Timely communicate with tracker and tester to update the technical issues that are tackled.
Linze Li	Student	Tester/Programmer	Internal	L	H	Weekly group meetings via video call	**Tester**: Design overall testing plan, implement, and conduct functional tests. Ensure the testing result fits into the requirements of related user stories. Document the testing results in both text explanation and visual evidence, update them to both the head programmer and the person in charge of that part when the test failed; **Programmer**: Estimates, generate, and implement user stories, split each story into achievable, measurable, and traceable Engineering Tasks. Being able to contribute to the development of the project technically.

Stakeholder name	Title	Role in Project	Type of Stakeholder	Power (H/L)	Interest(H/L)	Type of Communication	Expectations
Kimia Asarroodi	Student	Tracker/Programmer	Internal	L	H	Weekly group meetings via video call	**Tracker**: List the requirements and expected deliverables, document the progress, problems, issues, and plan in the status report each week. Check with the programmers and tester regularly to see if everything is under control or require some help. Being able to quickly respond to the changes in the project to ensure that the developing process is on track and does not have dangerous slippage in the schedule; **Programmer**: Estimates, generate, and implement user stories, split each story into achievable, measurable, and traceable Engineering Tasks. Being able to contribute to the development of the project technically.

1.4 Identification of resources and risks involved in the project

We have classified our resources into three separate categories including human, time and material/equipment resources.

The human resources available for this project include Ben Sand as our client's project manager who provides us the requirements of the project and assistance upon request on a regular basis, Adhish Panta as our tutor who assists us in terms of the management and administration of tasks of the project, and our team members who cover different XP roles including customer point of contact, manager, tracker, programmer and tester. Team members with a background in different areas of IT including web development, server side development and UI/UX come together to satisfy the combination of skills required for this project. The risk associated with this category of resources is that, although the aforementioned people are each expert in their field, during the process of implementation of the project, issues might arise which will require knowledge in different or more specific areas. An instance of these areas can be the questions that arise which need to be directly asked of developers from the Jitsi/Zoom team who have a more in depth and detailed understanding of their particular system.

Like any other project, time is a crucial resource and a key driver of this project. Identification of hours required for the completion of each task, creating multiple deadlines and milestones based on major due dates for deliverables, and monitoring the time spent on different activities have been crucial for maintaining proper management of this vital and limited resource. Poor management of time would result in missing targets, wasting human and material resources and the reduction in the quality of work being delivered to the client. Therefore, these risks need to be considered throughout the whole process of this project (Atkinson, 1999).

In this project we have been using tools including Bitbucket and SourceTree as our source code management system, Jira as our issue tracker, USYD library website as our academic resource provider, Visual Studio Code as our code editor, Jest as our JS unit testing framework, Termius and PuTTy terminals, AWS route 53 as our domain registry, vultr as our server service provider and Jitsi open-source projects as our base code. Having limited knowledge in terms of the actual functionality and having access to a limited number of supporting documentation for the platforms mentioned previously, is considered a risk associated with this category of resources.

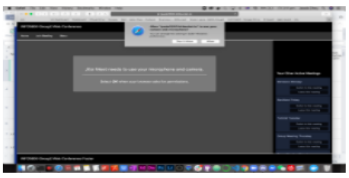
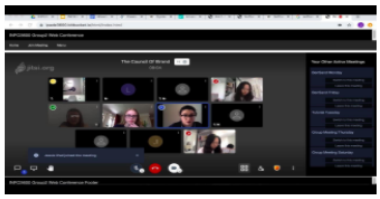
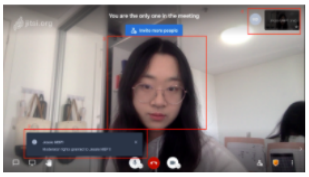
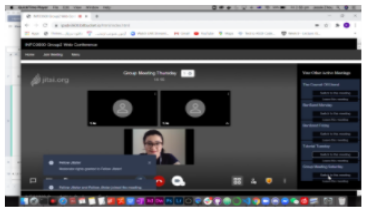
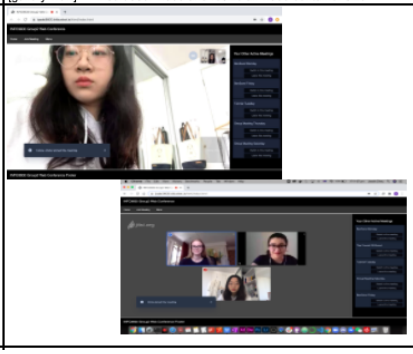
Overview of system from user view (user stories)

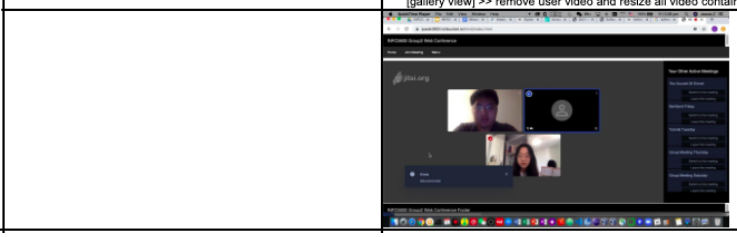
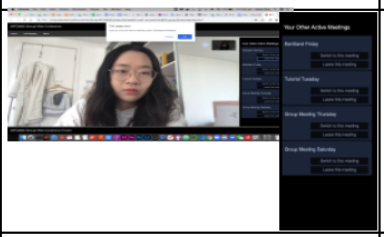
2.1 User

It is critical for designers and developer to have a clear view of what operations should be able to be performed by the users, what activities they need to do to complete each operation, and what might happen to both of the manipulator and the people participating the meeting.

We will use Green Yellow Red (GYR) to represent the current construction status of corresponding operation of user. "G" means that we have included that functionality in the latest version of development; "Y" means that we have partially achieved the goal of that operation, but are still working on enabling all of the functionalities; and "R" means that this part is in plans for development, but has not been started yet.

Please note, we tried to fit the text content into wiki, but its format is not friendly to present a table with pictures so we were forced to use screen shots instead. If you would like to see the original version of this table, feel free to look at the original excel sheet here.

#	Action User	Desctiption			Screen Event--What will happen? (Screenshot)		User story	Status: - G: Complete - Y: Constructing - R: Planned
		Operation	Setting & Timing	Where & How	User (me)	Participant (if any)		
1	User (me)	Join a meeting	First meeting	Use join form on homepage	1. Redirected to meeting.html 2. Pop-up window: microphone and camera access 3. Pop-up window: user name 	1. Left-bottom corner popup: [user provided name/ninja] join the meeting 2. Video containers: [speaker view] >> load user video in main container,having self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip as well [gallery view] >> insert user video and resize all video container size 	1,6,7	Y
2	User (me)	Join a meeting	When the user has no, or less than 10, concurrent meetings, and the meeting to be joined has no other participants	Use join form on the homepage/meeting page OR join via invite link	1. Pop-up window: user name 2. Left-bottom corner popup: Moderator granted to [user provided name/ninja] 3. Large video container: load user video 4. Filmstrip: small canvas on right-top corner with displaying name: "[user provided name/ninja]" 	N/A	1	Y
3	User (me)	Join a meeting	When the user has no, or less than 10, concurrent meetings, and the meeting to be joined has no other participants	Use join form on homepage/meeting page OR join via invite link	1. Pop-up window: user name 2. Left-bottom corner popup: [other participants' provided name/ninja] join the meeting 2. Left-bottom corner popup: Moderator granted to [participant provided name/ninja] 3. All participants' video load in gallery view 	1. Left-bottom corner popup: [user provided name/ninja] join the meeting 2. Video containers: [speaker view] >> load user video in main container,having self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip as well [gallery view] >> insert user video and resize all video container size 	1	Y
4	User (me)	Joining a meeting	When user already has 10 concurrent meetings	Use join form on homepage/meeting page OR join via invite link	Pop-up window: {"oooops! you have reached a maxium of 10 concurrent meeting! you can: A. discard this joining B. leave one of the following meeting [concurrent list]"}	N/A	1	R
4.1.1	User (me)	Joining a meeting	When user already has 10 concurrent meetings, and selects "discard this joining"	Use join form on meeting page OR join via invite link	Stay in meeting.html and load previous main session	N/A	1	R
4.1.2	User (me)	Join a meeting	When user already has 10 concurrent meetings, and select s"discard this joining"	Use join form on homepage OR join via invite link	1. Redirect to meeting.html 2. Load previous main session 3. Large video container: load user video 4. Filmstrip-self canvas: small canvas on right-top corner with displaying name: "[user provided name/ninja]" 5. Filmstrip-if have other participants: other users' video/name canvas	N/A	1	R
4.2.1	User (me)	Joining meeting A	When user already has 10 concurrent meetings and user selects "leave meeting B"	Use join form on homepage/meeting page OR join via invite link	1. Redirect to meeting.html (if join from homepage) 2. Load joining meeting in main session container 3. Pop-up window: user name 4. Meeting B disappear in user concurrent meeting list sidebar 5. Previous main session appear in user concurrent meeting list sidebar 6. Left-bottom corner popup: [user provided name/ninja] join the meeting 7. Large video container: load user video 8. Filmstrip-self canvas: small canvas on right-top corner with displaying name: "[user provided name/ninja]" 9. Filmstrip-if have other participants: other users' video/name canvas	Case a. Participant in abandoned meeting B and previous main session: a.1. Left-bottom corner popup: [user provided name/ninja] disconneted a.2. Video containers: [speaker view] >> reload onstage participants' video in main container and self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip [gallery view] >> remove user video and resize all video container size Case b. Participant in new joined meeting A: same as what happen in: b.1. Left-bottom corner popup: [user provided name/ninja] join the meeting b.2. Video containers: [speaker view] >> load user video in main container, having self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip as well [gallery view] >> insert user video and resize all video container size	1	R
4.2.2	User (me)	Joining meeting A	When user already has 10 concurrent meetings, and user selects "leave meeting B"; meeting A reached a pre-defined number of MAX_PARTICIPANTS_FOR_HD_VIDEO	Use join form on homepage/meeting page OR join via invite link	1. Redirect to meeting.html (if join from homepage) 2. Pop-up window: user name 3. Load joining meeting A in main session container with lower video quality 4. Meeting B disappear in user concurrent meeting list sidebar 5. Previous main session appear in user concurrent meeting list sidebar	Case a. Participant in abandoned meeting B and previous main session: a.1. left-bottom corner popup: [user provided name/ninja] disconneted a.2. video containers: [speaker view] >> reload onstage participants' video in main container and self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip [gallery view] >> remove user video and resize all video container size Case b. Participant in new joined meeting A: b.1. auto reduce the video quality without alert b.2. left-bottom corner popup: [user provided name/ninja] join the meeting b.3. video containers: [speaker view] >> load user video in main container, having self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip as well [gallery view] >> insert user video and resize all video container size	1	R
5	User (me)	Joining meeting A	When user has already joined that meeting	Use join form on homepage/meeting page OR join via invite link	1. Reload the meeting.html with meeting A 2. Concurrent list sidebar renewal	Case a. Participant in abandoned meeting = previous main session: a.1. Left-bottom corner popup: [user provided name/ninja] disconneted a.2. Video containers: [speaker view] >> reload onstage participants' video in main container and self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip [gallery view] >> insert user video and resize all video container size Case b. Participant in joined meeting A: b.1. Left-bottom corner popup: [user provided name/ninja] join the meeting b.2. Video containers: [speaker view] >> load user video in main container, having self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip as well [gallery view] >> insert user video and resize all video container size	1	Y

6	User (me)	Leave a meeting	When user has no other meeting	Click red hang-up button on meetiing page	Redirect to homepage 	Participant in abandoned main session: 1. Left-bottom corner popup: [user provided name/ninja] left the meeting 2. Video containers: [speaker view] >> reload onstage participants' video in main container and self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip [gallery view] >> remove user video and resize all video container size	1	Y
7	User (me)	Leave a meeting	When user have other concurrent meeting	Click red hang-up button on meetiing page	1. Reload the meeting.html with the concurrent session on the top position of the sidebar list 2. Concurrent list sidebar renewal	a. Participant in abandoned meeting = previous main session: a.1. Left-bottom corner popup: [user provided name/ninja] left the meeting a.2. Video containers: [speaker view] >> reload onstage participants' video in main container and self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip [gallery view] >> remove user video and resize all video container size b. Participant in reloaded meeting A: same as what happen in: b.1. Left-bottom corner popup: [user provided name/ninja] join the meeting b.2. Video containers: [speaker view] >> load user video in main container, having self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip as well [gallery view] >> insert user video and resize all video container size	1	Y
8	User (me)	Leave concurrent meeting B	When user have other concurrent meeting	Click "leave this meeting" button inside concurrent list sidebar on meeting page	Concurrent list sidebar renewal 	Participant in abandoned meeting B: 1. Left-bottom corner popup: [user provided name/ninja] left the meeting 2. Video containers: [speaker view] >> reload onstage participants' video in main container and self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip [gallery view] >> remove user video and resize all video container size	1	Y
9	User (me)	Change display name	During meeting	Click on the self canvas on filmstrip or from bottom functional navigation bar >> me	Instantly change the name that shown on the self canvas on filmstrip	If user disabled video, then participant should visualize the change of username that shown on the user canvas container; otherwise nothing happens instantly	2	Y
10	User (me)	Change video quality	During meeting	Click on the bottom functional navigation bar >> video quality	instantly change the video quality of the meeting and show the current quality inside a samll circle on the right-top corner	N/A	1	G
11	User (me)	Switch from meeting B to meeting A	When user have other concurrent meeting	Click "switch to this meeting" button inside concurrent list sidebar on meeting page	1. Reload the meeting.html with Meeting A 2. Concurrent list sidebar renewal 3. Username that display on screen is the one that unique to Meeting B or a ninja	Case a. Participant in abandoned meeting B = previous main session: a.1. Left-bottom corner popup: [user provided name/ninja] disconneted a.2. Video containers: [speaker view] >> reload onstage participants' video in main container and self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip [gallery view] >> remove user video and resize all video container size Case b. Participant in switched to meeting A: b.1. Left-bottom corner popup: [user provided name/ninja] join the meeting b.2. Video containers: [speaker view] >> load user video in main container, having self canvas on right-top corner with displaying name: "[participant provided name/ninja]", other participants' canvas in left filmstrip as well [gallery view] >> insert user video and resize all video container size	1,3,4,8	G
12	User (me)	Require a thumbnail of meeting B	When user is currently in Meeting A and has Meeting B in concurrent list sidebar	Mouse click the name of Meeting B in concurrent list sidebar	1. A section of thumbnail of Meeting B show up left next to the name of Meeting B in sidebar (won't affect the visilibity of switching button) 2. User can click on the name of Meeting B again to close the thumbnail 3. User can click "switch to this meeting" button while having the thumbnail working, and will lead to Action No. 11	N/A	1	Y
13	Participant	Join a meeting	When user is currently in that meeting	N/A (from user's aspect)	1. Left-bottom corner popup: participant provided name/ninja] join the meeting 2. Large video container: [speaker view] >> load user video OR [gallery view] >> insert user video and resize all video container size, small canvas on right-top corner with displaying name: "[user provided name/ninja]", other participants' canvas in filmstrip	1. Pop-up window: participant name 2. Left-bottom corner popup: [participant provided name/ninja] join the meeting 3. Large video container: load participant video 4. Filmstrip: small canvas on right-top corner with displaying name: "[participant provided name/ninja]", user(me) and other participants' video/name canvas	1	G
14	Participant	Leave a meeting	When user is currently in that meeting	N/A (from user's aspect)	1. Left-bottom corner popup: [user provided name/ninja] left the meeting 2. Large video container: [speaker view] >>load user video OR [gallery view] >>insert user video and resize all video container size 3. Filmstrip: small canvas on right-top corner with displaying name: "[participant provided name/ninja]", other users' canvas get repositioned	N/A	5	G

2.2 Host

We want our web conference to be a quickly running, easy to hop-on hop-off, installation free application. In that way, the host can be any participant in the meeting and have mostly the same access level as general users. But since we employ Jitsi API as our conferencing base architecture, the host of a meeting should have the following features and privileges:

1. If a user joins a meeting with no other participants, the user will be assigned as the host of the meeting.
2. The host will be able to modify the meeting security settings, including: enable/disable lobby mode; require a password from future participants to join the meeting
3. The host can start live streaming without asking others for permission, but when a participant clicks the start streaming button, they will be alerted that they need host's permission to complete the operation
4. The host can start screen recording without asking others for approval, but when participant clicks the start recording button, they will be alerted that they need host's permission to complete the operation
5. The host can mute all users anytime during the meeting, and before users join in the session
6. When the host leaves the meeting, he or she will be asked to grant moderator to another participant

3. Evaluation

3.1 Overview

General Overview of User Stories and Acceptance Criteria

#	User Story	Acceptance Criteria	Issue
1	As a user of this web-based, video conferencing system, I want to be able to move around between meetings effortlessly so that it simulates a real conference floor environment wherein one can easily move between conversations.	User should be able to move between meetings with 2 clicks or less, and in less than 3 seconds. This should all be able to be done by a button on the current page they are on (rather than having to leave the meeting and rejoin another one).	IG-37
2	As a user, I want to be able to have visibility over the other meetings going on whilst I am in my own meeting myself so that it feels like I am in a real conference room wherein you can see who is congregated into groups and having conversations around you.	When in the video conferencing meeting, users should be able to see what other meetings are occurring at the same time, and should have visualisation on what participants are in those other meetings	IG-33
3	As a user, I want to be able to access the web conference via the web browser so that I do not have to rely on zoom client.	A user can run web conference from their browser that supports at least three other users, with working video and audio.	IG-34
4	As a host, I want to be able to make up to 10 meetings run simultaneously so that I can effectively make use of small groups and the benefits that come from breaking up large conversations into more intimate ones.	This user story will be satisfied if 10 meetings can be initiated and run on this platform simultaneously and a preview of all 10 meetings be available to the host so he/she can be aware of participants and a snapshot of topic which is being discussed in each of those 10 meetings.	IG-38
5	As a user, I want to be able to move between multiple video meetings occurring at the same time so that I can move between these different conversations at will.	User is able to access more than one meeting at a time from the video conference platform.	IG-36
6	As a user, when I am in a meeting, I want to be able to move between meetings whilst currently in a meeting so that I can get to join another conversation easily.	Functioning buttons that transfer the user into the specified meeting. This should take no longer than 3 seconds.	IG-32
7	As a user, I don't want to have to give permission for using my video camera more than once so that it is quicker to switch between meetings.	When the user joins one meeting for the very first time, the browser will pop up the alert for permission to use the camera. Then no matter whether the user chooses to join and switch between the other meetings, quit and return to the current one, or restart their browser or computer and then back to a session, they will not be asked to give permission again. The new meeting session will automatically inherit the camera permission user granted the system if they continue using the same browser.	IG-45
8	As a user, I don't want to have to give permission to use my audio more than once so that it is quicker to switch between meetings.	When the user joins one meeting for the very first time, the browser will pop up the alert for permission to use the audio(with the request the use camera inside one pop-up box). Then no matter whether the user chooses to join and switch between the other meetings, quit and return to the current one, or restart their browser or computer and then back to a session, they will not be asked to give permission again. The new meeting session will automatically inherit the audio permission user granted the system if they continue using the same browser.	IG-44
9	As a user of this web-based, video conferencing system, I want to be able to join a meeting with my user name so that others can see who I am in lieu of me sharing my video.	User is able to enter their preferred name. In lieu of the user turning their camera on, just their name will appear as visible on the screen. This username must be between 5 and 40 characters long.	IG-35
10	As a host, I would like to be able to make sure that each meeting is able to handle at least 5 participants so that the tool can be utilised to successful for small discussions (but not too small).	This is accepted if each meeting can take at least 5 participant in it.	IG-42
11	As a host, I would like to be able to make sure that the video feed of participants in each meeting can be seen in gallery format view so that I can see everyone who is in my meeting simultaneously.	This is accepted if each meeting contain gallery view beside the current speaker view.	IG-40

3.2 Details of tests

#	Acceptance Testing	Normal	Boundary	Abnormal	Usability Testing	Current State	Link To Issue
1	A user can run a web conference from their browser that supports at least three other users, with working video and audio.	The user can access the web conference platform via the web browser, and this platform allows them to share their audio and video with others.	The platform should allow at least three other attendees to join on the call.	The platform does not successfully run without Jitsi client or does not allow audio and video sharing.	A user can run web conference from their browser that supports at least three other users, with working video and audio.	Done	IG-34
2	User is able to enter their preferred name. In lieu of the user turning their camera on, just their name will appear as visible on the screen. This user name must be between 5 and 40 characters long.	A user enters their preferred user name upon entering their first meeting. If a user's video is turned off, their user name is presented in absence of their video.	User name must be between 5 and 40 characters.	Name is not presented when the user's camera is turned off.	User can enter name upon entering first meeting. After entering user name, this name is displayed in place of their video when their camera is turned off.	To Do	IG-35
3	User is able to access more than one meeting at a time from the video conference platform.	User is able to click a button on the web browser and this will take them to a different meeting.	User is able to access up to 10 meetings at any one time from the browser.	The user is unable to move to more than one meeting from the same initial page.	Multiple buttons exist, each redirecting the user to a different meeting. Upon clicking a button, it takes the user to the meeting specified on the button.	Done	IG-36
4	Functioning buttons that transfer the user into the specified meeting. This should take no longer than 3 seconds.	A user clicks on a button to move into a meeting with people in it. A user clicks a button to move into an empty meeting.	A user can only move into an already initiated meeting.	No other active meetings for the user to move to.	A user can click on a button resulting in being moved to that meeting. Transition to the new meeting should take no longer than 3 seconds.	Done	IG-32
5	When in the video conferencing meeting, users should be able to see what other meetings are occurring at the same time, and should have visualisation over what participants are in those other meetings	A user is in a meeting and can see other meetings in progress.	A user can see at most 10 other meetings that are currently in progress.	There is only one meeting in progress, in which case there are no other meetings the user can see.	A user can see who is in other meetings whilst still in a meeting. Quality of the video streams of other meetings is enough to identify individuals.	To Do	IG-33
6	When the user joins one meeting for the very first time, the browser will pop up the alert for permission to use the camera. Then no matter whether the user chooses to join and switch between the other meetings, quit and return to the current one, or restart their browser or computer and then back to a session, they will not be asked to give permission again. The new meeting session will automatically inherit the camera permission user granted the system if they continue using the same browser.	Users can choose whether to give permission to the browser to enable their camera for meeting or not before the it's launched.	After user gives permission to the browser for the very first time, they won't have to see the pop-up alert for requiring access to camera anymore.	User's camera is disabled or missing video content on screen after the camera permission was granted.	Refresh the meeting page and/or switch to another meeting after given the permission for the first time, browser won't ask for the access to camera again. If user changed the camera setting, the meeting session will automatically update the permission status without popping user the alert.	Done	IG-45, IG-46, IG-47, IG-48

#	Acceptance Testing	Normal	Boundary	Abnormal	Usability Testing	Current State	Link To Issue
7	When the user joins one meeting for the very first time, the browser will pop up the alert for permission to use the audio (with the request the use camera inside one pop-up box). Then no matter whether the user chooses to join and switch between the other meetings, quit and return to the current one, or restart their browser or computer and then back to a session, they will not be asked to give permission again. The new meeting session will automatically inherit the audio permission user granted the system if they continue using the same browser.	Users can choose whether to give permission to the browser to enable their audio for meeting or not before the it's launched.	After user gives permission to the browser for the very first time, they won't have to see the pop-up alert for requiring access to microphone anymore.	User's audio is disabled or other participants cannot hear user's voice after the audio permission was granted.	Refresh the meeting page and/or switch to another meeting after given the permission for the first time, browser won't ask for the access to audio again. If user changed the audio setting, the meeting session will automatically update the permission status without popping user the alert.	Done	IG-44, IG-46, IG-47, IG-48
8	User should be able to move between meetings with 2 clicks or less, and in less than 3 seconds. This should all be able to be done by a button on the current page they are on (rather than having to leave the meeting and rejoin another one).	A user clicks on a button in current meeting to join another meeting with one click. User may want to leave the current meeting while joining another which adds 1 to number of clicks.	User can only join the already existing meeting.	User does not get access to other meeting through each individual meeting web page and can only access other meetings through home page.	Click on any of the meeting buttons on home page or any of the meeting pages will direct user to expected meeting. Number of clicks to join a meeting or switch to a meeting is not more than 2.	Done	IG-37
9	This user story will be satisfied if 10 meetings can be initiated and run on this platform simultaneously and a preview of all 10 meetings be available to the host so he/she can be aware of participants and a snapshot of topic which is being discussed in each of those 10 meetings.	A host will be able to make up to 10 meetings and run them at same time. The host will be able to have access to all 10 meetings and see a snapshot of whats going on in each meeting.	Host cannot run multiple meetings (max 10) at same time and needs to leave one to look into another meeting.	Host cannot run multiple meetings (max 10) at same time and needs to leave one to look into another meeting.	Host can run 10 meetings simultaneously and has access to a snapshot of participants and topic being discussed in each meeting.	Done	IG-38
10	This is accepted if each meeting can take at least 5 participants in it	A host will.be able to allow 5 people in each meeting at same time.	Host runs meetings based on the ability of the computer.	Host cannot have at least 5 people.	Host can run meeting with more than 5 people and the video is not lagging or consuming too much of computer resources.	Done	IG-42
11	This is accepted if each meeting contains gallery view along with the current speaker view functionality.	A meeting have gallery view displaying video from different attendee.	The meeting shows video from other attendees with consideration for computer resources.	A meeting does not have gallery view displaying video from the different attendee.	The function is working without any bug.	Done	IG-40

3.3 Conclusions

Utility of our tests

In the early stages of our project, we put together a comprehensive testing plan to ensure all that our testing was sufficient, and to help us adhere to the test-driven notions of XP processes. It helped us decide upon what would be in the scope of our testing, and what would not. We decided we would mainly need to focus on frontend testing. As we will be relying on the Jitsi API, back-end testing (e.g. SQL testing, API testing) was deemed out of scope for this project. We realised we would need to cover frontend testing, with a focus on the frontend functionality, GUI, usability and performance to ensure that users of the system are properly protected from bugs and errors and the system is "fit for use" by users. This front end testing would need to include:

- **Unit Testing:** To ensure that independent parts of the code work based on expectation before building the major features
- **Integration Testing:** To ensure that integrated version of single units of code will run smoothly
- **Acceptance Testing:** To ensure proper functionality of system and compliance of system with client requirements at the end of each phase
- **Performance Testing:** To ensure that stability, responsiveness and speed of system aligns with predefined benchmarks. This will include aspects such as load testing, soak testing, spike testing and stress testing.
- **Cross-Browser Compatibility Testing:** To ensure the system works well across different browsers including Chrome and Firefox, and different operating systems
- **Usability Testing (summative & formative) ** To ensure that the system which is being delivered to the end user is effective, efficient and brings satisfaction to user while interacting with it

At this phase of project we have mainly focused on familiarising ourselves with Jest as our testing framework and have created few test cases which can be found along with their results in Appendix C Unit Testing Summary. In order to form satisfactory tests, we employed the technique of equivalence partitioning. This ensured our tests provided sufficient code coverage. At this stage of the project this technique of equivalence testing has been utilised for unit testing to ensure that all possible cases are being considered. It is done by dividing input being given to the function to be tested into different equivalent classes of input/partitions. Utilising this technique provides an opportunity to the project team to minimise the number of test cases and the time taken for testing without loss of quality in testing and reduction in percentage of test coverage. It allows us to form tests that cover every scenario, without having to write an excessive number of tests.

Example of this technique being utilised in our unit testing:

The button which is supposed to take a user to the top of the meeting page should only appear if the distance from top part of the page to the part the user is viewing is more than or equal to 20px. If the distance is less than 20px the button should remain invisible to a user. The test cases have been partitioned into three separate intervals as shown below. The hypothesis behind this technique is that if one condition/value in a partition passes all others will also pass. Likewise, if one condition in a partition fails, all other conditions in that partition will fail.

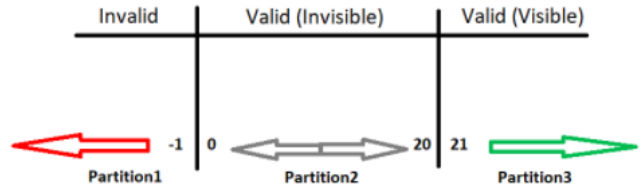
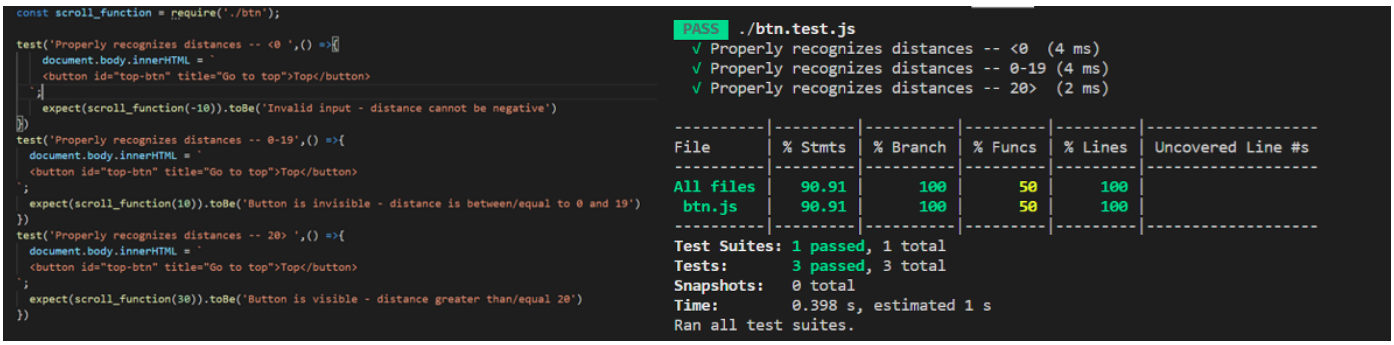


Figure 3-1: Example of the validation range on testing the "to-top" button functionality



```
const scroll_function = require('./btn');

test('Properly recognizes distances -- <0 ',() =>{
    document.body.innerHTML = `
    <button id="top-btn" title="Go to top">Top</button>
`;
    expect(scroll_function(-10)).toBe('Invalid input - distance cannot be negative')
})
test('Properly recognizes distances -- 0-19',() =>{
  document.body.innerHTML = `
  <button id="top-btn" title="Go to top">Top</button>
`;
  expect(scroll_function(10)).toBe('Button is invisible - distance is between/equal to 0 and 19')
})
test('Properly recognizes distances -- 20> ',() =>{
  document.body.innerHTML = `
  <button id="top-btn" title="Go to top">Top</button>
`;
  expect(scroll_function(30)).toBe('Button is visible - distance greater than/equal 20')
})
```

```
PASS ./btn.test.js
  √ Properly recognizes distances -- <0  (4 ms)
  √ Properly recognizes distances -- 0-19 (4 ms)
  √ Properly recognizes distances -- 20>  (2 ms)

----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |   90.91 |      100 |      50 |     100 |
 btn.js   |   90.91 |      100 |      50 |     100 |
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.398 s, estimated 1 s
Ran all test suites.
```

Figure 3-2: Testing code and result that examine the "to-top" button functionality and validity

Limitations of the System and Tests

As previously mentioned, we are using Jitsi as the backend of our project. This has limited us in our control over the system. We are able to alter aspects of the server side code, controlling aspects such as video resolution and the visual aspects of the video platform but having a such a great reliance on this Jitsi server-side code which we did not write ourselves has meant that locating where this code executes certain functionalities is particularly challenging. A key difficulty stems from 'minified' files. These files are incredibly difficult to read, and hence are difficult to discern what they are responsible for. For example, function names have been reduced to one letter, and similarly, variables are reduced to just a single letter also.

This difficulty experienced in understanding and manipulating the server-side code has meant that we have dedicated more of our time to research and drafting system diagrams, rather than directly coding. It has meant that we are still struggling to implement the functionality of being able to provide visibility of all concurrent meetings, whilst a participant is currently in a meeting. We plan to do this by isolating wherein the Jitsi code Jitsi is accessing participants' webcams, and redirecting these streams in low resolution to the meeting page of all participants in all meetings. However, the responsible section of the Jitsi code has not been easy to isolate.

These limitations of the system have also impacted our ability to test the system. Our difficulty in understanding and controlling the Jitsi code has made it challenging to test the backend of our system. At this stage of the project, we have mainly focused on familiarising ourselves with Jest as our JS unit testing framework and creating few test cases and during the pair testing session. It has been found that since the majority of JS code is embedded into "external.js" file which has been used from the jitsi-meet project source code, it is challenging to cover all individual modules and functions in unit testing.

Furthermore, although we have prepared a general test plan which covers the type of tests we are aiming to conduct, we have not specifically set a schedule for conducting our higher level tests including integration, system, acceptance and usability testing. What is more, even though we have used our platform for video conferencing with 16 participants in one meeting, we have not yet tested the system with more than two concurrent meetings, and with more than 16 participants. Having said this, this is dependent on the server, and as such, upgrades to the server will support this.

4. System structure overview

4.1 System Specification and Design

There was a critical issue, in which our team switched from ZOOM API to Jitsi API during the development process. Thus we generated two sets of user stories to represent users' needs appropriately. However, even though we changed our codebase, we still count the implemented Zoom functionalities as part of our progress. Moreover, the technical constraints, design thoughts, and the problems we dealt with when using Zoom were and will continue to provide excellent guidance throughout our Jitsi-based development life cycle.

Therefore, we will mention work and concepts from both aspects of Zoom and Jitsi. Elements related to Zoom will not appear in the sections related to the current Jitsi system structure and specifications. Still, they will show up in the implementation progress and parts related to context or requirements in a more general way.

System Requirements

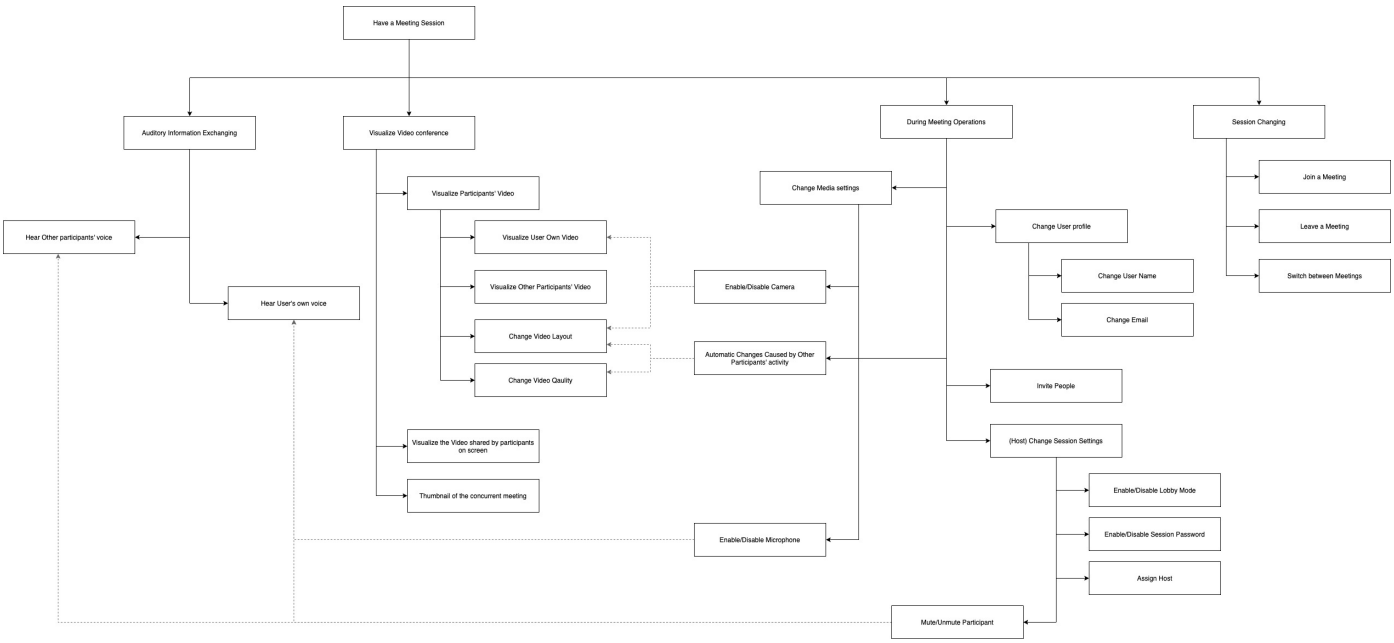


Figure 4-1: general system functionality break down

Figure 4-1 presents a breakdown structure of what functions and operations the system needs to carry out. The grey dotted arrows indicate that those operations don't have any belonging relationships, but act in a sequence or as subtasks when one operation or function is being called and executed. The user operations stated in the user stories are achieved by executing a single function or an integrated set of functions from the minimal functions shown above. The system will perform these operations in a specified context under a critical sequence to provide users their desired output.

User Stories

For the current sprint version, we managed to complete a few user stories' requirements. However, due to the late switch of the API codebase, we are continuing to modifying the Zoom related ones and have been adding more stories to fit better into the Jitsi structure. We will briefly go through a few stories that we completed now, and will provide more details in the implementation section of this page.

The first user story completed was providing the ability for a user to access the web conference via a web browser. This user story was completed for the Zoom API in the first sprint and was completed again using Jitsi Architecture.

The second allows a user to join a meeting using a username so that other users could identify them. This was achieved using the Zoom API but has not yet been completed in the Jitsi version. We now allow users to modify their name using the functional buttons once the have joined the meeting. However this process should happen before the user fully joins the meeting. This will be tackled in a future sprint.

The third user story we approached was providing the ability for a user to move between multiple meetings which were concurrently occurring, which was completed for both the Zoom and as well as the Jitsi API. We will continue working on improving the performance of this process to provide better user experience, but in regard to functional requirements, this user story is complete.

The next user story was to support at least five participants per meeting that was currently taking place, allowing for small discussions. We flagged that when more than 5 participants joined the meeting based on Zoom API, the consumption of CPU is noticeably high and the videos on screen have a significant delay and the frames are not continuous. This was one of the reasons the client decided to focus on a new API (Jitsi API). However, this user story was successfully met for the Jitsi API, and was tested to able to support a meeting with 10 out of 15 participants with their camera on without impacting quality.

Another user story required the ability to have gallery view. This was the main reason the client wanted to move away from the Zoom API. Gallery view was not implemented in Zoom API, and all custom solutions had CPU usage that was far too high and there was significant lag. The Jitsi Meet API, however, fully supports gallery view in their code with an acceptable amount of CPU usage and lagging was not an issue. That is to say, with the current Jitsi API structure, this user story has been met.

The final two user stories state that the user should be and only be required to grant permission for use of their microphone and camera once upon joining a meeting for the first time. No matter whether the user decides to join a new meeting, switch between meetings, quit and return to the same meeting, or even restart their browser or computer, the alert should not pop up again. We approached those two stories in the first half of the project with the Jitsi API.

Technical Constraints

The main technical constraint specified by the client is the reduction of CPU consumption. This constraint arose when trying to support gallery view using the Zoom API and resulted in the shift from the Zoom API to the Jitsi API.

Other technical constraints such as security are largely supported and managed within the API. For example, we want to achieve multisession communication, and specifically being able to have a thumbnail of other ongoing meetings. Technical constraints may include thinking about questions such as: how much detail can I preview the other meeting in? Where can I get access to the media stream of that session? Is there a legal process, which means the data transferred is properly encoded and decoded by API or I have to decrypt the media track? Hence, limitations and security checks should be carefully designed and put as a primary constraint.

4.2 System Architecture

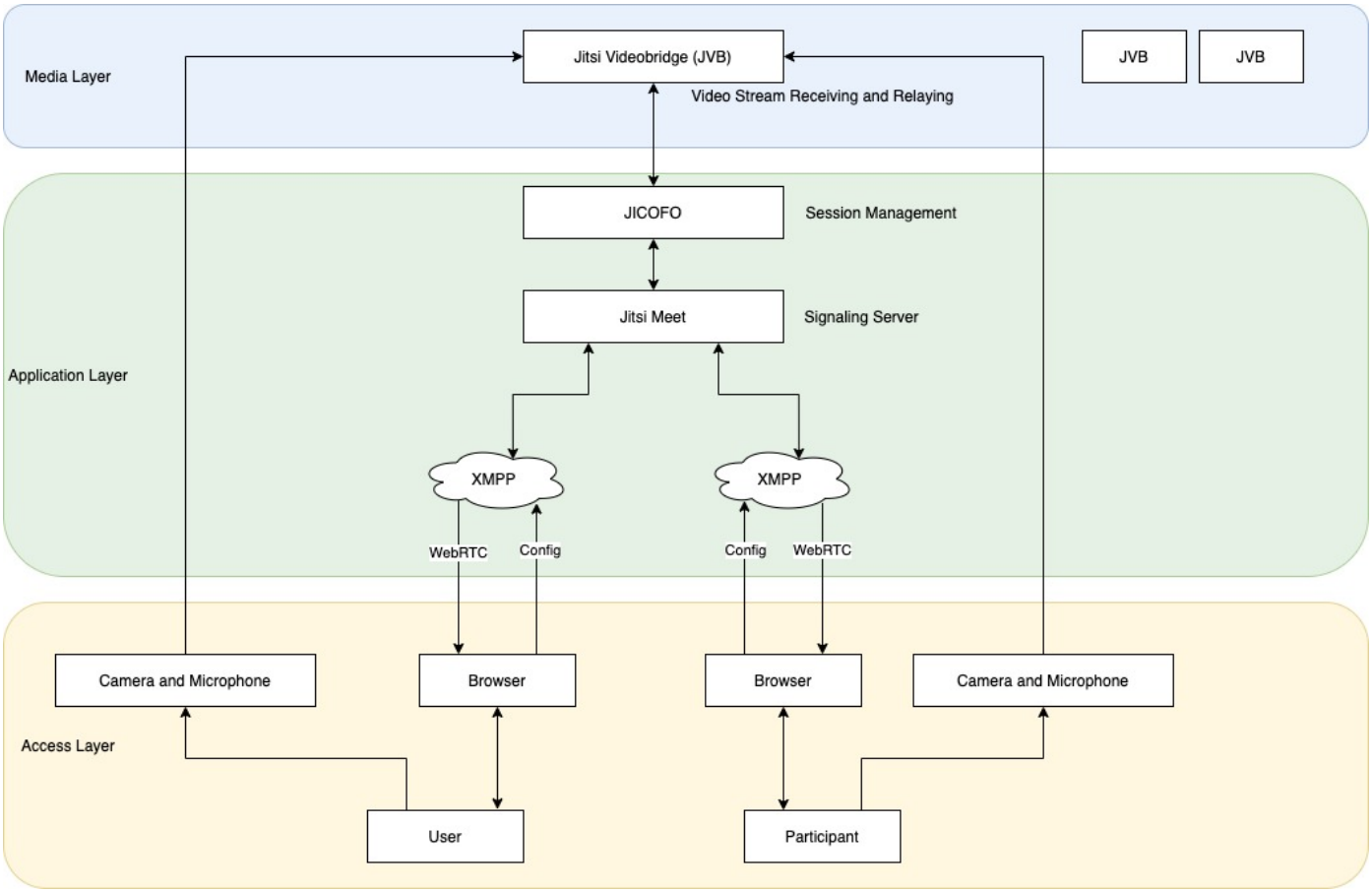


Figure 4-2: High-level System Architecture Diagram of Jitsi-based Web conference Application

Generally, our web conference application has three layers:

- the Access Layer, where the user directly interacts with the frontend structures via the browser.
- the Application Layer, where the online meeting is configured, signalled, and get the meeting information managed.
- the Media Layer, which is the place that receives the media tracks, decrypts and encodes them again, and relays to the participants of the meeting.

Exploration into System Layers

*** Client Side: Access Layer***

Since we employ the Jitsi API to hold the application's conference, the Application and Media Layer will simply follow Jitsi's architecture. The access layer, which can be understood as the client-side, consists of Jitsi-Meet and the frontend UI we built, and the user interface is where most of the events that happened can be visualised.

In general, the components that are physically involved in this layer are human and computer. To get the meeting functionality fully enabled, the user's computer needs to have a Web (IP) camera and microphone, which can be satisfied by almost every computer around the world now. For the browser, Jitsi has provided a solid requirement table for different browsers.

	Safari	Chrome	Opera	Firefox	Edge
Least Version	10.2	34	22	52.4	/
Audio Input	Yes	Yes	Yes	Yes	Yes
Audio Output	Yes	Yes	Yes	Yes	Yes
Video Input	No	Yes	Yes	Yes	Yes
Video Output	No	Yes	Yes	Yes	Yes
Screen Sharing	No	Yes	No	Yes	No

Table 4-1: Requirement and Capability Table for browsers

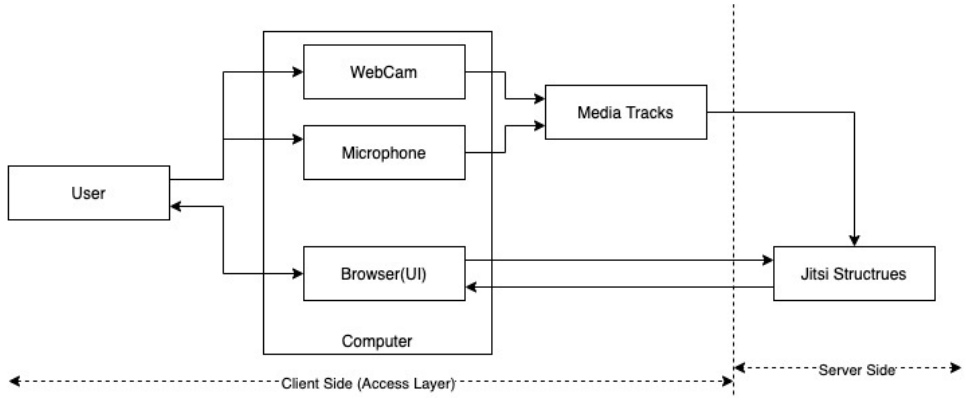


Figure 4-3: Abstract Diagram of the Components involved in Access Layer

As shown in Figure 4-3, there are two streams that users will communicate with the server structure via a local device: Users provide data including meeting and personal participant information to the browser UI, then the UI transports those data to Jitsi Structures (demonstrated in next section). The backward data flow would be a Jisti host configured meeting session on the user's browser, along with the media tracks from the other participants of the meeting. Once the user has granted the Jitsi structure access to the microphone and webcam via UI, they will capture the user's media track, then encode and directly send it to the Jitsi Structure. Through there, user's media track will be relayed to the other participant's screen and audio channel.

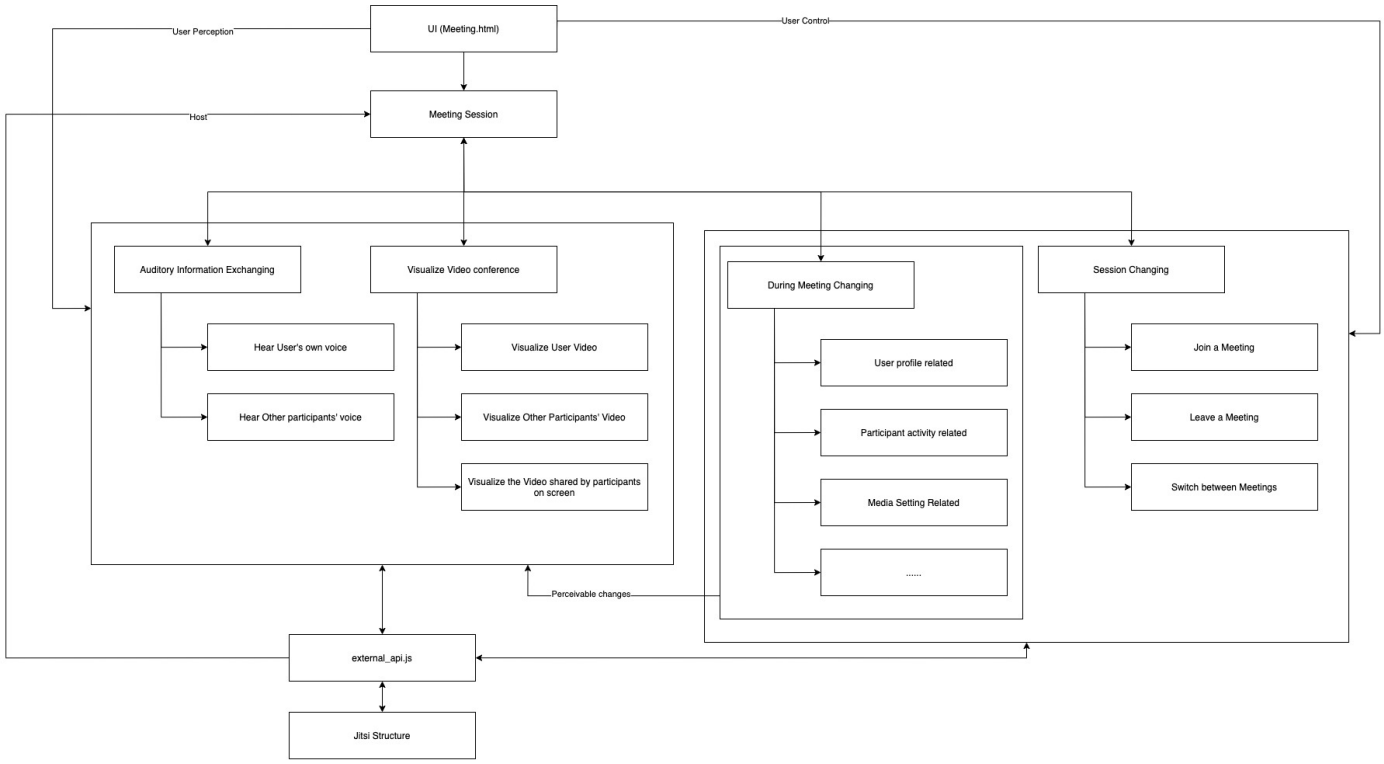


Figure 4-4: Preliminary diagram on the connection of code files and meeting session control

Figure 4-4 above shows the abstract structure of what the system should do, how the developed UI and the Jitsi side communicated, and together takes control over events. The external_api.js that is provided by Jitsi is the most critical part of the whole application to get working. This file acts as the key to the Jitsi structures to call server side functions that actually manipulate the components on the screen.

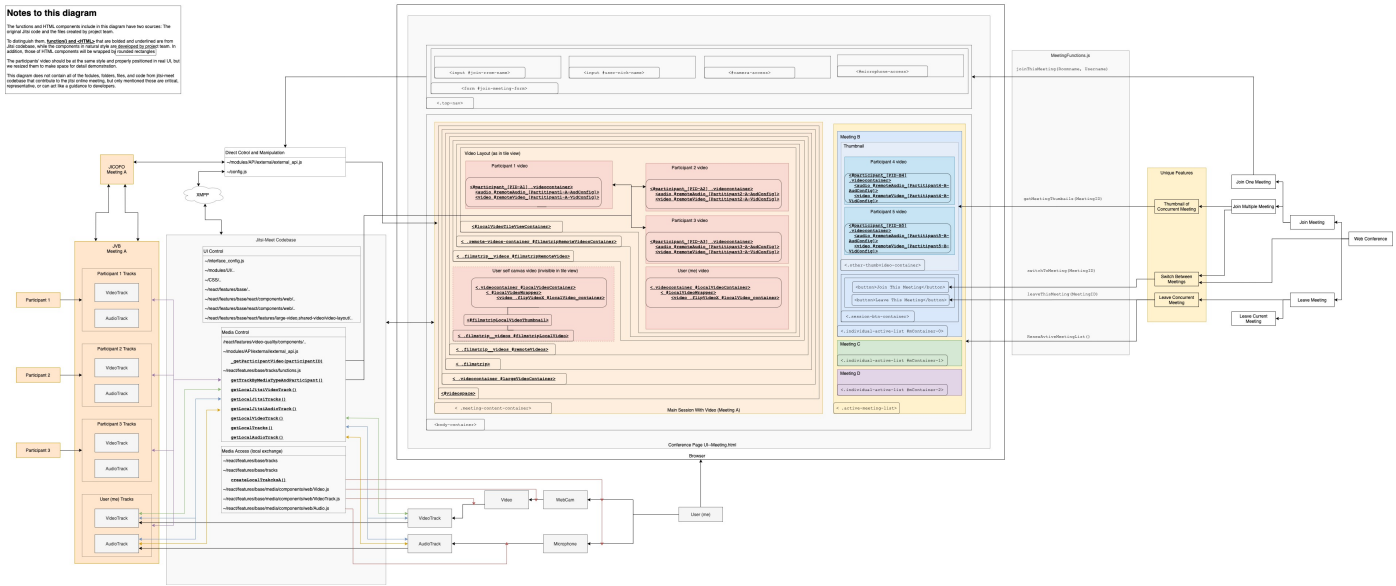


Figure 4-5: Enhanced and detailed version of diagram on the connection of code files and meeting session control

In Figure 4-5, we break down the events on the screen into primary HTML components, and link them with the actual code, files and folders that we considered involved in the process of controlling the meeting session. Jitsi provides the structure to actually access and manipulate the session settings, and our developing code provides various additional interaction methods such as simplifying the process of moving between meetings. We will go through the process later to provide a deeper insight into the collaboration across layers, and explain how the Jitsi Structure manipulates the events happening on the user's screen.

Server Side: Application and Media Layer

Except the first Access layer where the user directly interacts with the system, there are two more layers that make things happen in an invisible space, which are the Application layer and the Media Layer. However, we can take those two parts together and refer to them as "server-side": "Jitsi Meet Layer", and as well as "Jitsi structure". This is where the Jitsi meeting actually takes place, where sessions are managed, and relay participants media tracks.

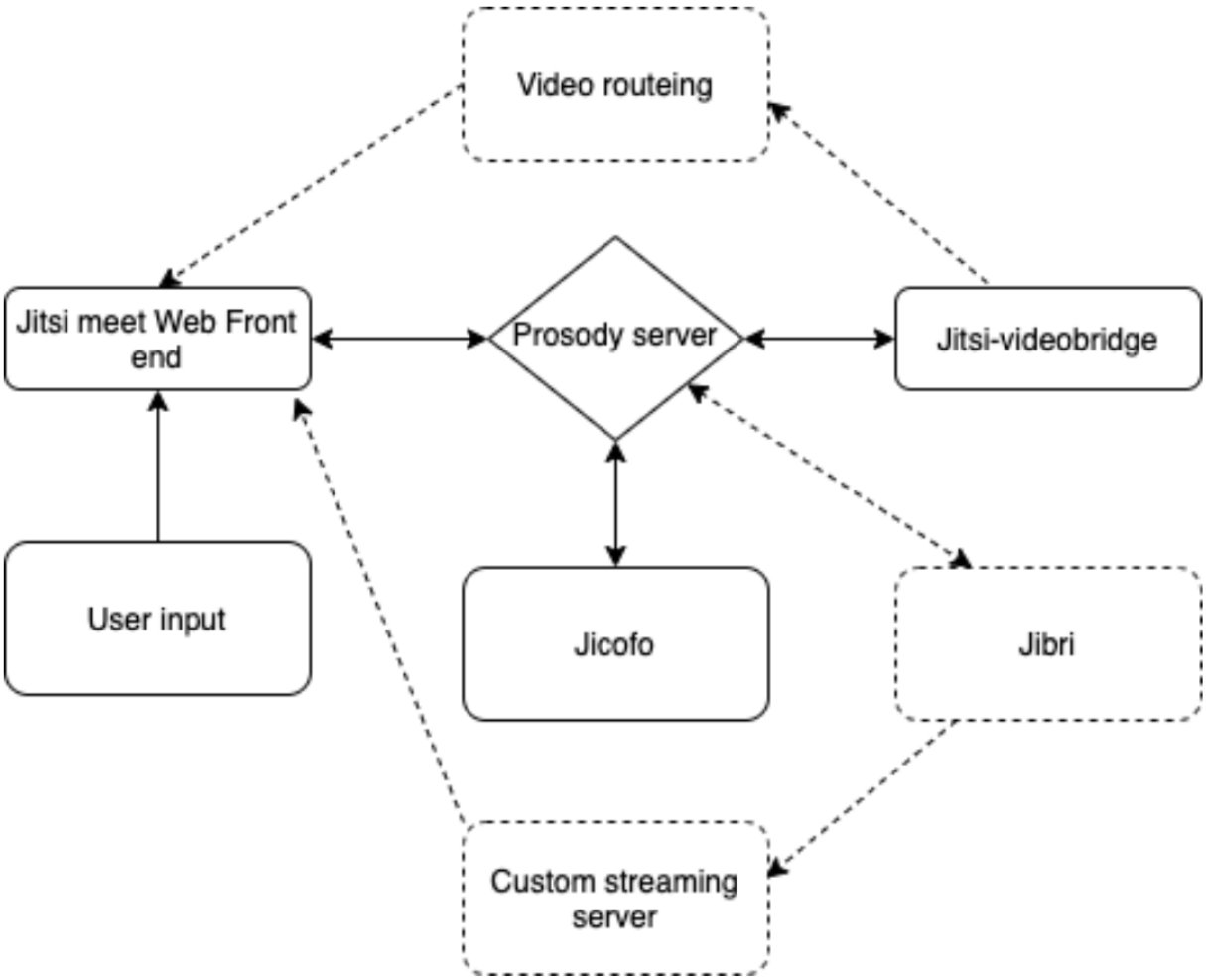


Figure 4-6: Sample Diagram of the components of Jitsi Structure and the communication flows between them

The Jitsi Structure consists of several sub-systems, as shown in Figure 4-6, but only three of them are considered the main components for a Jitsi session:, Jitsi meet, Jicofo and Jitsi-videobridge(JVB).

- Jitsi Meet front-end is where the user would join the meeting through the browser. The communication is established using WebRTC protocol. After receiving the input, Jitsi meet will convert them into XMPP format and transfer to XMPP server. The input is then passed to Jitsi-videobridge and Jicofo for processing.
- Jicofo is the session managing sector in Jitsi Structure. It manages the meeting information and media session between users and Jitsi-videobridge. It also manages and establishes a Colibri channel to communicate with Jitsi-videobridge.
- Jitsi-videobridge handles stream relaying and forwards the selected streams to other attendees in the same meeting. It receives the encrypted media track directly from participants' devices, decrypts them and then encodes them again. This is then relayed to the other participants that are joining the same meeting session.

In simple words, users can join more than more Jitsi Meeting session, but each session is managed by one JICOFO sector and recieves media tracks transmitted via one JVB Structure. Hence, we can provide a system components structure diagram that contains the essential parties to perform multisession cross layers communications and a few screen components. In that way, we can clarify if it affects both the operating user and the other participants in the meeting.

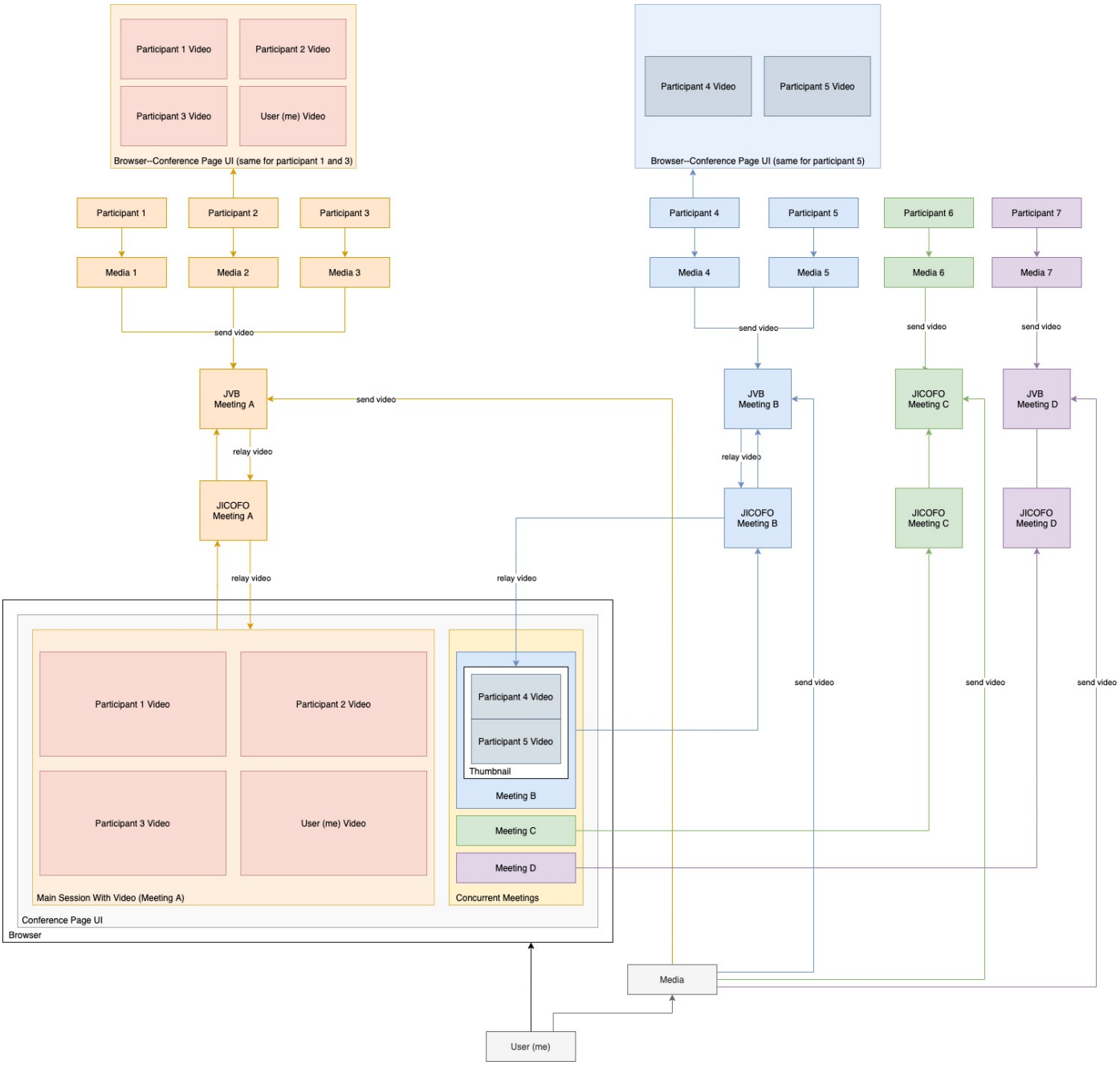


Figure 4-7: High-level system components digram

4.3 User Stories Implementation

Implementation Progress

So far, a number of user stories have been completed across various sprints. It is important to note that sprints and deliverables were not necessarily a fully complete user story as per the client's instructions. The client preferred to build up to a complete user story over a number of sprints due to the complexities involved. We managed to get several user stories fulfilled by the end of week 4. However, the client requested the use of a new API base (other than the Zoom API) to achieve a few more unique functionality, meaning that most completed user stories needed to be completed again. As described in the sections before, we moved to the Jitsi API codebase.

The first user story completed was providing the ability for a user to access the web conference via a web browser, not the Zoom client. This user story was completed for the Zoom API in the first sprint and was completed again using the new API option. This user story required a user to be able to run the web conference software

produced in a browser. Non-functionally, this should be completed with no more than a single click and low latency.

Moving on from this, we then focused on allowing a user to join a meeting using a username so that other users could identify them. This required a text box entry allowing the user to enter their name, which was to be between 5 and 40 characters. Non-functionally, this was to be done with a single text box entry and a join meeting button, limiting the number of clicks to two (one for entering the name and one for submission/joining). This was achieved using the Zoom API but has not yet been approached for the new API and will be tackled in a sprint.

The third user story we approached was providing the ability for a user to move between multiple meetings which were currently occurring. This required the addition of buttons to the home page allowing the user to choose the meeting they wished to join, but also being able to return to the home page to join a different meeting. This needed to take place with a single click, which is why we introduced new buttons, rather than dropdown options. This was completed for the Zoom API as well as having been completed with the Jitsi API. Following on from this, we approached the user story that required the integration of those buttons to other meetings into the meeting page itself. This required the user to be able to swap to any of the ongoing meetings from within the meeting they were already in. Non-functionally, it needed to happen based on a single click (of the required meeting button), and the meeting swap needed to happen in no more than 3 seconds. This user story was also completed for the Zoom API version as well as the Jitsi API version. The next user story was to support at least five participants per meeting that was currently taking place, allowing for small discussions to take place. From a non-functional perspective, the quality of the image and audio should not differ drastically with the addition of more participants. This user story was met for the Zoom API, however, there was noticeable lag when more than 5 participants joined the meeting, which was one of the reasons the client decided to focus on a new API (Jitsi API). This user story was successfully met for the Jitsi API, and was able to support up to 10 participants without impacting quality.

The subsequent user story we approached was the ability to have gallery view. This was the main reason the client wanted to move away from the Zoom API. Using the Zoom API, gallery view was not supported, and all custom solutions had CPU usage that was far too high and there was significant lag. The new API, however, fully supports gallery view. Moreover, there was an acceptable amount of CPU usage and lagging was not an issue. That is to say, with the new API, this user story has been met.

The final two user stories we approached in the first half of the project were requiring permissions for microphone and camera once. As a user, permissions for each of these should be required once upon joining a meeting for the first time, and then should not be requested again, even if the user switches meetings, quits and returns to the same meeting, restarts their browser or computer. From a non-functional standpoint, this should be achieved in a single click per permission. These two user stories have been met with the Jitsi API.

It should be noted that the difference in approach to user stories between the Zoom API and the new API is largely because of the reasons the API was changed. Different user stories were targeted for the second API to ensure that it would be able to succeed where the Zoom API had not resulting in completion in a different order.

Future Plan

Throughout the second half of the project, we will aim to finish all user stories within the first few weeks. Once all basic functionality has been achieved, our focus will shift to improving the user experience and the user interface, which were declared per the client's description as the primary focuses of this project.
Currently, the system we have developed fulfils the majority of the required functionality, and we are not anticipating much will change in this regard. Thus no storyboard or mock-up is provided as all this information can be found in the Overview of System From User View section of the report.

The first user story we will tackle will be to allow visibility to other meetings, which means that a user is able to see who is in other meetings from within the same page as their meeting currently.

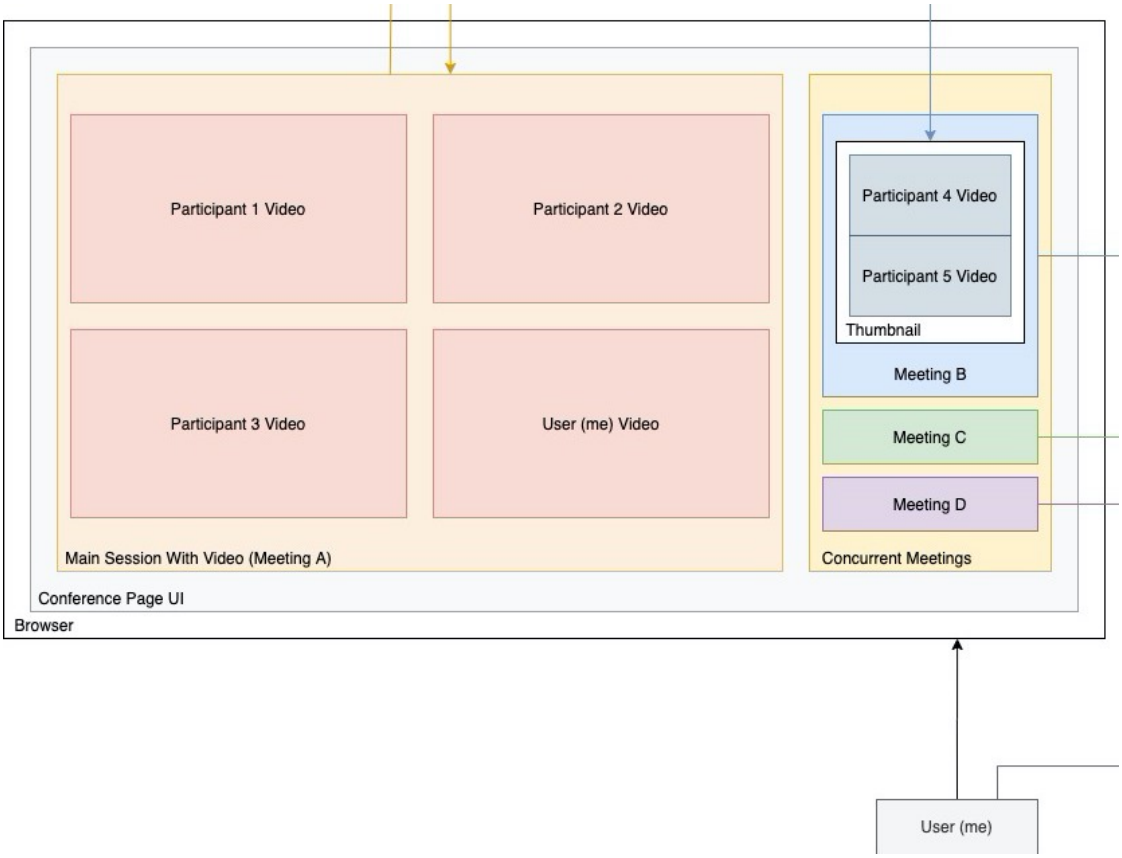


Figure 4-8: Sample screen when user have a thumbnail of another concurrent meeting

Following this, the user story specifying the ability to join using a username will be approached. At the moment, a user can enter a username once currently in a meeting. However, as specified in the user story, this should happen before the user joins a meeting.

Once the existing user stories are complete, we will elicit further user stories that are functional. However, we anticipate that from this point, most user stories and further work will be focused on user experience and enhancing the user interface.

5. Tools used to build system

Upon commencing this project, we used the Zoom API to start creating our web conferencing platform as per the initial requirements provided by the client. However, after achieving a working web conference, with the ability to switch to other meetings from the current meeting you are in, it became apparent that the Zoom API would not support the further deliverables required from us. Extensive research lead us to realise that we would not be able to produce a 'gallery view' using the Zoom API. Whilst we could dive into the code and manipulate it to obtain this view, these changes would lead to massive CPU consumption, failing the performance requirements for this project.

As such, extensive research was conducted into alternative APIs that we could exploit for our project. We compared bandwidth, CPU consumption, the number of participants each would support and the functionality offered, across different platforms. After summarising our research , we presented our findings to our client and collectively agreed to progress with the Jitsi API.

This platform was able to offer us all the functionality we had received from the Zoom API. Additionally, it also offered gallery view functionality. It would allow for up to 35 participants to join a meeting before quality began to degrade and provides complete access over the code, meaning we could achieve complete control. Whilst it did require high CPU usage, the platform offered control over the video resolution so that this usage could be mitigated.

5.1 Languages used for each part (refer back to Section 4)

- Access level: our access level consists of a webpage with a customised user interface. The languages used are CSS, HTML and javascript. The reason we use HTML is that HTML is both easy to learn and easy to implement. This allows us to reduce the time needed to produce our deliverable. Another big advantage is that HTML is supported by every browser. This is very important as we want to make our project as inclusive for all users as possible. CSS is chosen for formatting due to its ability to provide spontaneous and consistent changes to our website which also saves time. Javascript is used due to its speed and popularity.
- Application-level and Media level: Components in these two levels are based on the Jitsi-meet project retrieved from Github. The languages used are java and javascript.
- Testing: Our unit testing was conducted using Jest. The most coding we did is in the access level and was written in Javascript. Jest provides a simple and fast testing environment. It can be directly installed using npm and does not require other dependencies. It is also fast compared to other traditional tools.

5.2 Source code management

- Git and Bitbucket: Git allows us to track files changed in the group repository. As a team, git allows us to always work on the same version and merge our own work together. It also enables us to track each other's progress. Bitbucket provided an integrated environment which includes issue trackers, wiki and online git hosting.
- Source tree: Source tree is a desktop app for git management that has a GUI. It allows us to more easily manage the files and use the git function without having to use terminal commands.
- Visual Studio Code plugin: Visual Studio is heavily used in our coding process. Hence it is logical to also use it for version control too.

5.3 Server installation

- Jitsi server: The server that is serving our Jitsi instance is provided by VULTR. We are using a high-frequency VPS which has 1GB of ram, 32GB of NVMe and 1cpu. The operating system is 64-bit Ubuntu 20.04. VULTR and this specific set up is selected based on a balance between budget and performance. This service provides a strong CPU at a very low price. Considering we do not need much RAM and Disk storage, we decided to use this server.
- Amazon Route 53: For domain registration and DNS management, we choose to use Route 53 provided by Amazon. It's reliable and cost-effective.
- SSH tool: In order to access and change files on the server, we used SSH tools to connect to the server. As team members are using a different operating systems, we have to use two different tools. For mac users, Termius is the chosen tool. It's easy to install and easy to use. It's UI is very friendly and stylish. For Windows users, we choose PuTTY for a similar reason. It's easy to use and does not produce additional studying costs.

5.4 Source code editor

- Visual Studio Code: The main part of coding for access level is done in VS Code. It can handle multiple languages which are used in our project. It also has multiple plugins such as Bitbucket which significantly cuts down work needed to access our group repository. It comes with all the normal features such as syntax highlighting, bracket-matching, auto-indentation, and variable tracking. It is also very fast and light-weighted.
- Vim: Vim is used to edit code on the server. As our server is not running a Graphic UI, we need to use a command-line text editor. Vim is our choice because not only is it quite powerful and has many plugins but also is our most familiar text editor. It is consistent with our choice of tools which is creating as little study cost as possible.

6. Information Search

6.1 Research into the Zoom API and getting familiar with javascript, CSS and HTML

When commencing this project, all group members had limited experience with front end development, APIs and using Bitbucket as a version control platform. As a result, we consulted YouTube tutorials to learn the languages javascript, CSS and HTML, and learn about APIs (What are APIs?). We used an Atlassian tutorial, Learn Bitbucket with Git, to become familiar with Bitbucket and Git. Throughout the project, any time a team member found a particularly useful resource, they made sure to share this resource with the rest of the group via the #resources channel on slack. This initial research allowed us to establish sound foundations upon which we could begin our project and dive into the Zoom API.

The first task presented to us by our client involved creating a working web conference using the Zoom API; one that works through the browser rather than Zoom client. As such, we needed to familiarise ourselves with the Zoom API, how it works, how we could make use of it in our project, and how we would make it run. We consulted the Zoom API Documentation and a Zoom sample web app which we were able to adapt for our own code. We were also able to use a guide for building Zoom SDK app to help us establish a minimum viable product. Beyond this, we also had to research, and educate ourselves on npm, a package manager for JavaScript, and what we used to run our minimum viable product.

6.2 Research into Sandboxie

In week 3, our client asked us to look into "sandboxie":

Part of the requirements of this project was to add the functionality of being able to get access to a snapshot of what is occurring in other active meeting sessions and who the participants are within those meetings. One way of approaching this requirement was to send an invisible participant to different meetings and capture media of those meetings through the presence of that participant. For this purpose, we were asked to do research into sandboxie, software which gives us the ability to run multiple instances (different meetings) of one application which, in this case, would be our web conference through a web browser at the same time as using different sandboxes (isolated environment). In the research, it was investigated that sandboxie could be installed and run only on Windows. Since most of our group members are Mac users we looked into alternatives of this software for Mac and Linux users. Deep freeze, BufferZone and SHADE Sandbox were identified as mac alternatives of sandboxie (Teresa, 2020). Moreover, the research revealed the major differences between a sandboxing software and a virtual machine which can be considered as an alternative to it. An abstract demonstration of how each of these two techniques work has been provided below:

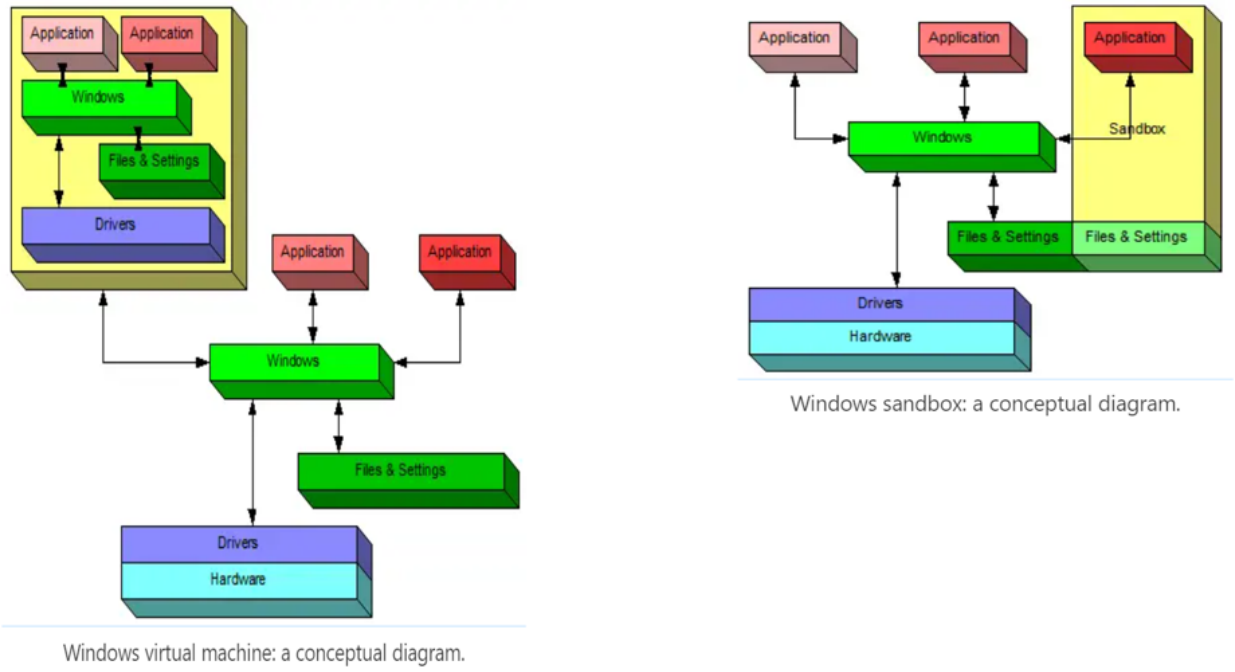


Figure 6-1: Comparison of virtual machines and sandboxie.

The diagram demonstrates that since the virtual machine runs a totally separate copy of the operating system (i.e. Windows), extra memory (RAM) and disk space will be allocated to it whereas using a sandboxing software does not require much in regard to additional space (Notenboom, 2012). Moreover, setting up a virtual machine has a more complex and time-consuming process compared to the download and installation of sandboxing software. Therefore, it has been extracted from research that a sandboxing tool can be considered as a more reliable option compared to its alternative VM which is a resource-hungry option for purpose of this project.

6.3 Research into alternative video platforms

Trying to find ways to support gallery view

Initially, this project was expected to use the Zoom API in order to create a web browser conference platform. However, after some work and completion of some user stories, it became clear this would not be an option. Extensive research into the Zoom API for web development revealed that aspects of the final product which were necessary would simply not be possible using the Zoom API library. The main issue that Zoom had was the inability to provide a gallery view using the API. Our research on this topic mostly stemmed from reading through the Zoom API documentation and Zoom forum posts. Whilst this is an issue Zoom is aware of, and one they were hoping to have resolved by Q2 2020, this has not been achieved so far.

We attempted to recreate this gallery view representation by manipulating the source code. We were able to find sources that helped us calculate the video size and ensure that all videos inside the container would occupy as much room as possible without overflowing (Dosov, 2020). Despite successfully getting multiple videos to appear on the screen, in a manner similar to gallery view, this gallery view functionality was not attainable through manipulation of the code without having a massive impact on CPU consumption, meaning there was discernible lag. Extensive research, alongside the Zoom web API forums, made it clear that this was not something we would be able to work around. As such, further research was needed to find an alternative API that would provide the functionality that was required.

Research Zoom API alternatives

Multiple alternatives to using the Zoom API were looked at and over the course of Week 4, a compilation of options was made. This compilation was carefully detailed to include performance measures such as CPU usage, bandwidth usage, the maximum number of participants allowed in each meeting session and other main functionalities of each of the covered options. Research revealed that among the 5 options that we had, WebRTC and QuickBlox required payment for premium plan and

therefore we have omitted them from our list. Following this, we selected the best three options of Jitsi, BigBlueButton and OpneVidu to pitch to the client in the next client meeting.

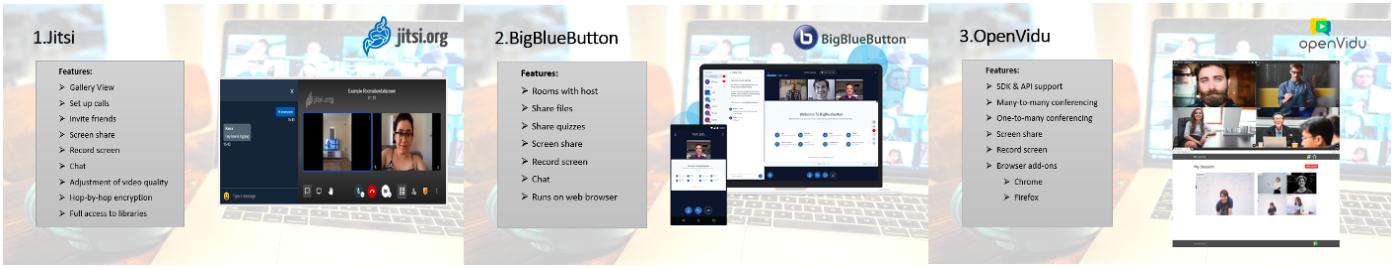


Figure 6-2: Screenshots from presentation to Client when outlining our options for alternative APIs.

Further analysis done on these three preferences disclosed that although BigBlueButton requires lower network bandwidth of 1MBps and supports larger number of participants (100p) in comparison with Jitsi but it does not support peer to peer video conferencing which is a fundamental requirement of this project (GCC, n.d). Moreover, even though OpenVidu has better bandwidth usage and more consistent CPU performance compared to Jitsi but it supports only 25-30 participants in a meeting and does not allow manual adjustment of video quality which is an important feature for this project as we are required to get access to lower quality version of other meetings' participants media (OpenVidu, 2018). Therefore, Jitsi stood out and was selected as our number one choice which supports gallery view, live streaming, manual video quality adjustment and full access to the libraries.

6.4 Research into Jitsi Architecture

In week 5, we had achieved the functionality of having multiple meetings occurring simultaneously, and being able to switch between these meetings from the meeting page itself. The next step was to establish visibility over these concurrent meetings; that is, a user should be able to see who is in each meeting, even if they are not in the same meeting.

As such, we needed to gain a deeper understanding of the Jitsi-Meet architecture, and how Jitsi-Meet handles the routing of participants' video stream, from the user in question's video camera to the screens of the other meeting participants. Additionally to this, we needed to understand just how this media was being encrypted, and how we could intercept these video streams before they left the Jitsi-Meet server. By understanding how this worked, we hoped to be able to gain control of this video stream and use it for our own purposes of enhancing visibility across meetings.

Initial research into Jitsi-Meet unveiled the following system flow. Jitsi-Meet is the web front-end; what we, as users, interact with and see. Beyond Jitsi-Meet, Jitsi also consists of a web server, an XMPP server, a focus component and video bridges (Luo, 2016). The user interface is run through a browser that employs WebRTC. WebRTC is an open-source project that enables web browsers to achieve real-time communication through APIs. Ultimately, it permits the usage of audio and video communications within web pages without needing to install plugins or download native applications (WebRTC, 2020).

An XMPP (Extensible Messaging and Presence Protocol) server allows Jitsi to connect each component in its architecture, and provide text chatting facilities. The Focus Component, Jicofo (Jitsi Conference Focus), is present to guide audio and video streams to the video bridge. Lastly, Jitsi video bridge handles the stream mixing and relaying.

Jitsi also includes the component Jibri. This component provides recording and streaming services for the Jitsi-Meet conference. Jibri acts as a silent participant inside a conference, routing users' streams so that they can be recorded or live-streamed to YouTube (Jitsi, 2018).

The system flow of a Jitsi web conference begins with a user entering the web conference. As a user joins a web conference, WebRTC protocol is utilised to establish the request to join the MUC (Multi-User Conference). This, in turn, is transformed into XMPP format as a video or audio extension by Jingle, the XMPP method for transmitting multimedia data (Luo, 2016). RTP is the actual transport protocol underneath this.

After a user has successfully joined the web conference, a special user called Focus materialises in the Multiuser Chat Room. This XMPP component is in charge of allocating transmission channels for media onto video bridges and back to the users. This focus component creates a Jingle session between Jitsi videobridge and the participant, enabling media to flow between the participant and the videobridge. This is possible as the Focus user allocates Colibri channels on the Videobridge, using them as its own Jingle transport (Luo, 2016).

Video bridges are another XMPP component that enables us to achieve multi-user video communication. Multimedia streams with different bandwidth capacities pass through the videobridge, and the Jitsi Videobridge mixes all the audio streams and replays the video streams. Ultimately, once multimedia channels are allocated on the Jitsi Videobridge, users in the web conference can send and receive audio and video streams from the Videobridge. The Videobridge has the capability to adjust the transmission rate and quality of the audio and video streams, adjusting them according to the users' current condition. As a result, the users of the web conference are presented with a smooth, and apparent high quality, web conferencing experience (Luo, 2016).

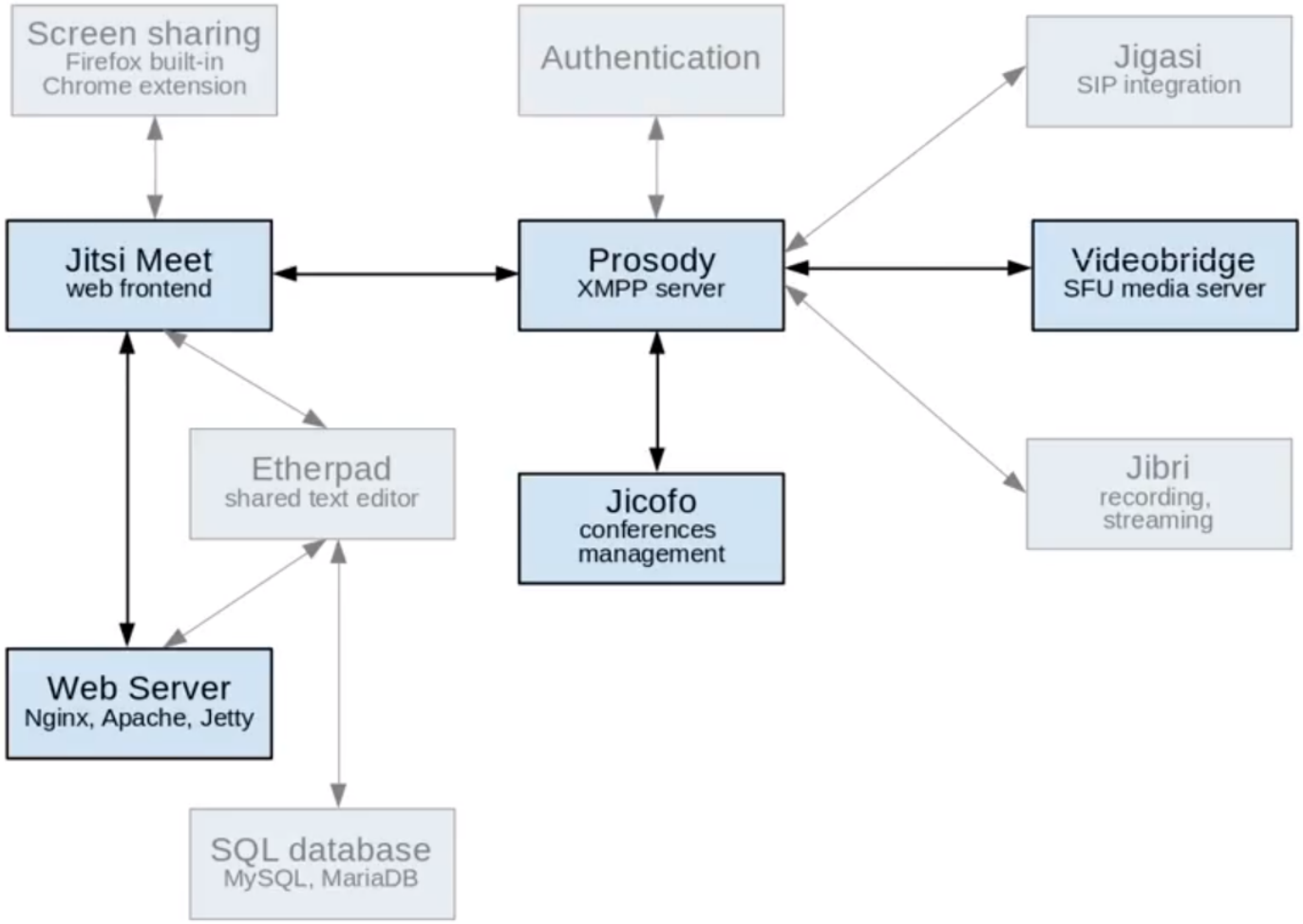


Figure 6-3: Overview of Jitsi's main components.

6.5 Research into encryption

Following on from understanding the architecture of Jitsi, we next needed to understand how the multimedia data was being encrypted. Being WebRTC compatible, the encryption for Jitsi Meet is provided by the browser-like environment in which it is run. All media transmissions between the client and the server are encrypted by WebRTC, which mandates that DTLS-SRTP is used for encryption. This means that were the data intercepted, the interceptor would not be able to make sense of it. By using this means of encryption, it means that data streams are encrypted using Datagram Transport Layer Security (DTLS) and media streams are encrypted using Secure Real-time Transport Protocol (SRTP) (Jitsi Meet, 2020).

As DTLS-SRTP is strictly tied to PeerConnection, when using a video router such as Jitsi Videobridge, DTLS-SRTP can only provide hop-by-hop encryption. This means that, in order for the media from one participant to reach another participant, it is encrypted, sent to Jitsi Videobridge where it is extracted from the sender's crypto context and when re-encrypted by Jitsi Videobridge with the receiver's crypto-context. Ultimately, Jitsi Videobridge ends up establishing as many channels as there are participants (Jitsi Meet, 2020).

After better understanding the system architecture and encryption methods, we were able to create a system diagram and start digging into the code to attempt to narrow down where this video was being captured, and where the encryption was occurring. The findings of this investigation were summarised on our wiki week 5 research page.

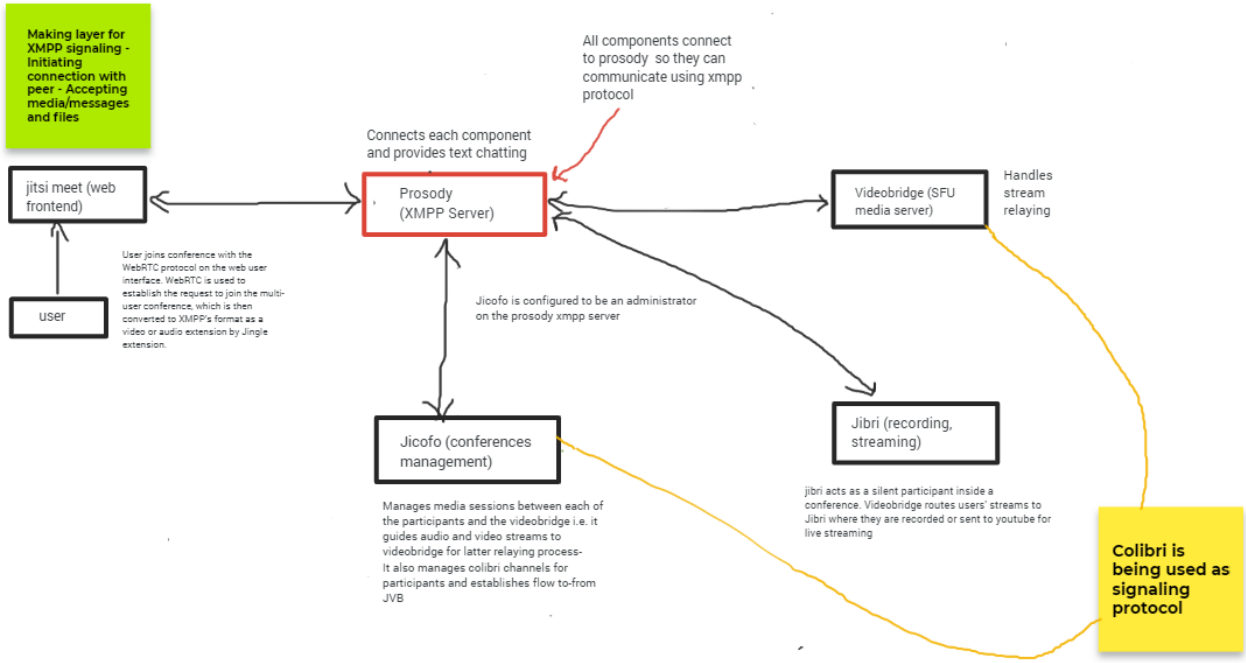


Figure 6-4: Our collaborative efforts to understand the Jitsi architecture.

7. Critical assessment of project and processes by group

7.1 Challenge / Risk analysis

Description of Risk	Impact of Risk	Risk Response	Risk Level
Unable to obtain control over video and audio stream via Jitsi-Meet codebase.	May result in inability to satisfy some user stories.	Reach out to Jitsi Developers for advice on this. If we are unable to find the solution then we can resort to more CPU intensive work arounds. For example, we have considered making use of invisible participants and using a Jibri like mechanism to livestream the calls to our page.	Medium
We are using Jitsi Meet code as the basis of our server side code base. A key risk here is that many files have been minified.	This makes it very difficult for us to understand and manipulate the code within these minified files and ultimately, reduces the control we have over the code base.	For minified files, we need to experiment with the following strategies: 1. turn off the minification, 2. compile the code (there will exist a compile tool somewhere that is creating the minified outputs), 3. sometimes a developer function is included so it will always call the raw files rather than the minified versions. Depending on how Jitsi is setup, this could be a server flag when one starts it up, or something in a config file, or a special path or argument we can use.	High
There is a risk that the CPU usage will increase dramatically when increasing the number of participants using the platform. This is because the more participants that are in a meeting at any one time, the more videos, which already consume large amounts of CPU, must be processed and displayed by the CPU. High CPU usage will also have a negative impact on the battery life of mobile devices.	May reduce the user experience significantly, and may reduce the usability of our platform for those with lower CPU capacity machines.	Our platform currently has the functionality that enables users to auto select their video quality to reduce CPU consumption. Further, we will ensure to conduct extensive load testing, soak testing, stress testing, spike testing and scalability testing before deploying our final product.	Medium
There is a risk that CPU usage will increase drastically when adding visualisation of other meetings from within our own meeting.	This will dramatically reduce the usability and scalability of our platform.	We will ensure the streams of other meetings are of low resolution and have a slow frame rate to reduce the CPU usage required to handle them. Further, we will ensure we conduct extensive load testing, soak testing, stress testing, spike testing and scalability testing before deploying our final product.	Medium

Description of Risk	Impact of Risk	Risk Response	Risk Level
Potential inability to visualise who is in other zoom meetings as the thumbnail preview is too small	This would lead to us not significantly achieving a key part of of the project scope. A key user story is that users should be able to visualise who else is in the other meetings they are not currently in. If the images of their faces are too small to recognise them, this user story will not be achieved.	If this is the case, we can use a facial detection package such as face-api.js, to enable us to crop these individual streams to just the user's face, so that they are more easily recognised.	Medium
Risk of there not being enough server bandwidth available to satisfy the platform requirements.	This may inflate the cost of the project and potentially cause budget overruns.	To prevent this occurring, we will analyse the budget, research the bandwidth requirements for our system and discuss with our client the potential to upgrade to a server with more bandwidth if necessary.	Low
Different RAM availabilities across different computers may reduce the quality of the user experience.	As the RAM availability can differ between machines, different users may experience different video qualities as a result.	To combat this, we will produce a minimum system requirement as a guide.	Medium
Security is a concern, mainly as we are yet to access this part of the code base.	May result in user information leaking out.	This is not a high concern as many people have used the code base without issue. Moreover, we know for a fact that intrusion countermeasures electronics (ICE) are used.	Low
Due to different external and university requirements across team members, work loads are high and team member schedules are busy.	Unless incredibly organised, this can lead to work not being achieved, deadlines being missed or delayed, and reduced quality of work.	To prevent this, we make use of productivity tools such as slack and Jira tickets to manage our time efficiency. More detail on this is discussed in below sections.	High
As this project has many moving parts, effective communication is essential to ensure tasks get done on time. Without this, work can be missed.	The impact of ineffective communication would be that multiple people ending up completing the same task without realising, or that a task gets neglected all together.	Make good use of slack, Jira issues, and frequent commits. Our plan for improving our communication in the second half of the semester is outlined below.	Low

7.2 Limitations in terms of functionality

Our platform still has not achieved the key user story of being able to visualise all concurrent meetings occurring, whilst a user is in a meeting themselves. Our client's vision for how this would appear is to have the main meeting screen at the top of the page, and buttons to switch between 9 other meetings below. Aligned with these buttons would be a low resolution, low frame rate stream of what is occurring within that meeting that the button would take you to. To attempt to achieve this, we have spent a lot of time studying the Jitsi architecture, how its various components fit together and crafting a diagram of our hypotheses for which functions and files are involved in this process. A summary of the Jitsi architecture and its encryption can be found in the Information Search section of this report, and the system diagram we crafted can be found in the System Structure Overview section of the report.

Another functionality we have not achieved is to allow the host to be listed as the host for multiple meetings. The creator of the meeting should be the host for all the meetings and the authority should carry over when the host switches to another meeting. The host should be able to mute everyone in each meeting he or she joined. A proposed solution is to keep track of who created the meeting. It can be achieved by setting up a SQL database and user account name as key. The account name, unlike user nickname, should be kept consistent through one section.

The functionality wherein we keep track of which meeting joined by one user is also not implemented. In our current state, the thumbnails on the right-hand side of the webpage are just static links to the pre-set meeting rooms rather than dynamically changing based on which meeting the user joined. We expect that we can change it based on the user's action. This function should also be implemented by using a SQL database in a similar way as to how we achieve the host functionality. It also provides us with a solution in allowing the user to return to the last meeting joined when leaving one meeting.

The homepage functionality has also not been implemented to an extent that satisfies our requirements. It should display all active meetings on the server at the time and allow the user to create a new meeting from it. In the current state, we do not have a way to get a list of meetings on the server. We proposed to read jitsi-server log file and retrieve information and actively update it on the webpage.

7.3 Limitations in terms of structure, design, implementation

A key limitation for our project is its inherent reliance on the Jitsi API. This has limited the control we have over the functionality of our platform, and has meant we are restricted in our ability to alter and decide upon structure, design and implementation. This has lead us to put a greater focus on the user interface so far in semester, without having access to full support for understanding the backend. There is also usage of webpacks in the code we are using, and although this makes the code easier to deploy and smoother on the browser, it makes it more difficult to identify where code elements are. At present, there does not exist a system diagram for Jitsi publicly available, and as such, we have had to make a lot of assumptions, spend a lot of time on Jitsi forums, craft our own diagrams and workflows for how Jitsi operates, and reach out to Jitsi with questions.

Another limitation with our project implementation, is that a lot of the code we need to manipulate, is hosted on our server. Due to the server which we are using, only one person can directly access the server and hence the code at any one time. This means that it is simply not possible to have more than one pair of programmers working

on server side code at a time. Moreover, it can cause problems when a team member forgets to log out of the server, as it means others cannot access it until he/she logs out.

7.4 Primary strengths

Primary strengths of current product

A key strength of our current product was the decision to switch to using the Jitsi API rather than the Zoom API. This API has enabled us to achieve 9/11 of our user stories, and supports gallery view functionality. Our current platform is able to host multiple meetings at once, and users can switch between these meetings from their current meeting, with only one click. This was even better than our user stories outlined, as the user stories only asked for two or less clicks to be required to change meetings. The platform also allows users to switch between meetings in less than 3 seconds, in line with what was outlined in our scope statement and respective user stories. Furthermore, with 16 people using our platform, it only consumes between 80-110% CPU, rather than the 140-200% CPU usage of our platform found when using the Zoom API. Managing the quality of the video where necessary, and if we are looking to further reduce CPU usage, we can stop the audio from regulating, which has been shown to improve up to 40% CPU capacity.

Another strength of our product is the test plan we have driving it. Before programming, we clearly defined our scope, and used equivalence partitioning testing to define tests for our system. As we are still in the initial project stages, having restarted the implementation of our project in week 5, only a few test cases have been prepared at this point. However, our comprehensive plans to extend these test cases to define the remainder of our required functionality as unit tests can be found in our Appendix in the test plan. This also covers our use of integration testing to ensure smooth interaction between the different units of code, system testing to ensure all components of the system work well together as a whole and acceptance testing to ensure our web conferencing platform is "fit for use" from the user's perspective. It also covers the usability testing we will employ, a summative process throughout the development phase, used to identify bugs and errors, and a formative process prior to introducing the final product of our platform to our client.

Lastly, a primary strength of our product comes from the way it was designed and programmed. We wanted to support the principles of SOLID, KISS and YAGNI in our code, enhancing the code's extensibility, readability and maintainability. The Single Responsibility principle of SOLID was clearly practiced in our codebase as each file in our codebase has only a single responsibility in our system. For example, the general.css file maintains all the css definitions for our meeting page, separating the concerns of functionality from presentation. We also closely adhere to the You Aren't Going to Need It (YAGNI) principle, an extreme programming principle wherein we don't build anything before we actually need it (Fowler, 2015). This is based on the idea that in XP programming, the scope can change easily and to be presumptively creating functionality before you actually need it can be a drain on resources. It can easily lead to you analysing, programming, and testing a feature you end up not needing due to scope changes. To keep our code readable and maintainable by all group members, we employed the Keep It Simple, Stupid (KISS) mantra (Bjornard, 2020). This prevented needless complexity in our code.

Primary strengths of group members

As a group, we have found that we have quite diverse, yet complementary, skill and knowledge bases. Moreover, we have each taken on an expert role which we have researched extensively and made sure we are able to assist other members. Linze has strong proficiencies in information systems and server side web programming, meaning that he has been instrumental in setting up, and maintaining, our server for this project. He is always willing to field our questions about the backend of the project, and help us build up our own understanding of the backend architecture. For his expert role, Linze has been Bitbucket expert. In this position, he demonstrated to the group how to create and merge branches. Coming from a background in cloud computing, security and programming, Jamie has been able to provide the group with access to not only her strong programming skills, but also to her in depth knowledge of security, a key consideration for our project. She has filled the role of Git expert and has helped the team throughout by answering any questions or helping with git issues which arise. Holly has majors in computer science and neuroscience, meaning she was able to come into the project with strong skills in software construction and design, but also usability engineering and consumer psychology. This was advantageous as it provided a new perspective on how to design our interface, and how to design our code. Throughout the project, Holly has acted as the Slack expert. This has meant that she has actively set up channels, integrated necessary other platforms and ensuring that team members use channels for the right purposes. Jessie also has a background in user experience design, but combines this with strong web application skills (JS, CSS, HTML). Coupled with this, her understanding of databases has made her an extremely valuable team member as she has been able to consult on the end-to-end processes and help us by drawing out in depth system diagrams to enhance the understanding of everyone else in the group also. For the duration of the project, Jessie as been the XP expert. She has provided advice on best XP practices and ensured that team members are well equipped to participate in the XP processes. Lastly, Kimia has good project management skills and a plethora of other skills important in web design such as UX and UI design, and HTML and CSS skills. Kimia has filled the role of Git expert. She has been active in helping team members out with questions they have and any problems they face. We have two members filling this role as we anticipated it would be the most consuming expert role. It has been very beneficial as it means there is almost always a Git expert on hand when issues arise.

Primary strengths of the group as a whole

As a whole, our group has been working well together; something which upon reflection, we have attributed to three main things.

1. Mutual respect amongst team members. This mutual respect stems from collaboratively creating a group contract wherein we outlined our common goals and group 'rules.' For example, we pledged to *"[attend] all the meetings and equally [commit] to achieving our goals to a high quality and on time"*. As a result, it has lead to a team who is always willing to help each other learn and understand. We are all actively trying to contribute to the personal and academic development of each group member, and are always happy to field questions from other group members and help them achieve their work.
2. Dedication and perseverance from each group member. This project has often required hours of relentless, and sometimes fruitless, research before any coding could be accomplished. All group members have not only been willing to put in the time and effort required, but have exhibited resilience and not given up despite not always achieving results/finding information they required, after many hours of researching and digging through the code. As we outlined in our group contract that we would all *"aim to create a functional product that creates real value for the client"*, this has encouraged us to all contribute equally and never resort to social loafing.
3. Organisation. As a group, our high work load and frequent deliverables require a heightened level of organisation to prevent us from falling behind on our deadlines. We achieve this organisation by having a minimum of two set group meetings a week (Tuesday and Saturday mornings). Within these meetings, we review our deliverables, decide upon action items for the next couple of days, and delegate these action items to different team members. We are all aware of each others' timetables and are good at working around these differing schedules, making the most of everyones' availabilities. We ensure that everyone is constantly up-to-date with the project progress by ensuring that after completing work on the project, we post what we have done on the #progress channel on slack and marked off any relevant tickets. We also ensure we commit our changes regularly to the bitbucket repository, so that everyone always has access to the most current version of the codebase.

7.5 Programming practices

Reflections on Extreme Programming

In the initial couple of weeks of semester, we struggled to apply practices of Pair Programming as all our group members were situated remotely and we did not have a chance to meet up in person due to vast geographical distances between us. As a result, we had to learn new ways to practice pair programming remote. One such source that we found helpful was Atlassian's article about remote pair programming. In this blog article, Prakapchuk(2020), a Senior Developer on the Jira team, shared tools and techniques that can help make pair programming successful.

We found pair programming to be helpful as it allowed us to share our ideas and differing knowledge whilst programming, and to be able to review code as it is written. One model that the article suggested using was the Driver/Navigator model. As per this model, we would have one team member implementing the code. This is the driver. The Navigator, on the other hand, would be reviewing the code in real time. They would then swap these roles after around 20-30 minutes. Majority of the time, we used the screen sharing function of zoom to achieve this. Also with Zoom, we utilised the push/pull functionality of Git on Bitbucket to be able to share the code we were working on and swap roles of Driver and Navigator. This not only helped us to collaborate on the code we were writing, and to review it in real-time, but it also helped us to transfer skills and share knowledge amongst team members. This was particularly advantageous as, as alluded to previously, each member of our team came to the project with quite different backgrounds and technical skillsets.

Despite the positives that this offered, at times we found it difficult to follow along this way as Zoom connections could fail or be delayed, and audio could cut in and out. Another method we are considering trying in the second half of the semester is Visual Studio Live Share. This enables group members to work in parallel and was recommended by Prakapchuk.

Beyond pair programming, we also practised pair testing. This involved having one team member to be in control of machine and another team member to take notes, discuss different test scenarios and ask questions in regards to test cases. This enabled us to increase the quality of the testing process as different aspects of system could be seen from a different perspectives by the team member reviewing the tests. Again, it also brought the developer and tester together to share knowledge and improve the quality of work done by effectively conducting root cause analysis and linking the problems to their cause. This process was instrumental in making sure we adhered to the test-driven mantra of XP development. This mantra outlines that the requirements for a project should be transformed into test cases before programming is commenced on a project, and passing these test case should drive the programming of the project.

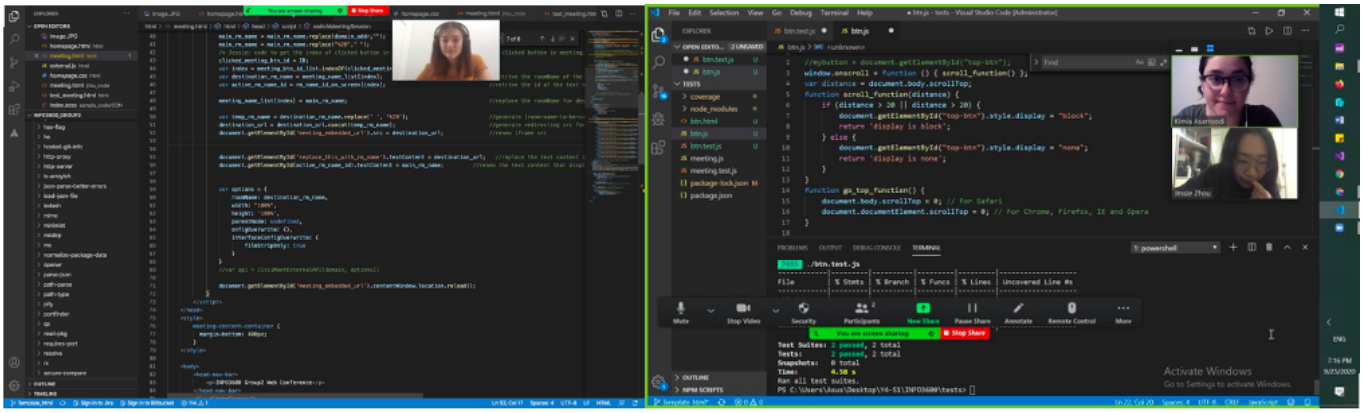


Figure 7-1: Holly and Jamie practising pair programming (left) and Kimia and Jessie practising pair testing (right).

Ways of applying version control, issue tracking, coding styles

The setup of tooling for development

For consistency, we decided that it would be preferable to use the same tools across group members. This meant it was easier to follow along within pair programming sessions, and created a common language across group members for us to discuss the project in. It also meant that, as we began to gain greater proficiency with the tools we were exploiting, we could share insights and shortcuts with the other team members when we came upon them.

As our IDE, we decided upon using Visual Studio Code because of its free nature, cross-OS compatibility and pleasing interface and shortcuts. It is an interactive debugger, allowing us to step through the source code and trace function calls. We used Bitbucket for our version control and SourceTree to easily integrate with Bitbucket. Our evaluation on our use of these is discussed below.

Lastly, we use Termius to maintain our server-side code. Termius has been fairly easy to set up and use, making server side fairly easy to update and maintain. On the other hand, due to limited resources, we can only have one person active on the server. However, this is not an issue with Termius but the server.

Use of Bitbucket with Git and Sourcetree for version control

We used Git for our version control, and Bitbucket to support this version control. As this was the first time many of our team members had used git version control through bitbucket, we kept a introductory summary on how to use Git and Bitbucket on our wiki page throughout the course of the project. Every time a new feature was attempted, a team member would spawn as new branch, and sometimes more than one team member would approach the same new feature in different branch so that they could experiment on this branch with no negative consequence to the other code branches.

Whilst our use of Bitbucket with Git and Sourcetree enabled us to experiment with these new languages, new code bases, and new APIs with relative freedom and minor concern about impacting someone else's work, we have learnt some valuable lessons to-date about how to use these tools and how we can improve upon our use of them. Some key observations about what we plan to do to improve in the remaining weeks are as follows:

1. We often have many branches open simultaneously. At the time of writing this report, we have 12 open branches. Part of this is due to us having begun our system using the Zoom API, and then starting again from scratch, but this is also due to group members being hesitant to impact other peoples' code, and still familiarising themselves with the code base. They feel safer creating a new branch. However, this ultimately leads to a proliferation of branches, many of which end up being

discarded. What is more, the branches do not always have names that are clearly indicative of what they represent. This can confuse other group members and leads to siloed working on the code; that is, less collaboration on the different branches. In the second half of semester, we will endeavour to close branches that are no longer relevant, and use more consistent naming schemes for our branches. We have looked into the following sources to try learn about how we can adhere to common naming conventions for our branches across group members: Git branch naming conventions (Saurav, 2020) and Branching (Helmes, n.d.).

2. Furthermore, due to each group member's hesitation to impact the code, the master branch was not added to very often. We will endeavour, in the second half of the semester, to merge to the master branch more regularly when new features/user stories have been achieved. This will, in turn, help to reduce the number of open branches at any one time.

3. To improve our code review processes, we have also discussed the benefits of using pull requests to merge to master. All members did review the code on the pushes to master we have so far, however it would be better to do this using pull requests for transparency.

Use of Jira for issue tracking

In all, we used Jira successfully to keep track of the work that needed to be completed. After each team meeting, we would create tickets for the action items we had decided upon in the meeting. If possible, we would assign a team member to get this action item done. This was usually decided based on the times people have available, when tasks needed to be completed by, and the order in which they needed to be completed based on the dependencies between them. We also tried to take into consideration the individual skills of members in the team when dividing tasks.

One confusion we did experience surrounded us changing platforms in week 4 of the semester. Changing platforms meant that we need to open new tickets for each of the user stories that we had already completed for the Zoom API. This naturally lead to confusion surrounding which ticket related to which platform. As a result, Kimia set up a clear labelling system, wherein tickets were labelled as either Jitsi or Zoom, depending on which API they pertained.

Another issue we experienced was in assigning the tasks. As we made frequent use of pair programming, user stories and other tickets were often allocated in our team meetings to two people. Jira does not allow you to assign multiple group members to the same ticket. To overcome this, we would try to add the responsible party in the ticket description when multiple people were responsible for that ticket.

Lastly, upon reflection, we realise that we can still improve the way we write, and keep track of our issues on Jira. At times, the 'to-do' item outlined by the ticket was not clear, whether due to inadequate grammar or limited information being given. To overcome this, we will aim to add clear descriptions to each ticket and where possible, outline when each is due. Ultimately, we need to improve on the SMART quality of these ticket items as it is not always evident when one has completed the task at hand. For example, the task of Checking out use of Sandboxie and its alternatives which can be used for both windows and Mac, was not written in a SMART format and was not bitesized. A more measurable and bitesized way of defining this task could have been to break it into the following two tasks: "Download Sandboxie and get it function on mac or Windows computer" and "Create a 1-page summary outlining the purpose of Sandboxie, its limitations and benefits, and the 3 main competitor softwares". Beyond this, to ensure we are adhering to the SMART goal guidelines, we need to ensure the issues we are creating are time bound, and these deadlines are recorded on the issue itself (Boogaard, 2019). Furthermore, there were a couple of occurrences wherein people forgot to create tickets for every key action item decided upon in our meetings, and when people forgot to mark them 'done' immediately after they had been achieved. This lead to multiple people working on the same functionality in parallel, and redundancy in our work.

Use of slack as a communication tool

Slack has been a vital tool in our group communication and collaboration. We have organised our slack workplace in a way that is conducive to transparency of work, ease of finding resources, and ease of sharing and notifying the group. As such, we created the channels #progress, #meetings and #resources, on top of the pre-existing #general and #project channels.

In the **#progress** channel, we post informative yet concise updates on the progress we have made after a session of working on the project. This was created in week 3 to improve our communication, and increase the transparency of workload amongst team members.

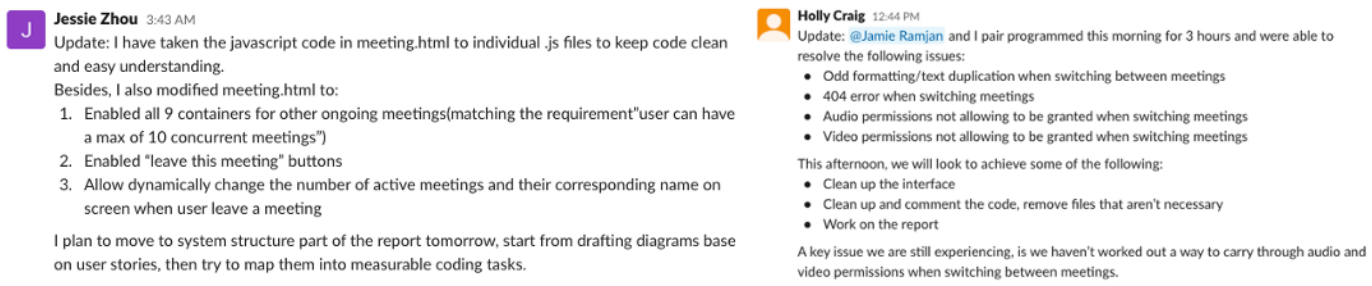


Figure 7-2: sample messages from Jessie(left) and holly(right) sent to the Progress channel, showing which kind of information we need to update the other members

In the **#meetings** channel, we keep track of any meetings we are planning to have outside of our set meeting times. This enables all group members to keep track of the meeting schedule, a necessity as we often meeting between 4 to 8 times a week.

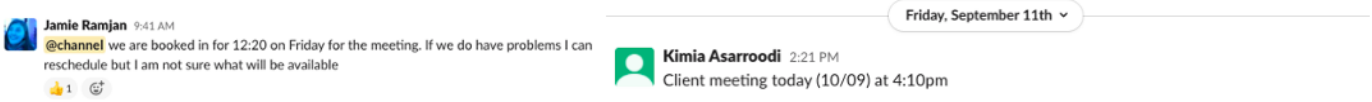


Figure 7-3: sample messages from Jessie(left) and holly(right) sent to the Meeting channel, showing how we notice the group members of the meeting details

In the **#resources** channel, we maintain a bank of important resources that group members have come across. Examples of this include codebases that might help us, software installation guides, and API documentation. This is incredibly helpful as it means we have a central bank of resources that can be referred back to and that don't get drowned out by other messages.

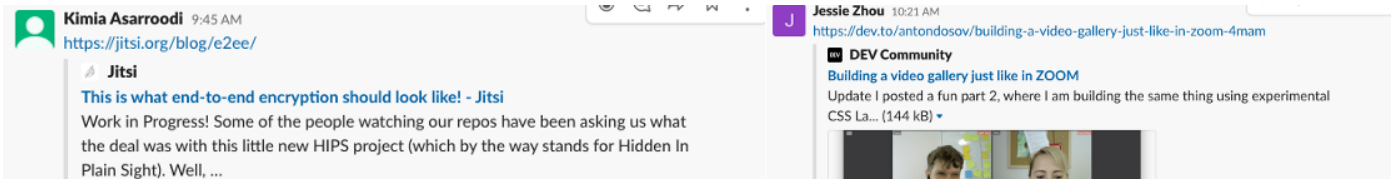


Figure 7-4: sample messages from Kimia(left) and Jessie(right) sent to the Resource channel, showing how we exchange our resources and knowledges informally

We used the **#project** channel for keeping track of other project necessities such as documents that we needed to share but didn't have on Wiki. For example, sharing the project statement and scope approval from the client. Generally speaking, this channel has been less important in terms of communications.

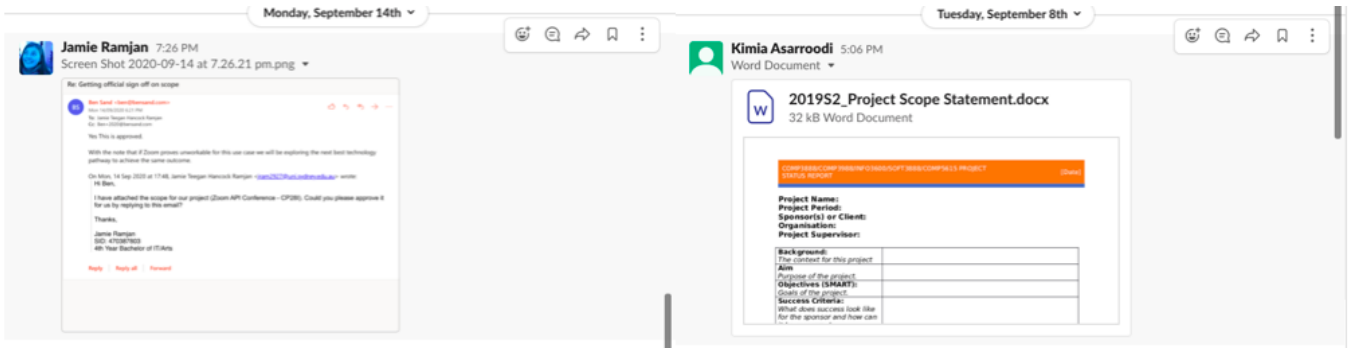


Figure 7-5: sample messages from Jamie(left) and Kimia(right) sent to the General channel, showing how we track on important documents

Lastly, we make use of the **#general** channel to notify group members of anything outside of these areas, for example links to wiki pages we are working on and requests for help with certain items.

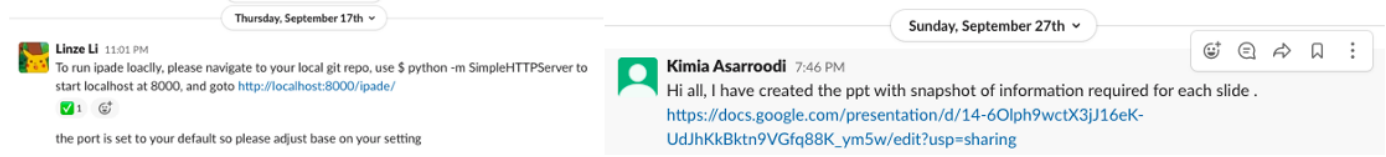


Figure 7-6: sample messages from Linze(left) and Kimia(right) sent to the General channel, showing how we quickly updating each other on uncategorizable but necessary information

Whilst using slack, we also ensure we make sparing use of** @channel** and **@here** tags. As each group member checks all new posts on slack daily regardless, we save notifications for when we have urgent items for the group to be aware of. We have also linked our slack channel to our bitbucket account, meaning the we get notified when someone commits to master. This again improves our communications and transparency of work load.

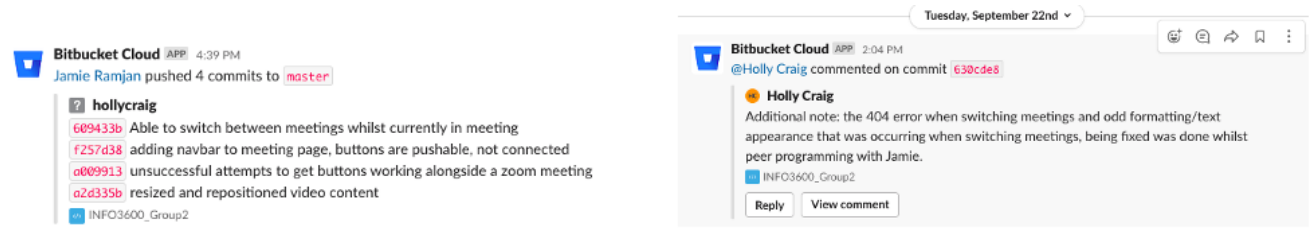


Figure 7-7: sample messages generated by Bitbucket when team members making changes to the master branch of our Bitbucket repository

One limitation to slack is that, when you want to get an instantaneous response from a group member, this is not always possible with slack. This is because, group members only have the slack application installed on their computers and not on their phones. This means that, when group members are away from their computers, they are not accessible. At times like these, we are forced to resort back to facebook messenger. In response to this, we are all planning on downloading the slack application on our smart phones so that we can make use of it as the primary method of urgent communications for the second half of the project.

Coding styles

In this project, we aimed to adhere to the Google JavaScript coding style guide (n.d.) and the Google HTML/CSS coding style guide (n.d.) to keep our coding styles consistent across team members. By having a common style guide, we were able to keep our code practices consistent across team members. For example, we made sure to write code that practises separation of concerns, separating structure from presentation from behaviour e.g. we had a general.css class for all our css code). There is still room for stricter adherence to this style guide. This will become easier as each team member becomes more familiar with the coding languages of CSS, HTML and Javascript and more familiar with the style conventions. Once key aspect we could improve on is making use of CSS shorthand properties were possible, as outlined in the google style guide. This would improve our code efficiency and readability. For example, the code we used to define the margins below, could have been simplified into one line.

```
#!CSS

margin: 5px auto 5px auto;
```

```css
.schedule-style-container{
    color:#000000;
    position: inherit;
    padding:5px;
    display: inline-grid;
    background-color:rgb(182, 130, 230);
    width:12vw;
    height:65vh;
    margin-left:auto;
    margin-right:auto;
    margin-top:5px;
    margin-bottom:5px;
}
```

Figure 7-8: current version CSS code for schedule-style-container

This code section was completed in an inconsistent period to meet different UI requirements. Margin-left and margin-right are written to control the content alignment, while the margin-top and margin-bottom are for aesthetic positioning concerns. It is much easier to keep the code separate during the designing refining stage and good for the team to get used to CSS properties quickly. Once we completed the designing phase, we will clean up the code and fit them into a universal coding style.

Group aspects as well as product and processes

Working with the client

We have met with the client two times a week, every week starting from week 2. All group members aim to attend each meeting, and if they cannot due to work or university commitments, review the meeting within 12 hours of it finishing. Each meeting involved creating a short, 1 minute, video update prior to the meeting on what has been achieved between meetings. We then would present this video to our client, discuss our progress and any blockers we were experiencing, and outline what we should achieve by the next meeting.

We have had a different relationship with our client compared to many other teams due to our client choosing to act as a project manager. This has resulted in less written communication with the client than expected, and more frequent meetings. If issues arise between meetings we alert the client immediately via Discord as per the client's preference.

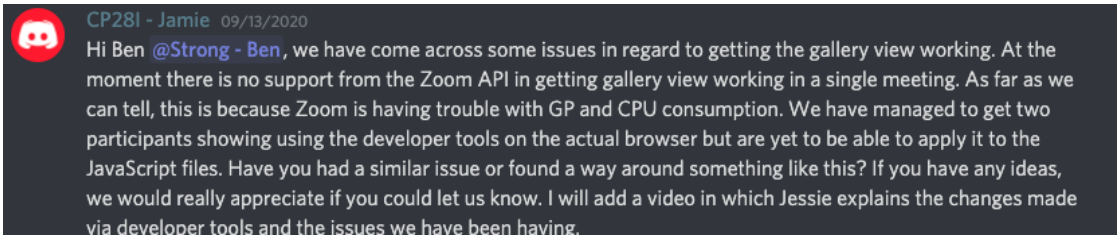


Figure 7-9: Sample message sent to client to keep him up to date to our progress and issue

By having regular meetings with the client and as a team has made development a very open and smooth process. It means that all parties involved are on the same page and has also meant that when team members are struggling, the rest of the team is aware and able to assist.

What has worked well and what needs to improve in terms of teamwork, group processes and XP terms

Whilst it was good for us to rotate XP roles in the beginning weeks of the project as it gave us all a chance to sample the different roles and work out what we were most successful at, now that we have chosen fixed XP roles for the remainder of the semester, we expect to see improved in quality of how the XP role is satisfied, and improved efficiencies. People will be given the opportunity to become more familiar with what is required by the role, and work out how best to fulfil this role. What has been done well, what needs improving and how we will endeavour to work to improve in the second half of semester regarding teamwork, group processes and XP processes, has been covered in the sections above.

8. Division of Work

Below is a table giving a basic outline of major work completed by the team. We have taken each major task, and broken them down into significant parts. It is important to note that most tasks were completed in pairs, and many tasks were completed by pairs in tandem. That is, multiple pairs might were often working on the same section of work. The main reason for this is due to our general lack of experience with the code base and programming languages it uses. Having multiple pairs working on the same task often meant that a solution to the task was discovered more quickly. Moreover, in many cases, one pair would start a section of work, and this would later be passed on to another pair who would either add or complete the work started initially.

Task	Subtask	Holly	Kimia	Jamie	Linze	Jessie	Evidence
Creating platform using Zoom API							
	Install zoom web SDK and create first example to work with	X	X	X	X	X	IG-3
	Find way to link button on web conference platform to meetings			X			IG-6
	Be able to join 4 different meetings	X		X			IG-13

Task	Subtask	Holly	Kimia	Jamie	Linze	Jessie	Evidence
	Functionality of being able to switch between meetings without going back to home page	X		X		X	IG-14, Commit 1724809
	Reformat the style of navigation bar	X					IG-19
	Reformat the homepage and meeting page layout for better UX		X				IG-70
	Resize and organize meeting components					X	Commit a2d335b
	Reformat video containers to allow gallery view					X	Commit c3f69f7
Creating platform using Jitsi							
	Create a MVP working web conference that runs via the web browser	X					IG-34
	Registry the domain				X		IG-28
	Set up server and install jitsi				X		IG-29
	Set up static web hosting for demo				X		IG-28
	Set up a HTML page that we can embed Jitsi meeting into		X			X	IG-31, Commit 280ebc7
	Achieve functionality of being able to switch between meetings from meeting page itself					X	IG-32, Commit a00441a
	Fix bugs: Audio and video not working after switch, 404 error after switch, problematic formatting after switch	X		X			IG-63, IG-62, IG-61, IG-60
	Clean up interface to achieve consistent fonts, colour scheme, and aligned buttons	X					
	Make video feed size dynamic	X					IG-49
	Remove the need for scrolling when switching page					X	IG-57, Commit 041eda8
	Enable Dynamic changes when leave concurrent meeting					X	IG-57, Commit 1278741
Testing							
	Define acceptance and usability tests for user stories	X	X	X	X	X	IG-33, IG-34, IG-32, IG-35, IG-36, IG-37, IG-38, IG-40, IG-44, IG-45
	Preparing test plan for the project		X				IG-74
	Creating test cases for JS using jest		X			X	IG-65
Research							
	Research into Sandboxie		X				IG-9
	Research into gallery view functionality	X	X	X	X	X	IG-21
	Research into alternative APIs	X	X	X	X	X	IG-25, IG-55
	Research into Node.js/NPM			X	X		IG-81
	Research into Jibri		X		X		
	Research into Jitsi video feed system	X	X	X	X	X	
	Prepare a system diagram (for Jitsi team)		X			X	IG-80
Report							
	Create outline of report on Wiki	X					IG-26
	Dot point thoughts into sections of report	X	X	X	X	X	
	Write Introduction	X	X				
	Write Overview of System From User View					X	
	Write Evaluation (including quality of work done)	X	X	X			
	Write System Structure Overview			X	X	X	
	Write Tools Used to Build System				X		
	Write Information Research	X	X				
	Write Group Reflections and Conclusions	X		X			
	Write Individual Contributions	X		X			

Task	Subtask	Holly	Kimia	Jamie	Linze	Jessie	Evidence
	Write Instructions For Running Project Code						
	Write Appendix A - User Stories	X	X	X	X	X	
	Write Appendix B - Research, Studies of Similar Systems	X					
	Write Appendix C - Unit Testing Summary						
	Write Appendix D - Acceptance Testing		X				
	Write Appendix E - Usability Testing		X				
	Write Appendix I - Test Plan		X				
	Edit Report	X	X	X	X	X	
Presentation							
	Create outline for presentation		X				IG-79
	Overview of Project	X					IG-85
	System Specification					X	IG-86
	Demo	X	X	X	X	X	
	Quality of Work		X				IG-84
	Project Plan				X		IG-87
	Group Processes			X			IG-88

9. References

Atkinson, R. (1999). Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International journal of project management*, 17(6), 337-342.

Barnum, C. (2010). *Usability testing essentials ready, set-- test* (1st ed.) . Morgan Kaufmann.

Bdaiwi, Y. (2017, March 24). Stakeholder Analysis using the Power Interest Grid. *ProjectManagement*. Retrieved from https://www.projectmanagement.com/wikis/368897/Stakeholder-Analysis--using-the-Power-Interest-Grid

Bjornard, K. (2020, September). KISS (Keep it Simple, Stupid) - A Design Principle. *INTERACTION DESIGN FOUNDATION*. Retrieved from https://www.interaction-design.org/literature/article/kiss-keep-it-simple-stupid-a-design-principle

Boogaard, K. (2019, January 25). Write achievable goals with the SMART goals framework. *Atlassian*. Retrieved from https://www.atlassian.com/blog/productivity/how-to-write-smart-goals

Bevan, N., Carter, J., & Harker, S. (2015, August). ISO 9241-11 revised: What have we learnt about usability since 1998?. *International Conference on Human-Computer Interaction*, 143-151. doi: 10.1007/978-3-319-20901-2_13

Dosov, A. (2020, June 8). Building a video gallery just like in ZOOM. *Dev*. Retrieved from https://dev.to/antondosov/building-a-video-gallery-just-like-in-zoom-4mam.

Fowler, M. (2015, May 26). Yagni. *martinFowler.com*. Retrieved from https://www.martinfowler.com/bliki/Yagni.html

Gartner CFO Survey Reveals 74% Intend to Shift Some Employees to Remote Work Permanently.(2020, April 3). *Gartner*. Retrieved from https://www.gartner.com/en/newsroom/press-releases/2020-04-03-gartner-cfo-surey-reveals-74-percent-of-organizations-to-shift-some-employees-to-remote-work-permanently2?source=BLD-200123&utm_medium=social&utm_source=bambu&utm_campaign=SM_GB_YOY_GTR_SOC_BU1_SM-BA-PR

Google HTML/CSS Style Guide. (n.d.). *GitHub*. Retrieved from https://google.github.io/styleguide/jsguide.html

Google JavaScript Style Guide. (n.d.). *GitHub*. Retrieved from https://google.github.io/styleguide/jsguide.html

Hambling, B., & Goethem, P. (2013). *User acceptance testing a step-by-step guide* (1st ed.). Swindon, England: BCS.

Helmes, J. (n.d.). Branching. *GitHub Gist*. Retrieved from https://gist.github.com/digitaljhelms/4287848

High CPU by FreeSwitch. (2020, May 9). *BigBlueButton GitHub*. Retrieved from https://github.com/bigbluebutton/bigbluebutton/issues/9472

Notenboom, A. L. (2012). What's the Difference Between a Sandbox and a Virtual Machine?. *ASK LEO!*. Retrieved from https://askleo.com/whats-the-difference-between-a-sandbox-and-a-virtual-machine/

Lakra, N. (2016, May 21), Demystifying JITSI [Blog post]. Retrieved from: https://www.tothenew.com/blog/demystifying-jitsi-2/

Learn how to Live Stream and Record on your Jitsi Meet Install. (2018, June 3). *Jitsi*. Retrieved from https://jitsi.org/blog/learn-how-to-live-stream-and-record-on-your-jitsi-meet-install/.

Luo, M. (2016). Horizontal Scaling of Video Conferencing Applications in Virtualised Environments. *Colorado State University*.

Nikolaev, A. (2014, Jul 25), Broadcasting of a Video Stream from an IP-camera Using WebRTC, *Code Definitely*, Retrieved from: https://www.codeproject.com/Articles/800910/Broadcasting-of-a-Video-Stream-from-an-IP-camera-U

OpenVidu load testing: a systematic study of OpenVidu platform performance. (2018, December). *OpenVidu.* Retrieved from https://medium.com/@openvidu/openvidu-load-testing-a-systematic-study-of-openvidu-platform-performance-b1aa3c475ba9.

Prakapchuk, R. (2020, July 2). Remote pair programming? Oh yes, you can. *Atlassian*. Retrieved from https://www.atlassian.com/blog/technology/remote-pair-programming-tools-process

Saurav, S. (2020, April 19). Git branch naming conventions [Blog post]. Retrieved from https://deepsource.io/blog/git-branch-naming-conventions/

Teresa, K. (2020, June 12). Best Sandboxie Alternatives for Windows, Mac & Linux [Blog post]. Retrieved from https://www.techowns.com/best-sandboxie-alternatives/

Trueman, C. (2020, April 4). Pandemic leads to surge in video conferencing app downloads. *COMPUTERWORLD*. Retrieved from https://www.computerworld.com/article/3535800/pandemic-leads-to-surge-in-video-conferencing-app-downloads.html

Wadhawani, P., & Gankar, S. (2020, May 19). Video Conferencing Market size worth over $50bn by 2026. *Global Market Insights*. Retrieved from https://www.gminsights.com/pressrelease/video-conferencing-market

Real-time communication for the web. (n.d.). *WebRTC*. Retrieved from https://webrtc.org/

Rodriguez, F. (2020, 8 April). Jitsi Meet Encryption. *GitHub*. Retrieved from https://github.com/jitsi/jitsi-meet/wiki/Jitsi-Meet-Encryption

Welson-Rossman, T. (2020, September 28). The Implications Of Remote Working As The Workplace Of The Future. *Forbes*. Retrieved from https://www.forbes.com/sites/traceywelsonrossman/2020/04/28/the-implications-of-remote-working-as-the-workplace-of-the-future/#5df995035ed2

What is the minimum bandwidth required to use BigBlueButton?. (n.d.). *GCC*. Retrieved from https://www.gcc.mass.edu/helpdesk/knowledgebase/what-is-the-minimum-bandwidth-required-to-use-bigbluebutton/#:~:text=We%20recommend%20that%20you%20have,data%20to%20the%20BigBlueButton%20server.