Video conferencing

Jonathan Kasongo

OCR A-level Computer Science NEA

Full name:	Jonathan Kasongo
Candidate number:	N/A
Centre number:	N/A
Centre name:	Harris Academy Purley
Qualification code:	H446

Contents

List of Figures

List of Tables

How to read this document

This document is my coursework component for the OCR A-Level Computer Science course. It is all my own work, and the parts which are not mine have all been referenced accordingly. This document was intended to be read as a PDF so if you have physically printed this document the following 2 points will not apply to you.

- In this work **blue** text indicates a link either to a website or to another section in this PDF.
- In this work **green** text indicates a reference to another piece of work, details on this reference can be found in the bibliography at the end of this PDF.

This was achieved using the LaTeX typesetting language. For more information see `https://www.latex-project.org/`.

Finally in this document we frequently make references to later/previous sections, figures and algorithms using the **blue** in-document hyperlinks. So that the reader does not lose their current page, it is recommended that they make use of the bookmark functionality in their PDF viewer to mark their current page (if available in your PDF viewer).

Thank you for reading, *Jonathan Kasongo*.

Analysis

§1.1 Context

My client Axel Alabi has asked me to create an interactive video conferencing application to allow others to view talks in real time. The current solution is to use the Zoom video conferencing application. While it is true that the application is technically sound and can work fine, there is a large number of elderly users that also try to connect to the conferences. These users often don't fully understand how to correctly use the application and then end up accidentally disturbing the conference/talk [1], by leaving their microphone's on, accidentally raising their hands and having to ask Axel for help with their various technical issues and so on and so forth. This makes my client's job difficult since he is in charge of managing the Zoom call. To combat this situation he would like a simple and user friendly video conferencing application that provides the features needed for people to view and interact with the conferences in easily and comfortably in real time. This includes features like (but not limited to) audience participation, the ability to speak to others via one's microphone and the ability to vote on polls. The application should be created specifically to help elderly people have a better experience whilst watching any conferences, so may also include extra accessibility features to ensure comfortable viewing for all, irrespective of one's age and/or disabilities.

§1.2 Justifying a computational method

The population of elderly ones in the UK has seen a 52% increase in the last 40 years. This group of people includes a large number of those who are isolated and feel a sense of loneliness in their lives. However through video conferencing these ones gain the ability to socialise and interact with others from the comfort of their homes. This is especially useful as many elderly ones have limited mobility or are bed bound. Without regular opportunities to socialise and interact humans become depressed and our mental health will begin to decline. By developing software to enable those disabled people to talk to others can improve the quality of their mental health significantly. When people are limited by their disabilities or by their illnesses the opportunities to go and talk to new people in real life become far and few between. For these people real physical interactions may not be possible, meaning that a computational solution to their problem is not just preferred but necessary.

Accessible video conferencing software is useful to young people as well. People like my client also have problems with current systems. In order for him to carry out his job effectively he requires a simple and reliable computational solution. Current systems pose a challenge for elderly people to use, which in turn means that my client has to spend a good majority of his time helping people set up their webcams, microphones or other settings. If my client instead had access to video conferencing software that was easy for elderly people to understand how to use comfortably his job would be simplified significantly, and in order to create that new software it is evident that a computational method can be used. To justify this claim we examine the average download speeds for UK residentials and compare it to the amount of download bandwidth needed to transfer clear video and audio for 1 second. *"Average download speed of UK home broadband connections was 69.4 Mbit/s"* as of 2023 [11]. To compare this speed we use data from a study that took place at the University of Chicago *"On average, these applications used about 1-2 Mbits/s of download bandwidth"* [5]. Even when taking the worst case of 2 Mbits/s of download bandwidth, only $\approx$ 2.88% of the broadband is used. Moreover we have evidence from the previous systems like Zoom that were in use, that creating such an application is definitely amenable to a computational approach. Applications such as Zoom are able to transfer 1080p quality video and audio data at 30 FPS with millions of users worldwide. These results demonstrate the validity and feasibility of a computational approach to solving this problem.

Decomposition is a computational method that involves breaking down complex problems into multiple smaller and more manageable problems. Real-time video/audio feeds is the ability for a system to take the live video and audio from a user's device share it to others with negligible latency. We justify the feature of having real-time audio and video by examining the client's request as in 1.1. *"My client ... asked me to create an interactive video conferencing platform."* Oxford Languages define a video conference to be *"a conference in which participants in different locations are able to communicate with each other in sound and vision."* Hence in order to satisfy the client's request we

[1] From this point forward we will avoid using "conference/talk", and simply replace it with "conference".

will need to have real-time video/audio feeds. A justification of whether or not real-time video/audio is computationally feasible was given in the previous paragraph. The problem of sharing real-time audio/video feeds between users can be decomposed into numerous sub-problems that are easier to accomplish. For example we could break this problem into 4 sub-problems:

1. Establish a connection to the other user's computers
2. Ensure user has connected a suitable webcam/microphone
3. Access the webcam/microphone using the relevant API's
4. Send the video/audio data to the other users in the conference

Breaking larger problems into multiple simpler problems reduces the complexity of a system and promotes a maintainable system. Rather than trying to debug a large and complex system, it is much easier to debug a single function or class that accomplishes only 1 task. This is because the large and complex system may be throwing errors for a multitude of reasons, perhaps it could also be throwing errors because of one mistake that was written in the code some several hundred lines ago. With smaller and more concise code organised into functions and similar structures, we can tell exactly where the code is throwing an error and start working on resolving the issue immediately. This problem is amenable to a computational approach as we are able to improve the maintainability of our code base by applying the technique of decomposition. Moreover decomposition allows for a much simpler approach to problem solving in programming. When we face a large and complicated task we can first decompose the problem as we did in the example above, and then piece together those smaller solutions to the sub-problems, into 1 solution that achieves our intended goal.

Often once we have decomposed a problem, patterns emerge from the smaller decomposed problems. In recognising these patterns we can reduce the amount of code needed to solve each sub-problem by placing the common operations of each function/class into a function/class of it's own. In order to ensure these repetitive actions aren't found in my code I will employ the use of the DRY software development principle. That is the "**D**on't **r**epeat **y**ourself" principle. The purpose of the DRY principle is to avoid writing redundant code by replacing it with abstractions. This principle guarantees clear and concise code. One feature of the problem that could benefit from pattern recognition is the username and pass code system. A detailed justification for this feature is provided in section 1.5 under the sub-heading *Usernames and passwords to join a call.* There are 2 cases in which we would need to check usernames and passwords, those are when the user initially logs into their account and when the user enters a pass code to enter a video conference. Instead of writing 2 separate pieces of code to check for the correct username and password I could apply pattern recognition and instead implement 1 function called `Check_User_and_pass code()` to be used in both cases. We have seen that this problem is definitely amenable to a computational approach, through the application of pattern recognition to ensure clarity and conciseness in our co debases.

As mentioned in the above paragraph abstraction is another computational technique that is suited to being used in my solution. Abstraction is the process of removing unnecessary details, and only keeping in the parts of the solution that are important. Abstraction could be applied to the design of the UI of my application. The main focus of this application should be for it to be simple and easy to grasp as highlighted by my client in section 1.4.5. So in the design of my UI I will not give the user every single piece of information available about the call because the vast majority of that information will be useless to them. Furthermore it can be argued that the presentation of less pieces of information is better for the user's mental health than the presentation of large amounts of information [3]. When humans have too many options we become unable to make decisions, so by limiting the amount of information available to the user we allow them to focus on the few important configurations that they should be in control of, in turn permitting them to give the majority of their attention to whomever they are having a call with. This appropriately demonstrates the amenability of applying abstraction in our final application. This choice will enhance the user's experience while on our webpage and make the design less cluttered and more aesthetically pleasing, making the application more desirable to use.

The final computational technique is algorithmic thinking. Algorithmic thinking is the process of creating a finite sequence of steps in order to solve a given problem. Another one of the main objectives for the application is for it to be accessible to all. Adding the option to have macros, (programmable keys that are set to perform common actions) can prove to be very convenient for those with limited mobility. If the person cannot use a mouse comfortably, then they will have the ability to program a selected key/button to be able to perform pre-determined tasks like turning video on/off and muting/unmuting the microphone. Algorithmic thinking can be applied when implementing this feature. The task can be broken into a number of finite steps like so:

1. Ask the user what button they would like to use for the macro.

2. Update the relevant class to ensure that the action can now occur via the macro.
3. Save this configuration onto their account through the database.

The approach to use algorithmic thinking is justified innately by the nature of problem-solving itself. Algorithmic thinking is not just convenient when applied in the example of implementing macros, rather it is essential to the implementation of nearly every feature that the finished system will include. Suppose we did not need to apply the technique of algorithmic thinking when programming, in other words we did not require any finite sequence of steps to solve the problem, it means one of two things. Either the problem needed no sequence of steps because it was solved instantly or that the problem can be solved using an infinite sequence of steps. In the first case if the problem needed no steps to solve it, it must have been a trivial problem. This case wouldn't apply to this application because if it were trivial then my client Axel Alabi would easily program the system himself, because of his expertise in computer science. The second case implies that the problem will be solved in an infinite amount of time, however our clients do not have an infinite amount of time to wait for the solution to their problems, ruling out the second case entirely. In showing that solving problems without algorithmic thinking is absurd we hence show that algorithmic thinking must be used when solving problems, sufficiently justifying the use of algorithmic thinking as a computational method.

Explaining why the problem is amenable to a computational approach. We have already seen that the problem is definitely computationally feasible thanks to our research in paragraph 2 of this section. So we now explain why the main component of our problem is amenable to a computational approach. Indeed many API's do already exist to provide a way for developers to implement video conferencing on the web browser, and we examine such API's in depth in section 1.4.6. The existence of these API's allows us to implement real-time video conferencing using efficient, robust and well-tested algorithms. Without them we would have to spend unjustifiable amounts of time studying the necessary security requirements, developing and rigorously testing all of the relevant algorithms ensuring that they are all efficient enough for commercial usage and reading through the specifications on how data should be formatted in order to send it over a network to the other users in the call.

Visualisation can be used to present data in a more understandable format, hence we make use of it in this report too. For example we provide a visualisation of the decomposition of our problem in section 2.1. Indeed by visualising the problem we have at hand we can more easily identify components of the problem that are amenable to abstraction, decomposition and etc. After applying our computational techniques the one main problem splits into many smaller sub-problems each of which we can solve by using an algorithm, demonstrating that our problem is indeed amenable to a computational approach. To provide an explicit example of how we can use a computational approach to solve a sub-problem we refer to figure 2.1. The collection of 4 nodes in the bottom right form a decomposition of the sub-problem: allowing other users to join a call. From our figure we can clearly see that we need a finite sequence of steps in order to solve this problem and so we can use an algorithm to solve this sub-problem, indeed a pseudo-code algorithm is then created in response to this sub-problem using our visualisation as a guide in algorithm listing 6. Finally we reinforce our argument that the problem is amenable to a computational approach by looking at effective existing approaches to this problem. Applications like Skype, Zoom, Whatsapp and many more are all very successful video conferencing applications all built using computational approaches (see section 1.4), if all of these applications have been built to great success then it in turn shows that the problem at hand must be amenable to a computational approach.

§1.3 Stakeholders

Stakeholder: Axel Alabi

Category: Client

Description. Axel Alabi is a 22 year old male, and is currently in charge of managing the video broadcasts for conferences. He also works as a data analyst for a company specialising in analysing geographical data, and has experience working in computer science and related fields.

How will they use the solution? He would use the proposed solution in order to host video conferences. The new system would help the video conferences run smoother as he wouldn't need to worry about spending his time in helping users solve their technical issues, rather he will be able to focus solely on the conference and whether or not the visuals and the audio are clear.

Explain how they will use the solution. Axel would open his web browser and navigate to our system's website. He would log in to his personal account then go to the create call page. After starting a video call he would then receive a code to allow others to be able to join the call. Axel would then share this code with the people who were invited to join the video conference. Finally he would manage the video conference, muting/unmuting, admitting and removing users into the call when necessary.

Why is it appropriate to their needs? The proposed solution would be appropriate for my client as it would provide a smoother and simpler system in order to host video conferences on. Furthermore the new system would ensure that he no longer faces the challenges that were discussed in 1.1, like users leaving their microphone's. This will no longer happen as the new system will have a clear and intentional design and the user won't obliviously have their microphone on, without being notified. This would improve his life greatly as he would be able to now focus solely on ensuring that video and audio is clear during the broadcast. In removing some of Axel's issues that he faces in his job, he should become more content and calmer with his work life, further demonstrating how this solution will be appropriate to his needs.

Stakeholder: People aged ∼50 and over, with limited experience working with technology

Category: Target users/audience

Description. This group of users typically have limited experience working with technology, because they will have spent the majority of their life without modern technologies. This will be my application's target audience, so it is important that the final solution is appropriate to their needs.

How will they use the solution? These users will make use of the proposed solution by joining video conferences that they were invited to and interacting with them in a manner that is convenient for them. They will be able to interact with the video conferences by raising their virtual hand, indicating to the speaker that they wish to talk. The speaker can then give them permission to talk and properly converse with their virtual audience. Moreover the solution will also enrich their experience by providing options for things like macros, enabling them to perform common tasks, like muting/unmuting a microphone with a set button.

Explain how they will use the solution. These users will navigate to our system's webpage and then log in to their account. They would then access the join call page where they would be able to enter in the code that they were given in order to join the video conference they were invited to. Alternatively the user may want to access the settings page instead in order to improve their experience by changing audio, video settings or perhaps by setting up macros.

Why is it appropriate to their needs? The solution will benefit them greatly as they will now no longer have to ask for help from others to resolve issues with their video conferencing application. Rather they will be able to experience a simple, easy to work with and intuitive system for their video conferencing. Moreover the users will now gain the opportunity to improve their social lives as they will be able to chat with their loved ones and friends over their web browsers easily. Not only will the final solution make their lives more convenient but it will also provide the user with the opportunity to improve their mental health through socialising and engaging in conversation with others. [12]

Stakeholder: IT Staff

Category: Support/Maintainers

Description. The IT Staff would be experienced in working with technology because of their qualifications in this field. These people would be expected to have a degree/diploma in computer science, mathematics or another closely related field. They should also be expected to have significant experience in working in various programming languages.

How will they use the solution? This group of users would be expected to be able to update and maintain the system as required. The staff will use the documentation provided along with the application in order to understand what each function and class does along with it's purpose. Furthermore the clear and readable code will enable them to perform any necessary changes with ease, something they could not have done previously with the off the shelf software they had before.

Explain how they will use the solution. After receiving feedback from the client or other users the IT staff would then brainstorm ideas on how they could improve the system. They would then access the code of this system via it's GitHub repository, and open the documentation available on the website itself. Once the IT staff has cloned the system's GitHub repository, they would be able to modify and update the source code as they please, implementing these new ideas that they brainstormed in order to improve this system's suitability in regards to my client and the user base.

Why is it appropriate to their needs? The solution will be appropriate to their needs as they will be able to access the source code of the final application and change the application to be tailored to work well for their specific needs. Additionally users who enquire about the security of their data when using the application will be able to check how the application handles their data themselves. This means that the IT staff will not have to be unsure about answering user's queries, but rather they will be able to read the source code and give an accurate response to the user every time.

¶1.3.1 Current system DFD diagram level 0 - Zoom

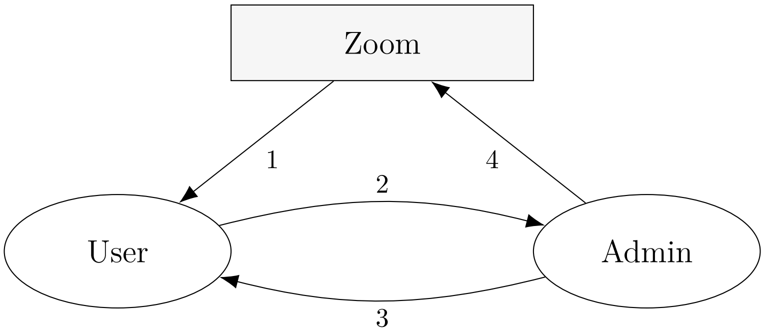


Figure 1.1: Zoom DFD diagram.

An explanation of the DFD is found below. The number closest to each edge refers to the number in the left column of the explanation table below.

Table 1.1: Explanation of DFD.

Edge	Commentary
1	Zoom allows user to join a video conference and gives user live video and audio of all the other participants in the meeting. User has access to a number of configurations and settings after clicking on their profile and then on settings.
2	If user encounters an issue they can contact the admin/host of the meeting for help. User can provide the context or other relevant information about the issue via the chat box.
3	Admin is able to read about the user's issue via the chat box and they can then provide a solution to the user through the the chat box. Admin is also able to invite other users to join the video conference by sending them a link or a code to join.
4	Admin can start or stop the video conference when appropriate. Admin can also interact with the app in order to manage the video conference. They can do things like spotlight speakers, screen share images/video and mute or kick participants.

Justifying the DFD. This level 0 DFD allows me to present a broad overview of the previous system's functionality and how it intended to work. The DFD is crucial to ensure that my client fully understands what the previous system offered to the user, this information will enable me and my client to verify whether or not the proposed system will be able to replace the old system by comparing the system's data flow diagrams and checking if the new system is capable of performing all of the tasks that the old system was able to. Consequently user satisfaction will increase as they are able to quickly adapt to the new system due to it's familiar set of features, i.e. users won't have to complain that they cannot use the proposed system since it doesn't have the features that they require in a video conferencing system, simultaneously improving client satisfaction.

§1.4 Research

¶1.4.1 Questionnaire

I asked a number of questions to gauge the scope of the problem, and to gain ideas on the features I should include. This research method would allow me to get information from a wide range of people. My surveyees included 10 people who would frequently access the video conferences that Axel sets up, and 10 of my peers at my current school.

Video conferencing habits questionnaire.

(1) How often do you video conference with others?
(A) Every day
(B) Two to three times a week
(C) Once or twice a month
(D) Very rarely

(2) What software do you use to video conference?
(A) Whatsapp
(B) Skype
(C) Zoom
(D) Microsoft teams

(3) How would you rate the accessibility in the app you use for video conferencing?
(A) Excellent
(B) Good
(C) Adequate
(D) Poor

(4) How would you rate navigating the UI in the app you use for video conferencing?
(A) Excellent
(B) Good
(C) Adequate
(D) Poor

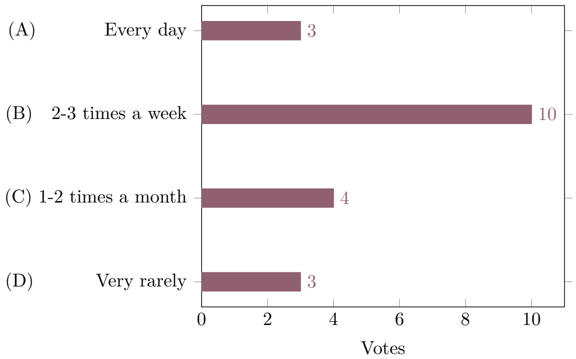
(5) How do you feel about the current level of visual clarity and size of video feeds?
(A) Excellent
(B) Good
(C) Adequate
(D) Poor

(6) How important is closed captioning for your video conferencing needs?
(A) Very important
(B) Important
(C) Neutral
(D) Not Important

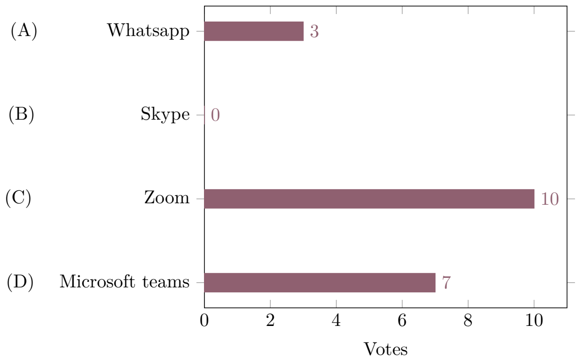
(7) Would you benefit from a text chat function during video conferencing?
(A) Yes
(B) No

Results below with abridged question statements.

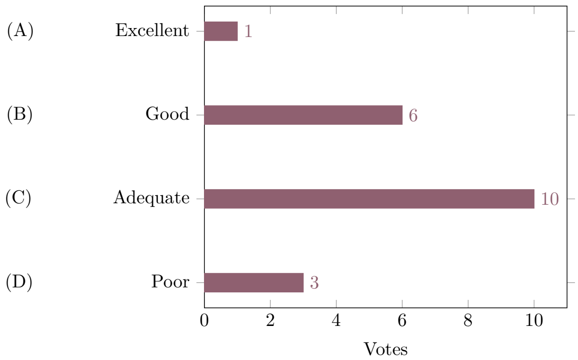
(1) How often do you video conference with others?



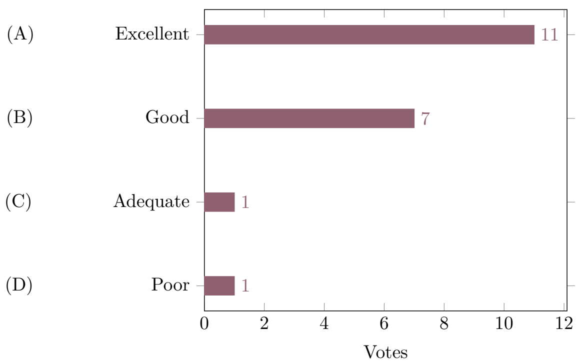
(2) What software do you use to video conference?



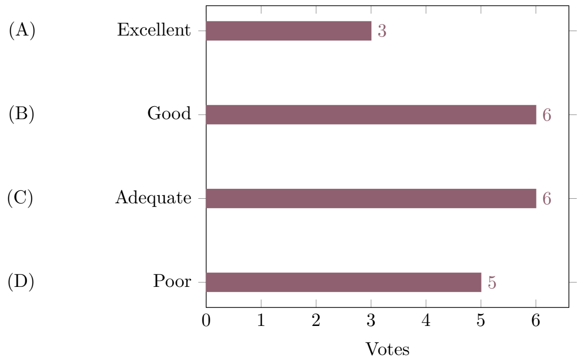
(3) Rate the accessibility in the app you use.



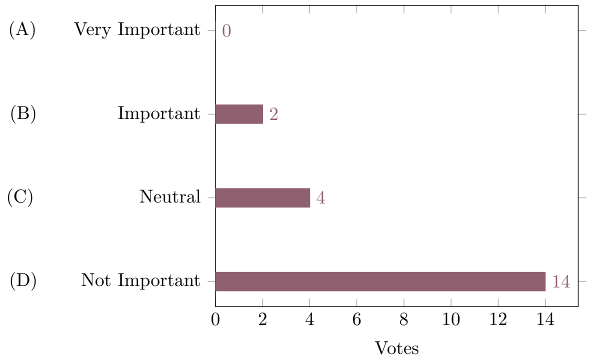
(4) Rate navigating the UI in the app you use.



(5) How do you feel about the level of visual clarity of video feeds?



(6) How important is CC for your video conf. needs?



Consequences.

Question **(1)** shows us that the majority of our surveyees are video conferencing around 2-3 times per week. This indicates that our data, in future questions, may be slightly skewed because this question shows that the bias will be towards people who video conference semi-frequently.

Question **(2)** shows us that Zoom and MS teams are by far the most popular video conferencing applications, so we will analyse each of them to deduce what makes each of them popular.

Question **(3)** demonstrates the potential of our product, the majority of surveyees think that the accessibility features in the software they use is adequate at best. Hence in developing this app we are tapping into a heavily under saturated market with not a lot of competition.

Question **(4)** highlights the excellence of the UI design in the applications that our surveyees are using. If most people find the navigation of their apps excellent, this means that the UI's of these apps were very well designed. We will examine the UI designs of the most popular video conferencing software to learn about what makes their design so strong

In question **(5)** we have a mix of opinions on how people feel about the clarity of the video conferencing apps that they use. One reason for this may be because the quality of the video conference is dependent on a number of external factors, like internet speed and user hardware just to name a few.

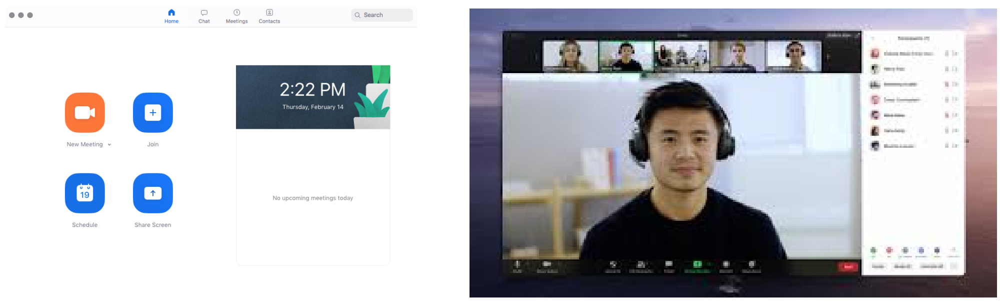
Finally question **(6)** clearly shows us that our users do not feel that having live closed captions will be very important for them. This may be because only a very small number our surveyees have trouble hearing, and again the data may be skewed.

¶1.4.2 Zoom

Zoom is a popular closed-source video chat application, developed by Zoom video communications.

Design

Zoom employs a simplistic approach towards their UI design.



(a) Home page. (b) During a video call.

Figure 1.2: Zoom's UI.

Part (a) of figure 1.2 shows the home page. The left side of the screen shows 4 clickable icons centred in the screen, each with their own functionality. Then the right side of the screen simply displays the current date and time with a simple plant-pot design behind it. Finally the navigation bar at the top has 4 icons showing the other pages that are accessible from the home page, and a search bar has been placed in the top right of the navigation bar.

Positive: This simplistic design seems to be clear and effective allowing users to easily navigate the UI.

Negative: The necessity of a calendar is questionable, additionally the usage of a central split between the 4 icons and the calendar fails to use the golden ratio making the layout displeasing to the eyes.

Part (b) of figure 1.2 shows the screen that the user sees during a video call. The screen is split into a right pane, showing all the attendees names and their respective microphone and video status, and the actual video call which is positioned on the left of the screen. Then on the bottom of the left side of the screen is a small control panel allowing users to mute, unmute, toggle their video feed and more.

Positive: The video call takes up the majority of the screen as it should since it's the most important part of the system, related buttons are grouped together like toggle mute and toggle video.

Negative: The options to change the layout of the video feeds and the speaker we are looking at currently are tucked away in the top right of the screen, some users may have trouble locating these options.

Zoom's UI is clearly designed with clarity in mind. In the design of this project project we draw inspiration from the simplistic approach that Zoom takes to design whilst reorganising the layouts in order to be more accessible and easier on the eyes. Zoom also employs very natural spacing between their buttons, the icons aren't too clumped together neither are they too far apart from each other and I will keep this excellent example of spacing in mind whilst designing the UI.

Accessibility features

From their website [14] they implement 5 major features to ensure that *"Zoom is for everyone"*.

- Live transcriptions
- Automatic closed captioning
- Customisation of font sizes
- Keyboard shortcuts
- Screen reader support

Positive: Some advantages of this software include the fact that there is a good amount of accessibility features able to help a wide variety of people. For instance for those who have limited mobility keyboard shortcuts can be set up to conveniently perform common tasks, whilst those who have trouble hearing well can enable closed captioning during the meeting.

Negative: However there are still a number of limitations with the way these features are implemented. Automatically generated closed captioning is unfortunately only available in English, and may have varying levels of accuracy depending on external factors such as background noise, speaker's clarity and proficiency in spoken English.

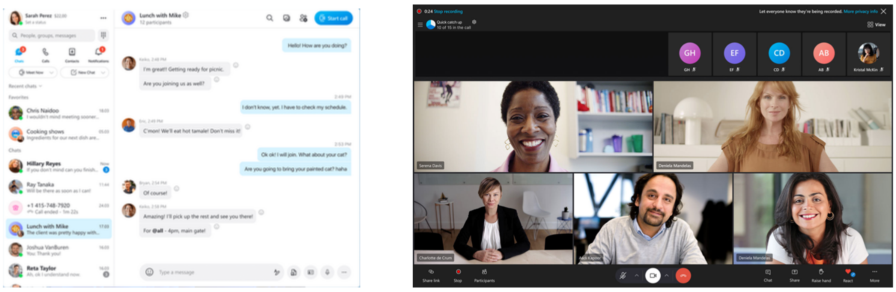
Zoom proposes a solution to the problem of exclusivity when in the context of video conferencing applications. Instead of only designing the application to be usable for one group of people they aim to instead tailor it to cater towards *everyone*. I believe that the wide range of accessibility features is a big advantage of the system, and this could motivate the decision to implement a similar set of features in my application. However while a good number of features is appreciable, it is also necessary to ensure that these features are simple to find and to use for an optimal user experience. This is justified by the fact that my client is specifically requesting an application for those who aren't comfortable with modern technology. To solve this problem I could perhaps implement a built-in tutorial that demonstrates how to use the accessibility features to teach the user how they can use the application to it's fullest potential.

¶1.4.3 Skype

Skype is a proprietary messaging and video chat application developed by Skype technologies a subsidiary of Microsoft [10].

Design

Skype employs an information-centric approach towards their UI design.



(a) Messages. (b) During a video call.

Figure 1.3: Skype's UI.

Part (a) of figure 1.3 shows the messaging component of Skype. The right side of the screen shows the current conversation, whilst the left side shows a thinner pane with information like the user's current status, previews of other ongoing conversations and various buttons that display recent notifications, calls, messages and etc.

Positive: This design allows users to see more information at once also allowing them to navigate the UI quicker since most of the options or information they need is on 1 page.

Negative: This amount of information being layed out so compactly may be overwhelming for the user. It is well known that having too many choices be taxing on our cognitive system, so much so that it may even slow down our decision making. [2]

Part (b) of figure 1.3 shows the video call component of Skype. The UI design is fairly similar to Zoom's design with the two sharing the same bottom control panel. Skype positions it's most important buttons in the centre of the control panel, that is the toggle mute and video buttons as well as the leave call button. The combination of the black and gray colours work to distinguish between the software and the actual video call.

Positive: The most important features are positioned in the middle, so that the user's attention is drawn there.

[2] This information was paraphrased from: https://thedecisionlab.com/biases/choice-overload-bias

Negative: The space for the video conference is encroached upon by the icons at the top showing the other attendees. This makes the video call experience feel cramped.

Skype makes good use of colours to guide the user's eyes. Since red is the complimentary colour of blue, the red bubbles indeed stand out from the ocean of grays and blues found in Skype's main colour scheme and are therefore used to draw the user's attention to an alert or something of similar importance. The black and dark gray compliment each other and allow each component of the UI to be noticeably separate without disturbing the user's attention. I may apply a similar colour scheming approach in my final UI design.

Accessibility features

From Microsoft's support webpage Skype for Windows 8 and above has the 3 following key accessibility features.

- Narrator screen reader
- High contrast colour settings
- Magnifier

Positive: These features have the advantage of being especially accessible for blind people or for those with low vision. The combination of high contrast colours along with a magnifier and/or screen reader ensures that low vision users can still make use of Skype, independently. Skype also doubles up to be an instant messaging platform, permitting users to have all their conversations and other communications in 1 place.

Negative: Whilst the features mentioned are very beneficial for those with low vision, there is no true support for people who are hard of hearing or have low mobility. This set of features is one-dimensional only catering to 1 group of less abled people.

Skype proposes a solution to *"help people with disabilities navigate and control their device as well as get better access to online content."* [3] The solution from Skype allows for disabled ones especially those with reduced vision to benefit from Skype the most, as previously discussed. Whilst Skype offers a good set of features for the disabled, the features that they mention on their website are all already either implemented on most popular operating systems, or can be installed as browser extensions with a few clicks. Therefore I believe that there is insufficient justification to take the time to implement any of the features that Skype implements, and will not be implementing any of those features.

¶1.4.4 Microsoft Teams

Microsoft Teams is a team collaboration application developed by Microsoft as part of their Microsoft 365 family of products, it offers workspace chat, video conferencing and many more features. [4]

Design

Microsoft's UI design provides a similar design to the last 2 examples, with a few changes to the layout.

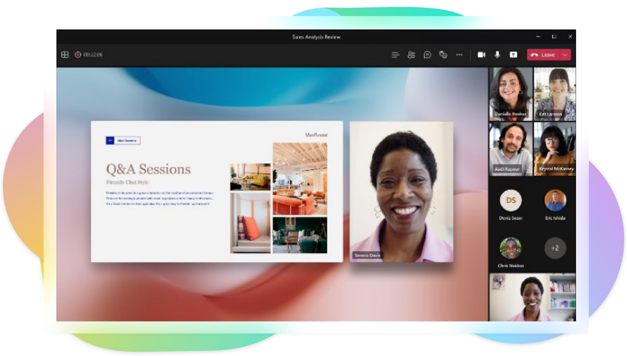


Figure 1.4: MS Teams's UI.

[3] Quote from https://support.microsoft.com/en-gb/skype/what-accessibility-features-are-available-for-skype-89c34c52-f463-437a-b3be-2ea114c5de13

[4] This information was paraphrased from: https://en.wikipedia.org/wiki/Microsoft_Teams

Microsoft team's video call UI has the main video call take up the majority of the screen, along with a list of all the other attendees on a pane along the right. The control panel is positioned at the top as opposed to the bottom of the screen like the 2 other examples. This design aims to maximise the screen economy by dedicating the majority of the screen to the contents of the video call.

Positive: The majority of the screen is used for what it's main purpose should be that is display the contents of the user's current video call.

Negative: The positioning of the control panel can make it hard to find and/or hard to easily access for some users.

In studying the Microsoft teams UI design, I understood that since the main purpose of this software is to allow users to video conference, the video conference itself should take up the majority of the screen. Moreover I will also make sure to avoid the smaller positioning of Microsoft team's control panel, due to the high likelihood that my target audience will have trouble seeing the icons when they are so small. Finally I will take steps to avoid the issue of having too many accessibility features to ensure that our user's do not feel overwhelmed with the amount of choices that they have.

Accessibility features

From Microsoft's support accessibility [5] tools webpage they have a whole host of features meant to ensure that their application is indeed accessible. A few features include:

- Keyboard shortcuts
- Live captions
- Sign language view

Positive: Microsoft provides a wide range of accessibility features designed to accommodate many user's needs. The list of features displayed above only highlights a small portion of the complete list.

Negative: With the large number of features it can become confusing, as well as time-consuming to have to set up all of these accessibility features on the app.

Microsoft Teams aims to provide options that allow people with all types of disabilities to be able to comfortably use their application. In doing so they offer an overwhelming amount of configuration and extra options to the user, some of which may be hard to find for the target audience of our app. The sign language view allows one person in the video conference to be selected as the signer, then for the duration of the video conference they will be pinned to the main screen so that the audience can always see what the signer is interpreting. This idea could potentially be very helpful to our users who live with hearing disabilities, moreover it could also make live captioning redundant. However since not all video conferences will have a dedicated sign language interpreter present, I believe that the feature will be used infrequently enough for us to not consider implementing this feature in our final software.

¶1.4.5 Interview

To obtain a better understanding of the nature of the problem and what features should be implemented in the final solution, I decided to collect some qualitative data through an interview with my client. This data should allow me to have an insight into what the final solution should look like based on my client's requirements and desires respectively.

Interview with Axel Alabi

Date: `29/06/24` **Time:** `3.50pm`

Q: What are some of the limitations of the current system used for video conferencing?
A: It tends to be difficult for people who aren't experienced with technology to properly interact in the conferences. Often times participants will accidentally turn their microphone on or are unable to turn their microphone on when the speaker invites them to.

Q: What are some essential features that should be required in the final application?

[5] Source: https://support.microsoft.com/en-gb/office/accessibility-tools-for-microsoft-teams-2d4009e7-1300-4766-87e8-7a217496c3d5

A: Well to start the app should allow users to see and hear one another in real-time, there should be a focus on simplicity and users should be able to raise their virtual "hand" to interact with the talk.

Q: What are some non-essential features that would be desirable in the final application?
A: The app could perhaps provide a suite of accessibility features to allow disabled ones to have a comfortable viewing experience. This may include closed captioning, volume control and a screen reader.

Q: What operating system should the application be designed for?
A: There is no preference for operating systems.

Q: What are the software requirements?
A: It should be a web-based application. Any suitable mainstream programming language is fine as long as the code is clear enough for me and the other IT staff to understand.

Q: What is your budget for the system?
A: I'd prefer the application to be completely free if at all possible.

Q: What are the security requirements?
A: There should be some form of end to end encryption to ensure that hackers or others cannot access the video feeds. There should also be some kind of username and password system in order to enter a call. Passwords should also be of a good strength e.g. at least 1 symbol, capital and lowercase letters.

Q: How often should database backups be made?
A: They should be made every 24 hours, so that no user data is lost.

Q: How will the new system benefit you?
A: This new system will ensure that all video conferences I am in charge of managing will run much smoother, not only giving me more time to work on other essential tasks but also providing a better viewing experience for all.

From this interview, I can see that my client has some features that should definitely be included in the solution: real-time audio and video as well as the ability to raise and lower one's virtual hand. Moreover I believe that security should also be a serious concern when designing the final solution. Indeed this we should have a requirement to ensure that our user's data remains safe, because we don't want any of our users to have their important data stolen during their video conference.

¶1.4.6 Further research to examine suitable approaches

We now take the time to discuss suitable approaches to implementing real-time video conferencing. In order to implement the real-time video/audio I could make use of the WebRTC API from Google, which would enable me to establish secure peer to peer connections from the browser. Not only does this approach allow us to establish audio and video, but it also ensures that these connections are secure through the use of signalling servers. Signalling servers are servers that manage connections between peers. To understand signalling servers we will go through an example. Consider 2 users, Alice and Bob that want to have a video call. Alice creates an offer for Bob to connect. Consequently WebRTC creates a session description protocol (SDP) object. The SDP object holds information like media types, name of the session and the video codec being used. This data is then saved to a *signalling* server. Bob then reads this SDP offer from the server and WebRTC creates a SDP answer and writes this to the server. Alice and Bob have now established a peer to peer connection. Once Alice and Bob have connected, they still need to establish how they will send each other data. Indeed this is a problem because most device's IP addresses are changed or hidden by firewalls or by network address translation (NAT). In order to combat this issue WebRTC uses the interactivity connectivity establishment (ICE) protocol. ICE allows WebRTC traverse NATs and firewalls by enabling devices to find and the most efficient network path between them, regardless of how the network is configured. [6] Alice and Bob then send each other their ICE candidates, where an ICE candidate is an address of a device that knows how to reach the other user. In essence, signalling allows for users to exchange the metadata of their connection through the WebRTC API. To justify usage of the WebRTC API, I believe it is first necessary to explain why I rejected the idea of implementing an API from scratch. Unfortunately implementations of fully functioning and secure peer to peer API's aren't trivial at all. The time it would take to fully understand how to implement peer to peer connections using the multitude of obscure and highly optimised algorithms like VP9 packe-

[6] This definition is not mine. Source: https://www.digitalsamba.com/blog/ice-and-sdp-in-webrtc

tizers [9] and SHA-256 cryptography for data security would simply be too time-consuming and cannot be justified when fully functioning, tested and performant implementations already exist through API's. The next portion of the justification covers why did I choose the WebRTC API over other API's? WebRTC is from Google and it is well known that Google is a credible technology company. Therefore it is sensible to assume that their API is of highest quality publicly available right now. However, there are alternatives to WebRTC like: VideoSDK, Twilio and MirrorFly. Whilst it can be acknowledged that these API's could potentially be a better fit for my project in terms of performance, features and simplicity, all of these alternatives are paid for. I don't think it would be sensible to pay for commercial API's when there are free one's like WebRTC available that will work just fine. This reasoning is also in harmony with my client's request to have the application *"be completely free if at all possible."* as seen in the interview.

When discussing security it is also important to discuss the username and password management system that may be implemented since it is one of my client's requests. In order to design a system that has secure password management it will be beneficial to study how other industry applications have chosen to solve the problem of managing user passwords. I chose to study Bitwarden, a freemium open source password management service. The following information was found on their architecture webpage [2]. As per their website they use the *"Command and Query Responsibility Segregation (CQRS) pattern"*. The CQRS model developed by Microsoft separates reads and writes into different models. *Commands* are used to write to the database, whilst *queries* are used to read from the database. Each command and query has one *single* responsibility and should be based on actions rather than operations on data. For instance Microsoft give the example rather than use the data-based command: `SET ReservationStatus to Reserved` we should prefer to use the action-based command: `Book_Hotel_Room()` instead. Some of the benefits of this model include:

- Security. It's easier to ensure that only people of authority are performing reads and writes to the database if they're separated.
- Separation of concerns. Splitting the read and write components makes the system more modular and more maintainable.
- Simplified commands. Instead of directly manipulating data commands should be based on the task they try to achieve.

Bitwarden's servers use the MSSQL database and makes automated nightly backups to ensure data is protected. Nightly backups are backups that occur every 24 hours. Moreover the company uses *zero-knowledge encryption* so that the company cannot see it's users data. To ensure zero-knowledge encryption the company uses *"AES-CBC 256 bit encryption for the data encryption and PBKDF2 SHA-256 to derive one's encryption key."* [7] Through this kind of encryption the company is able to have it's users complete trust as they are sure that their data is safe and cannot be read by anyone other than them. Both of these algorithms require large amounts of higher mathematics in order to implement them. Fortunately for our case we will not have to consider implementing these algorithms because WebRTC provides built-in encryption.

For this application's database we need a system that satisfies the following 2 requirements. Firstly the system must be cloud-based, this is because neither me or my client have servers available for use in this application. Secondly the system must be free *"if at all possible"*. It turns out that free database hosting systems do exist provided some conditions, like storage limits are imposed. The following table details the results from some of my independent research on this topic.

Table 1.2: Potential database hosting systems.

Service	Type	Storage	Limitations
Clever Cloud	PostgreSQL	256 MiB	Maximum of 5 database connections.
MongoDB Atlas	MongoDB	512 MB	N/A
Fly.io	PostgreSQL	3 GB	Credit card required, No way to avoid accidental overage fees.
Oracle Cloud	Oracle	20 GB	Payment method required.
IBM Cloudant	Propietary	1 GB	Data deleted after 30 days of no activity.

Fly.io provides "free" database hosting on the premise that you have to give your credit card information. Then once your database uses a certain amount of storage the company automatically charges you based on the amount

[7]Source: https://bitwarden.com/help/what-encryption-is-used/

you used. For this reason I feel that it is justified to rule out the option of using Fly.io completely. If the application were to suddenly grow to have a large user base the database could start charging me or my client whilst we weren't aware, this kind of mistake could be economically catastrophic for either one of us. IBM Cloudant delete data after 30 days of inactivity, meaning that large amounts of user data could be lost. Users should be able to use our application knowing that their data is safe and secure, if we chose to use IBM Cloudant then we wouldn't be able to make this guarantee. If users lose their data it may lead to complaints and frustrations with the design of the new system, leading to poor client satisfaction and users returning to the old videoconferencing system. Hence I feel that it is justified to rule out the use of IBM Cloudant as well. The remaining 3 options all have their relevant advantages and no very significant disadvantages so all 3 of them could potentially be used in this application.

After a careful consideration of all the available options I selected MongoDB Atlas as the database hosting system. We commence by explaining how MongoDB Atlas works and then justify it's usage in this project afterwards. MongoDB Atlas roughly organises it's data in the following manner. MongoDB provides each user with a cluster which they can use to host and store multiple collections. The system stores it's data as BSON documents. BSON is a binary format that stores Javascript object notation (JSON) documents. A document is essentially just a row in a database. Groups of documents are stored as a collection, where a collection can essentially be thought as a table. We can interact with our cluster via MongoDB's NodeJS API. Since the system stores it's data via BSON, when we make queries on our collection it returns the result as a JSON object. The format works very nicely in harmony with the JavaScript programming language, a language that we will make use of extensively seeing as our system is a web-app, because of the close relationship between JavaScript and JSON syntax. I could not justify using Clever cloud's database hosting service, simply because it offered the least amount of storage and there were other options available that offered more. If the database doesn't have enough storage space then it would be a pain to move over all of our data onto another system. Though this is subjective I personally found the Oracle Cloud API documentation a pain to navigate through and understand. They provide so much information on each page that it becomes overwhelming and difficult to find what you're looking for. Moreover they require you to add a payment method, even to use their free tier. Though Oracle can only charge you if you switch to a pay as you go plan, MongoDB remains to be the safer option as they cannot charge you at all without your payment information. In light of my client's remarks in the previous section, it is clear that we should select MongoDB since it ensures that the application remains free of charge.

Another concern pertaining to passwords is our method for checking a potential password's strength. I propose the usage of the `zxcvbn` algorithm [13] developed by Dropbox. This algorithm has been implemented into a number of programming languages by the author Daniel Wheeler, so we will simply import one of his implementations into our code base as a module ensuring that only accurate and tested code is used in the system. To justify usage of the algorithm we should first *briefly* examine exactly what the algorithm is doing. `zxcvbn` consists of 3 main phases: 1) matching, 2) estimation and 3) search. Throughout the rest of this paragraph define p to be some arbitrary password of length n characters. The matching phase takes contiguous sub strings of the input password and classifies each sub string as a specific pattern like a *token*, *date* or a *repeat*. Wheeler provides the following example in his paper; the password `lenovo1111` will be pattern matched like so.

$$\underbrace{\texttt{lenovo}}_{\text{Token}}\;\underbrace{\texttt{1111}}_{\text{Repeat}}$$

Different methods are used to find each class of pattern, for instance in order to match tokens the algorithm first takes a password p it entirely lowercase, it then checks all $n(n+1)/2$ non-empty sub strings with a frequency ranked dictionary, if there are symbols in p it creates extra strings making leet substitutions [8] and does the same check on these. The next phase is estimation where we estimate the number of guesses in order to successfully guess each match in some password. Again for the sake of brevity we will only consider the method to estimate guesses for one class of pattern: *bruteforce* patterns. *Bruteforce* patterns are "patterns" with no obvious pattern and are a random string of characters like `ae#U*$gmi`. The algorithm assumes that each character will take $C := 10$ attempts to guess correctly and subsequently estimates C^n guesses to be required to guess the entire sub string. Finally for the last phase, we define a match-sequence to be a sequence $s_1, s_2, ...$ of matches, such that no matches intersect with one another and the matches contiguously cover the *entirety* of p. The search looks for an optimal match-sequence $s_1, s_2, ...$ of length $|s|$ [9], from a set S of all the match-sequences found in p such that this sequence s has a minimal number of guesses m_G over all matches m in the sequence, where m_G is the estimated number of guesses for an attacker to successfully guess match m. More formally the algorithm computes;

[8] A *Leet* substitution is a symbol that is used to replace characters in words to create a stylised version of that word. For example, `apple` → `@pp1e`.

[9] Take note that we make use of this notation throughout the text, for some arbitrary string s it's length is $|s|$

$$\underset{s \subseteq S}{\arg\min} \left(D^{|s|-1} + |s|! \prod_{m \in s} m_G \right)$$

using a dynamic programming algorithm, where D is a real constant that acts to bias against long sequences that unintentionally minimise the $\prod_{m \in s} m_G$ term. Moreover the module provides other information about a given password in an object `result`. For instance `result.feedback.suggestions` provides a list of suggestions that the user can implement in order to make their password stronger; e.g. *"Add another word or two. Uncommon words are better."* Having now seen how the algorithm works the advantages of using this model are clearly numerous. It employs the use of rigorous and accurate mathematical models in order to estimate the number of attempts that a password can be guessed in and therefore the strength of any given password in *milliseconds*. Additionally there is built-in help for the user as they can consult the warnings and suggestions provided by the `zxcvbn` module to ensure that their password isn't weak.

In September 2016 a fleet of distributed denial of service attacks (DDoS) were launched on multiple company's websites some of which exceeded 600 Gbps making it one of the largest DDoS attacks ever recorded [6]. Moreover one of the affected companies *Dyn* happened to be a major DNS server provider; what ensued were a series of disruptions to some of the world's most prominent companies like Amazon, Twitter and Netflix. This event caused significant economic damage to the companies involved as well as causing harm to these companies public reputations. The source of the attack: The *Mirai* botnet. A botnet is a collection of devices with internet access, that are infected with a malware that enables a hacker or malicious internet user to coordinate and control the actions of the complete collection of devices via the internet. If our web-app is attacked by a botnet there will be severe repercussions for me and my client, so it is important to try and prevent any such occurrence before it actually takes place. If we are to prevent bots from accessing our web-app we have to first deduce whether the user is a human or a bot. This is where *Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA)* comes in. CAPTCHAs aim to determine whether or not a user is human by providing a test that only humans can pass. A classical example is distorted character recognition where, (see figure 1.5)

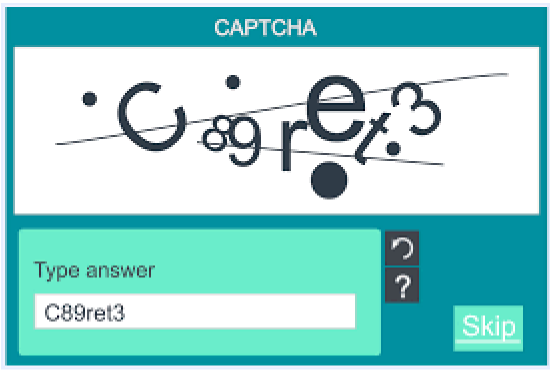


Figure 1.5: A classic CAPTCHA example.

the user is presented with some jumbled and warped text and is asked to type out the characters they see. This task used to be extremely difficult for computer programs to solve since they aren't able truly "read" these twisted characters, rather all the computer sees is a string of `0`'s and `1`'s that represent the given image. Nevertheless, computer scientists have found ways to solve these kinds of tasks with nearly perfect accuracy, through techniques in machine learning. To combat this new technology modern CAPTCHA systems use different tests; one example is the simple checkbox CAPTCHA. This CAPTCHA analyses the movement of the user's cursor as they come to check the box. When humans move their cursor around a screen there is often some unintentional random movement, that is no human can move their cursor in a completely straight line 100% of the time. If the user moves their mouse in a completely straight line towards the check-box then it is most likely that this user is a bot. In order to implement the CAPTCHA I propose the usage of the Altcha API. We now provide a justification for the usage of the Altcha API. Altcha is a completely free and open source CAPTCHA alternative. Moreover Altcha is completely GDPR compliant and is completely self-hosted. There are 2 main competitors in the market of CAPTCHA: Google's reCAPTCHA and Cloudflare's turnstile. The main disadvantage with both of these systems is that they are both cloud hosted, hence both options impose limits on the amounts of CAPTCHA verifications that one can use on their website. Google permits 10,000 CAPTCHA verifications for free per month, whilst Cloudflare offers 10 CAPTCHA widgets at most 10 different website hostnames. Whilst it is true that these free limits are generous my final decision is motivated by my client's words in our interview, *I'd prefer the application to be completely free if at all possible.* Altcha is completely free and poses no limitations or restrictions on it's use because it is self-hosted and we have seen that this API's philosophy aligns itself fully with the requests of my client, justifying it as a suitable choice for this project.

Concluding remarks. What are suitable approaches based on this research? The features that Zoom provides like keyboard shortcuts and custom font sizes would be suitable because they would aid the user and allow them to have a more comfortable tailored viewing experience. The interview also provided some valuable information for essential features in the system, like the focus on simplicity. This information guides the approach to the design of the system, the design should be created to prioritise ease of use and simplicity. Moreover we have seen that using the WebRTC API for peer to peer video connections is a suitable approach because of it's security, credibility and it's alignment with the client's goal to have the system be free if possible. Additionally we also outlined an approach to reading/writing to databases, the CQRS model. Usage of this model will help streamline the database for this system, because each command/query will be clearly named and let the developer know exactly what processes are taking place at that moment. We selected the `zxcvbn` algorithm to verify password strength, because of it's speed, accuracy and it's ability to guide the user into selecting a strong password through the built-in suggestions. Finally we chose the Altcha CAPTCHA system in order to prevent bot attacks on our website because it is fully compliant with GDPR, self-hosted and most importantly completely free.

§1.5 Essential features and their justifications

1 - Real-time audio/video feeds

Explanation: This will enable multiple users to connect to each other via their browsers and view each other's webcams, as well as hear each other in real-time.

Justification: The justification for this feature is that it is explicitly requested for by my client in our interview, furthermore we don't have a *video* conferencing application if we cannot actually see other people's video feeds on our application. Therefore real-time audio and video feeds are necessary for the system to perform it's primary task.

2 - Raising one's virtual hand

Explanation: This feature will allow people to be able to interact with the person giving the talk as if they were present in real life. The virtual hand will be made visible to all participants in the conference so that the speaker/host will be able to ask the audience members to speak when appropriate.

Justification: This feature is sufficiently justified because of my client's specific request to implement it as a feature in the interview. Furthermore the implementation of this feature will help the migration from Zoom to our new platform be familiar as this feature is also found on Zoom. In a lot of talks the speaker will sometimes ask the audience a question or ask the audience to raise their hands in order to engage the audience with the talk. Through the implementation of this feature we enable those who are attending via video conference the opportunity to be able to engage with the speaker even though they aren't physically present.

3 - Designation of a host

Explanation: This feature will allow the creator of the conference to assign 1 or multiple people to be the *host* of the conference. That means that these people will be in charge of managing the conference and will control who to admit into the conference call, who to unmute and who to remove from the call, and will be able to lower other's virtual hand.

Justification: The justification of this feature is that I believe there should always be someone of some sort of authority to coordinate and manage these conferences. This is very common in real life also because without management and coordination people would be clueless, and anarchy would run rampant. This phenomenon has been seen to occur in other video chat applications more specifically, on Zoom (e.g. See Zoombombing), and in order to provide the best experience for our users it is vital that events like this aren't allowed to occur. Through ensuring that there are measures in place to prevent strangers from entering the user's video call, users can be sure that their video call is totally protected and that their privacy is maintained.

4 - Usernames and passwords to join a call

Explanation: Each user will be able to choose their unique username so that users can easily identify one another upon joining a call. Furthermore each user can set their own password so that their account is protected and others cannot pretend to be them. Assigned to each user account will be their saved settings and options, so that if a user has a specific configuration of settings that are adapted for their needs they don't have to set these up each time they log onto a conference. Passwords will be made to fit some set of requirements to ensure that the password is of sufficient strength. When a user creates a call they will be given a unique pass code that they will be able to share with anyone else whom they would like to invite to the video conference.

Justification: The justification of this feature is clearly sufficient because it was explicitly requested during my interview with the client. Moreover without a pass code system to enter a call anyone would be able to enter anyone else's call, giving users no privacy at all. It would also cause issues very similar to the "anarchy" problems that were discussed in the previous paragraph. Through the addition of the username and password feature we offer our users privacy, as well as an easy way for them to identify other participants via their username, improving their video conferencing experience.

5 - Simple and user friendly UI

Explanation: The UI should be designed in such a way that anyone could be able to understand how to use the application. That is the UI should be made to be clear, intuitive and easy to navigate.

Justification: This feature aligns with the idea to have the system be accessible, through ensuring that all users are able to navigate the interface without having to ask others for help. Furthermore a nice and user-friendly interface makes the app more pleasant to use and the user will only have essential information on their screens to make sure that the interface is not cluttered or overcrowded. The justification of this feature is then reinforced by the fact that my client requested this as an "essential" feature during our interview in section 1.4.

6 - Text-chat box

Explanation: Users should be able to communicate with each other via text as well as face to face through the application. On every video conference there will be a text chat box available for every participant to use.

Justification: In the potential situation that a user lacks a microphone or forgets to turn on their microphone other users will still be able to communicate with them through the built-in chat box. This feature falls in harmony with one of the primary motivations for this application, that is, to have the system be accessible to everyone. Even if a user doesn't have the required financial situation to be able to afford a microphone, they are not isolated from others, and can still make use of the system to talk to their colleagues, friends or loved ones.

7 - GitHub repository

Explanation: The system's source code will be hosted in a GitHub repository.

Justification: GitHub will enable users to easily clone this system's repository and all of the source code that was used in order to create it. Moreover once a user clones the repository they will have access to GitHub's host of features, improving the developer experience. For instance each developer wouldn't have to manually track the changes that they made to the source code, rather tracking changes is done automatically by GitHub themselves. Hence the IT staff will be able to save time by ignoring the administrative work usually have to do. They will have more time to focus on the modifications to the source code, enriching the quality of the software, by solving user's issues with the system and creating a better experience for them. Finally I (as the primary developer) won't have to spend time worrying about how to distribute the source code to my client and others, instead it will be freely available via GitHub.

8 - Documentation

Explanation: The documentation of the source code will be provided in the software itself for those who would like to read it.

Justification: Documentation will be provided with the along with the source code of the project, so that my client or the other future maintainers of this project will be able to understand exactly what each function does and how it they are used in the project. The documentation will allow my client or others to be able to make tweaks or updates to the system so that the system becomes better suited for their needs. Additionally in the unlikely event that me and/or my client get into legal issues over this software the open-source nature of the project will be crucial in proving that our software is compliant with the relevant laws and regulations.

§1.6 Limitations

1 - Internet

Explanation: The application will require an internet connection. Since the application makes use of the WebRTC API, we require internet access in order to establish any peer to peer connections.

Justification: Though accessibility is one of the main focuses of the system, this limitation actively makes the app less accessible. This is because people may not always have access to the internet whether it be due to economic, geographical or other reasons. Hence in using the WebRTC API we have limited the number of potential users for our application as not everyone has constant access to the internet. More precisely, only 66% of people have internet access in the world according to Statista. [10] This limitation is sufficiently justified through the fact that we aim to have the app be as accessible as possible, but in order for the application to function an internet connection is required, hence alienating 34% of the world's population as they will be completely unable to use the system.

2 - Passwords

Explanation: As specified in the essential features, each user will have to create an account with a set username and password in order to use the application.

Justification: This means users will have to remember their passwords which could be difficult for this application's target demographic. This means that the app will be potentially inaccessible for people suffering from conditions like amnesia or dementia, further restricting our possible user base. If a user has forgotten their password they may be forced to create a new account, which is a pain for the user as they would have to use a new username and notify all of their contacts that their username has changed. It would also be a pain for the database as each time someone forgets their password a redundant account is left in the database, increasing the size of the database exponentially, if a user forgets their password multiple times. This limitation is justified by the fact that if this limitation is left to be there will be consequences to both the user and also to the database.

3 - Signalling servers

Explanation: In order to use the WebRTC API for peer to peer connections we will have to set up a signalling server as explained in section 1.4.6.

Justification: Signalling servers aren't available free of charge, for example Google Firebase charges nothing for the first 360MB of data transfer per day and after this they charge $0.15 per GB. From my interview with the client I know that Axel would like the system to be *"completely free if at all possible."* Whilst the prices from Firebase aren't extortionate by any means, the reality that my client would prefer the application to be free means that the scalability of the system would be limited. After a certain amount of data is transferred via the signalling server the system would no longer work meaning the number of users we could potentially have is capped. To justify this being a limitation we can notice the fact that we require a signalling server in order to establish a network connection means that one of the client's requests could perhaps, not be reasonably achieved.

4 - Social engineering

Explanation: Users must have access to a computer or some other device that allows them to connect to the internet and access websites. Consequently if the user logs into their account on a public computer they may be susceptible to various forms of social engineering, for example someone may be shoulder surfing the user whilst they are typing in their password.

Justification: These kinds of issues may lead to a breach of data protection regulations and users having their private information stolen. Since we would like to maximise users' privacy we should aim to ensure that no strangers are able to get access to accounts that do not belong to them. Unfortunately when dealing with social engineering it is not trivial for the software developers to completely prevent these kinds of data breaches. To illustrate say that in order to prevent an attacker from viewing a user's password whilst they are typing it, the developer makes the password "invisible" such that it looks like this ••••• whilst they are typing. An attacker knowing this could then install a key logger on the computer that our user is logging onto and will then be able to retrieve the password later.

§1.7 Client requirements

¶1.7.1 Data capture

Client

Requirement(s): PC, Mouse, Keyboard, Video Camera and Microphone.

Justification: My client has his own PC with access to a microphone and video camera and this is indeed what he uses to video conference. Hence data will be captured through these devices as well as through his mouse and keyboard when navigating the webpage. The microphone and video camera will be used to record audio and video data respectively. The mouse and keyboard will allow him to interact with the site and enter information into the site like his username and password.

[10] Source: https://www.statista.com/statistics/325706/global-internet-user-penetration/

Target audience

Requirement(s): A device with access to a standard web browser, along with a microphone and video camera.

Justification: The people in my target audience may not necessarily own a PC and may use devices such as a phone, tablet or something else in order to access video conferences. We should hence design our application with these alternative platforms in mind. The majority of phones and tablets in this modern age have cameras and microphones built-in. So the microphone and video camera will be able to collect the same data as was discussed in the preceding paragraph. The only difference is that users will instead use their fingers in order to navigate the software instead of a mouse and keyboard.

Maintainers

Requirement(s): PC, Mouse, Keyboard, Video Camera, Microphone and GitHub.

Justification: Since this group of people will be working with the source code of this project, it is expected that they will have access to a PC/laptop. The maintainers my want to use the application themselves for testing purposes and to achieve this they will use a microphone and video camera to interact with others during video conferences. In the context of maintaining this project our maintainers will use their mouse and keyboard in order to edit the source code. They may then wish to commit the changes that they make to GitHub, allowing them to automatically track and save the changes made to the source code.

¶1.7.2 Data verification and validation

Client and audience

Requirement(s): Passwords to be of an adequate strength.

Justification: We are dealing with our user's personal data in this project so in order for us to fully comply with GDPR, we should take the initiative to ensure the security of our user's accounts. Since security is an imperative in this software, we cannot allow our users to choose weak passwords for their accounts. By ensuring that users choose an adequately strong password we set up the necessary security of user data. If we take all of the necessary measures to strengthen the security of our users' data then they can be confident that their data is fully private whilst using our application, eliminating one of their potential concerns: privacy.

Maintainers

Requirement(s): Regularly check that the code passes all test cases, create new test cases as the code is added to.

Justification: In the context of updating our system, we would like to avoid deploying a system that is bugged or broken in some way because of a change that a maintainer has pushed. In setting up systems that prevent the update from being deployed if the code does not pass all test cases, we ensure that it isn't possible to accidentally deploy a system riddled with mistakes/errors in it's source code. The verification of test cases can be achieved automatically using GitHub's CI/CD (continuous integration/continuous delivery) pipeline. This decision may save us from catastrophic repercussions in the future.

¶1.7.3 Data processing

Client and audience

Requirement(s): Audio and video compression.

Justification: Audio and video compression is important because without it we would have to send large amounts of data to others over the internet by splitting the data up into a number of packets. Each frame would take a considerable amount of time to arrive completely and the user would be presented with a slow and laggy experience. Furthermore the more data that we have to send per second means that the user will require more bandwidth, and this may not be economically possible for all users. If we wish for our application to be accessible, as per the wish of my client, than we should develop it in such a way that there are minimal requirements to use the application smoothly and video/audio compression will certainly reduce the system and network requirements needed to run this application.

¶1.7.4 Outputs

Client and audience

Requirement(s): Video and audio from other participants are displayed on screen and played out of the user's speakers.

Justification: If a users volume is at 0 (or some other very low value) then we will notify them so that they aren't confused on why they hear no audio. Moreover the user will have access to information like whether their mic is currently muted/unmuted, whether or not their camera is working and if their virtual hand is currently raised. These pieces of information are some of the most important in the context of video conferencing, so it is essential to ensure that the user can clearly access these outputs. Finally in an effort to satisfy the client's request to have *"a focus on simplicity"* the GUI will be designed to present all this information in a clear, simple and professional manner. Justification for a simple GUI was given in 1.5 under *Simple and user friendly UI.*

¶1.7.5 User Interface

Client and audience

Requirement(s): Clear and simple user interface.

Justification: As discussed in the previous paragraph the user interface will be designed so that it is clear and simple to understand. The colour scheme will consist of muted earthy greens, brown and white for accents, giving the application a calm forest-like feel, and providing the users with a visually pleasant and elegant UI to look at when using the system. More specifically the colour scheme is as follows:

- Main `#69CA95`
- Secondary `#284333`
- Tertiary `#9A6442`
- Accent `#FFC6E5`
- Darker Accent `#C49DF2`

¶1.7.6 Security

Client and audience

Requirement(s): Strong passwords, secure data transfer.

Justification: For the video and audio data security will be managed by WebRTC developed by Google, a credible company that will have implemented all the necessary measures to ensure that this data is completely secure during video calls. To keep the database secure the database password won't be written in any of the viewable website files and will instead be read in from separate file that won't be viewable to anyone except authorised users. Security of the database is especially important because personal user information will be stored there, if an attacker gains access to this information it would be a breach of data protection regulations. The user's will be required to set a strong password that meets a set of criteria, they will also receive suggestions on how to improve the strength of their password via the `zxcvbn` module. Additionally, the entirety of the project's source code will be available on GitHub ensuring that there is full transparency between my client and the software's users. By having full access to the source code of this software, users will be able to know exactly how we deal with the security their personal data and look at the approaches we have taken to ensure the security of this system.

¶1.7.7 Backing up data

Client and audience

Requirement(s): Nightly data backups.

Justification: Data backups will be made nightly (every 24 hours) to ensure that none of our user's data is lost, as per the request of my client in 1.4.5. Throughout this project I will always prioritise the requests of my client in reference to the design choices that are made. This improves client satisfaction because my client will feel more in-control of the design process as well as ensure that my client is happy with the software created at the end of this project.

¶1.7.8 Software requirements

Client and audience

Requirement(s): A standard web browser.

Justification: Since this software is a web application, the user will have to access it via a web browser.

Maintainers

Requirement(s): GitHub, any text editor.

Justification: GitHub will host the source code for this project hence, in order for our maintainers to be able to make changes they must be able to access GitHub. Moreover git give us the ability to we can keep track of changes, requests, bug reports all in one place without any hassle so it is important that our maintainers can access it. To actually edit the source code of the software the maintainers will have to have access to a text editing program.

System

Requirement(s): Google Firebase, MongoDB.

Justification: This system will be hosted on Google Firebase a cloud hosted, web-app development platform. As discussed in paragraph 4 of 1.4.6, neither me or my client have access to servers that we can dedicate to this project, hence why the application must be cloud-hosted. Google Firebase allows for web apps to be hosted on the cloud and we will be making use of their free plan which allows us to host our app on the cloud within some limits. This will allow any user with access to a internet browser the ability to use this web app. We will use MongoDB Atlas in order to host our database for this system where information like usernames, passwords and account configurations will be stored. Multiple database hosting systems were discussed in 1.4.6 but I ultimately settled on MongoDB as was justified in that section. It was the safest of the 5 available options, and will hence allow me and my client to be able to use this system without any risk of incurring unwanted hidden costs.

¶1.7.9 Hardware requirements

General

The application will be accessible on any device that has an internet connection and access to a standard web browser. The *Minimum* column provides the minimum hardware requirements needed for the application to run with acceptable quality, and the *Recommended* column provides the minimum hardware requirements needed for the application to run smoothly with high quality.

Table 1.3: General hardware requirements.

Hardware	Minimum	Recommended
Processor	1 GHz	2 GHz Dual core
Ram	1 GB	5 GB
Download speed	5 Mbit/s	7 Mbit/s
Upload speed	3 Mbit/s	5 Mbit/s

Justification: These requirements are based on the old systems requirements. [11] Since we are creating a similar system to Zoom we can justify the use of similar performance requirements in our application because our application will utilise similar amounts of RAM and CPU time when it is being used for it's main purpose, video conferencing.

Client and audience

My client Axel Alabi also owns his own PC with a video camera and microphone. The minimum and recommended requirements for these 2 peripherals are given below.

Table 1.4: Client hardware requirements.

Hardware	Minimum	Recommended
Video Camera	720p at 24fps	1080p at 30fps
Microphone	50Hz - 12kHz frequency response	50Hz - 15kHz frequency response

Justification: For video quality 720p is the lowest resolution that you can achieve whilst still maintaining acceptable clarity in the video. Below this resolution video starts to become grainy. In the movie industry 24 fps is usually the minimum frame rate utilised for video, because it was the lowest frame rate that our brains could interpret as smooth motion. According to American neuroscientist Dale Purves [7], the human hearing frequency range is about

[11] Source: https://support.zoom.com/hc/en/article?id=zm_kb&sysparm_article=KB0060748

20Hz to 20kHz. However we generally will never need frequencies on the extreme high or low end of this range for a video conferencing application, since the frequency of a human voice ranges from 90Hz to 255Hz.

Target Audience

Our target audience may be using this software on their PC, laptop, tablet or mobile phone. If they are using a laptop/PC then the requirements and justifications are the same as the ones discussed above.

Table 1.5: Target audience device requirements.

Feature	Minimum	Recommended
Screen Dimensions	5 x 2 In	6 x 2.7 In
Resolution	720p	1080p
Camera	12MP	40MP
Speaker	70Hz - 20kHz frequency response	40Hz - 20kHz frequency response

Justification: For this group of users it is likely that they will be using our software on their personal device. The specifications layed out give some of the requirements that a device should have in order for the user to have a stable and smooth video conferencing experience. Because these users may be accessing our software via phones or tablets, their screen dimensions will be lower than that of a laptop/PC. Hence for the minimum dimensions I chose a size based on the dimensions of the Samsung Galaxy S7 since I found that this should be the smallest size in which video conferencing can be done comfortably on the previous system. Again for video quality I found that 720p is the lowest resolution that you can achieve whilst still maintaining acceptable clarity in the video. Below this resolution video starts to become grainy. After testing with a number of different camera resolutions I came to the conclusion that the 12MP sensor should be the minimum camera resolution in order to provide the user with an acceptable quality for their video feed. Finally the speaker's frequency response requirement is again based on the human hearing range of 20Hz to 20kHz. It should be noted that if the user is discussing confidential information in their conference they should consider using headphones as an alternative that meet the same requirements as the ones layed out for speakers.

Maintainers

The maintainers of this software are all expected to have access to better hardware than our other user groups because their job requires the usage of a computer day in and day out.

Table 1.6: Maintainer hardware requirements.

Hardware	Minimum	Recommended
CPU	2GHz Dual core	3.5GHz
RAM	4GB	8GB

Justification: We can justify setting stronger hardware requirements for this group of users because they will have to be able to write and test the new code that they add to the system's back end. In order to properly run the tests on their machine they should have a CPU and RAM that can handle interpreting and running moderate to heavily strenuous JavaScript code without causing any issues. These requirements are satisfied by pretty much all modern desktop computers, and most modern laptops, hence these requirements are not something that should be a problem for most users, and shouldn't be especially our software maintenance team.

¶1.7.10 Summary of requirements

Table 1.7: Summary of requirements.

Requirement	Proposed solution	Alternative solution

Continued on next page

Table 1.7: Summary of requirements. (Continued)

Data capture	A video camera and a microphone that are connected to the device can be used to record video and audio data respectively.	A virtual reality headset system may be used such that the headset provides live video/audio of the person in real life or live video of a persons 3D model interacting with the real world.
Data verification	Users will be prompted to enter their passwords twice. The table in the database will be created such that only data of the relevant correct type may be entered.	Users will enter their phone number or email address and information will be sent to them verifying whether or not they would like to update/set their password. Data will be checked for the correct data type in the code base before it is entered into the database.
Data processing	Video and audio data will be compressed and packetized by the WebRTC API, so that it can be sent to others over the internet quickly.	Develop my own algorithm to compress and format video/audio data into packets.
Outputs	Video data from other participants will be displayed on screen, audio data will be played on the user's speakers. User will be notified if their current volume level is 0 or very low. The GUI will present important information to the user and will be designed with simplicity in mind.	Alternatively, audio data could be played in the user's headphones potentially providing a better sounding experience.
Security	The video/audio data being sent over the internet will be sufficiently encrypted by WebRTC. Only authorised users will have access to the database's password. Users will *have to* choose an adequately strong password, satisfying some set of requirements. This will be implemented with the `zxcvbn` module.	Multi-factor authentication could be implemented so that an attacker would require more than just someone's password in order to access their account.

§1.8 Success criteria

¶1.8.1 Qualitative criteria

1 - The system should be intuitive and easy to grasp

Explanation: The system should feel and be easy to use, such that anyone and everyone will be able to use the app without any learning curve. This criterion should also apply not only to the front end design of the system but also to the relevant back end infrastructure. That is a competent programmer should be able to read the code base and understand roughly what each function or class is doing. Furthermore a logically structured and neat documentation should be produced alongside the system to ensure that any developer will be able to understand what a piece of code does if they are unsure.

Justification: My client explicitly asked for the system to be simple and easy to grasp in 1.4.5. Moreover the target demographic for this application is older people who are lacking in experience with technology, therefore in order to ensure a comfortable experience for these people the system should be designed with their background in mind and should hence be simple for everyone to be able to use. Finally the source code of this project will also be provided to the client, so in the scenario that he would like to make any tweaks or changes he may do so with ease by using the clearly written documentation.

2 - The user's data should be properly secured

Explanation: All user's should be able to make video calls and be able to use the system however they would like whilst knowing that all their personal data is secure and cannot be accessed by anyone but themselves.

Justification: If our users and clients are having their data stolen by hackers or other malicious internet users, not only would this generate massive client dissatisfaction but it may also create a horde of legal issues for all parties involved. The *General Data Protection Regulation (GDPR)* is the current legislation in act as of 2016 [8]. In chapter

4 article 32 it reads *"The processor shall implement appropriate technical and organisational measures to ensure a level of security appropriate to the risk"*. Therefore by ensuring that user data is properly secured we are complying with the GDPR saving myself and the client from data-related legal issues.

3 - The webpage design should be aesthetically pleasing

Explanation: The front end of the system should be professional looking and should be aesthetically pleasing, this means that the app will be more inviting and that users will naturally spend more time on this application. The aesthetics of the system will be rated via a collection of different people through a survey. If $\geq 85\%$ of people give a positive opinion on the design of the application we will consider the system to be "aesthetically pleasing".

Justification: If the application looks and feels nice to use then our clients and users will naturally feel more inclined to use it, a professional design will also promote a sensation of trust, robustness and reliability of the system. We will hence be able to ensure that our user's have a nice experience using this system improving client satisfaction.

4 - Users should be able to create and join video conferences

Explanation: All users should be able to create a video conference and join a video conference that they have been invited to using our system.

Justification: All users should be able to use our system in order to achieve the client's primary request, to create a user friendly video conferencing application, as was discussed in 1.1. For the application to be called a "video conferencer" our users should be able to create and join video conferencing sessions with other users freely. In achieving this criterion we will have successfully developed a video conferencing platform ensuring that our application is functional and that my client's main request is satisfied. We have seen that this criterion is fundamental to the client, hence justifying it's inclusion in this list.

5 - Users should be able to create, login and logout of their own accounts

Explanation: Our users should be able to create their own personal account with a username and password and then login or logout of their account as they wish.

Justification: By allowing users to create their own accounts we allow them to be able to save their own configurations. If users aren't able to log back in to their account then their configuration will be lost forever, and our user account database will have redundant records. Finally by allowing our users to logout of their accounts we allow them to be able to protect their private data from malicious internet users who might use their computer whilst the user is away from their computer.

¶1.8.2 Quantitative criteria

6 - User passwords should be of an adequate strength

Explanation: Passwords should have to satisfy some set of criteria, pertaining to length, number of special characters used and etc.

Justification: This criterion ensures that our user accounts will all be properly secured, consequently we will be following the current regulations in place as detailed in the GDPR. We will also be able to instill confidence in our users by making it significantly harder for malicious internet users to access their accounts, hence minimising the risk of a serious data breach.

7 - The video call latency should not exceed 150 ms

Explanation: The delay between a packet of data being sent the packet being received should not exceed 150 ms provided the minimum hardware and bandwidth requirements in 1.7.9 are met.

Justification: The new system should provide better or equal amounts of latency than the old system. Hence users will not only enjoy a more accessible and comfortable video conferencing experience but they will also be able to video conference whilst knowing that the video/audio they are seeing is occurring in real-time as explicitly requested in 1.4.5. The old system recommended a latency of *"150 ms or less"* [12] so we will seek to match or better this value.

8 - The system should be able to handle video conferences with more than 2 participants

Explanation: The system should be able to maintain acceptable performance whilst there are more than 2 participants in the video conference, whose respective devices all satisfy the minimum hardware and bandwidth requirements outlined in 1.7.9. We will measure the success of this criterion by running a test video conference with 6

[12] Source: https://support.zoom.com/hc/en/article?id=zm_kb&sysparm_article=KB0070504

participants, if average latency is less than 150 ms then we will consider the system to be sufficiently performant as to be able to handle ≥ 2 participants in each video conference.

Justification: My client often has to manage video conferences with more than 2 participants, so the new system should be able to handle an acceptable number of participants in each conference, without any significant performance degradation. In 1.4.5 my client said that the new system should be smoother and provide a better viewing experience for all. In order to achieve this the performance of the system should be optimised so that the application runs smoothly for my client and all other users even if the number of participants in their video call far exceeds 2 people.

9 - The system should be able to store at least 500 user accounts

Explanation: The system database should be able to cope with 500 user accounts, whilst still performing at an acceptable level. That is the speed of database reads and writes, shouldn't be compromised. We will say that read/write speed has been compromised if read/write speed exceeds 500ms.

Justification: The system should be able to handle a good number of user accounts. 500 users will allow us to have a good amount of leverage, concerning the growth of our application. If the application happens to gain large amounts of traction suddenly, we can have confidence that the system will not fail or experience significant performance degradation.

Design

§2.1 Breaking the problem down

We now provide a simple visual decomposition of the problem at hand. We let DB denote our database for typographical reasons.

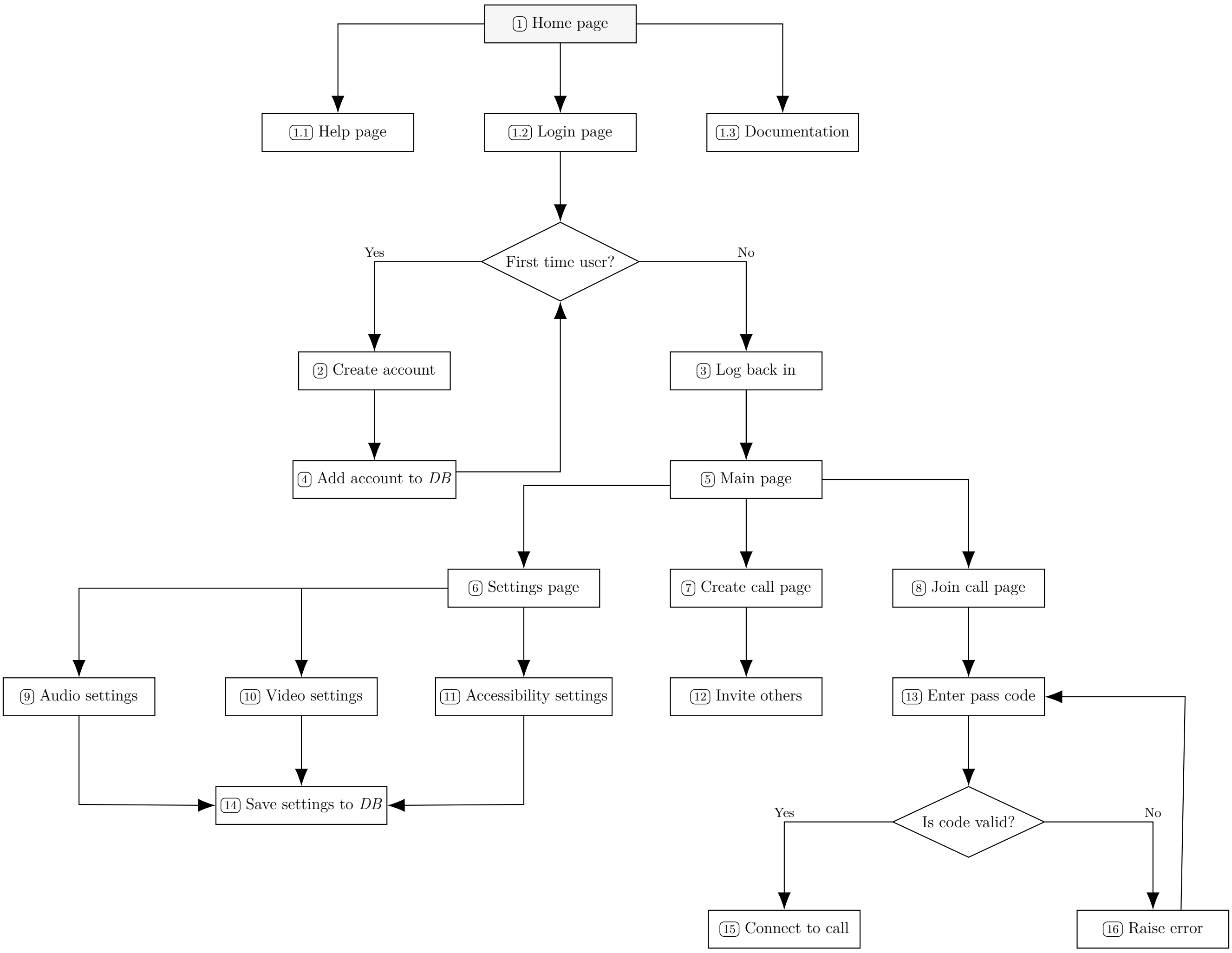


Figure 2.1: Decomposition of the problem.

Explaining and justifying the breakdown

As discussed in 1.2 decomposition can reduce the complexity of a system by providing clear sub-tasks that need to be achieved in order to solve a larger more complicated task. Moreover this method of organising tasks motivates a more modular approach to the implementation of our system; each one of the main sub-tasks is neatly and clearly visualised and the overall presentation shows how each sub-task relates to the others.

Starting from the top of the diagram I chose to display a home page once the user initially accesses our website. The home page will be primarily used to greet the user, show them what the web-app can do and get them to login. However from the home page users will also be able to access the system documentation as well as a help page if users are having issues with using the system. We justify the need for a homepage by highlighting the importance of a first impression. A well-designed homepage can capture the attention of the user and encourage them to explore the rest our web-app. If the homepage is able to provide a good first impression we will be able to garner a larger user base, and simultaneously ensure an excellent user experience as they move around the UI. Moreover if our users are satisfied then our client will be too. Documentation for the system will also be freely available to find on the the home page to be easily accessible for developers. Furthermore the inclusion of the documentation on the home page means that developers aren't forced to create an account just to read the documentation, saving much time for these users. In this manner the system becomes entirely *self-contained*, that is no other external resources would be necessary in order to use, maintain or update the system. Finally in the case that users are experiencing issues with the software a help page will also be clearly available on the home page in order to answer FAQs as well as guide the user through any troubleshooting.

The next pages require the user to first login. Upon entering the login page the user will be asked whether this is their first time using our system and if so they will be prompted to create an account. If it isn't the user's first time on our application then they will enter their username and password and login to their account. The reason that we ask the user to login is because we would like each user to video conference with the settings that are most comfortable for them, once the user logs in we can apply their specific accessibility settings that are tied to their account. Therefore in asking the user to login before they begin video conferencing we are encouraging the user to take full advantage of our software by allowing them to first, adjust the settings to match their needs and requirements.

Once the user has logged in they will have complete access to the entire functionality of our web-app. I decided to separate the main content of the system into 3 pages, 1) Settings page, 2) Create call page, 3) Join call page. I chose to split the application into 3 main pages so that users won't have to search the app through one long overcrowded page in order to find what they are looking for. With this system users will be able to quickly navigate to the page that they need and find what they are looking for easily on that page. Additionally, the concept of splitting our content onto multiple pages is easily scalable, if the site is updated and more content is added the developers can simply add a new page onto the site. Consequently the system can rapidly expand in order to accommodate the growing number of user demands without requiring a full website re-design each time an update is made.

The settings page will be where all the configurations and options for our system can be set/changed. It will be split into 3 main tabs; the audio tab, the video tab and the accessibility tab. Each tab will hold settings related to it's name that is, the audio tab will hold settings related volume and sound and etc. Once a user makes some changes to any of the settings their changes will be saved to their account and this data will remain on the database. The decision to split settings into 3 main tabs not only improves the user experience by allowing users to find the settings they are looking for easily but it's also in harmony with one of my client's main requests for this project; to have a *"focus on simplicity"*. The decision to have settings for our system will help each user to tailor their video conferencing experience to fit to their personal needs. This allows us to create 1 system that is able to accommodate for a large number of people enhancing the accessibility of our platform, a request outlined by my client in 1.4.5.

¶2.1.1 Navigation via speech to text

Suppose that we have a user that is blind or has some form of visual impairment. How would they use the system? Though these people may have limited vision, we often find that their other senses are heightened [1]. The person's sense of touch, smell, hearing or taste may be better in order to make up for the lack of vision. To allow our visually impaired users to be able to navigate the system comfortably we make use of their improved sense of hearing by allowing the user to talk to the system in order to navigate to different pages and perform various actions. In implementing this feature we prevent the alienation of visually impaired users by designing a system that allows them to be able to video conference when they perhaps couldn't have before. This will not only expand our potential user base but it could also help these people to connect with others, by creating more opportunities for social interactions. Moreover we directly achieve one of the client's requests to create an *accessible* video conferencing system.

One approach to the implementation of this feature could be to have a set of phrases that we recognise as commands for the system, once the user says one of these commands our system should recognise this and perform the corresponding action. An alternative may be to use some form of machine learning in order to process the user's speech and use natural language processing to determine the user's intentions. This method is arguably easier for the user, instead of memorise some set phrases for commands they can simply talk normally and the system would recognise what the user would like to do. Although this approach may seem more attractive it is not without it's downsides. It is well known that machine learning is not perfect and that it can make mistakes from time to time.

In this case the system making mistakes and performing the wrong actions for the user could be extremely confusing since the user may believe that they are on a different webpage to the one that they actually are on. Since the model would have to pick up on a few very specific commands in the user's speech it is unlikely that datasets would exist in order for us to use for the training of our machine learning model. Collecting the data on which to train the model on may be very time consuming for me. Considering all of the downsides mentioned it seems reasonable to go with the safer option and use the set phrases approach. In order to combat the potential problem of having the system misunderstand the user's command we could make the user confirm their command before it is executed. The system would read out the command that it has detected and then the user could say "confirm" or "cancel" in order to go through with the action or stop the action respectively. Finally in the case that the user forgets what they have to say in order to perform an action they can say "help" to be reminded of these set phrases.

We also need a method to actually recognise the user's speech. As justified in the previous paragraph this method will need to recognise certain set phrases and identify them correctly. Mozilla Firefox offers a Web speech API that implements features such as speech recognition, speech synthesis, and speech to text. We could make use of their speech recognition to turn their voice into text and then compare the text produced to a each of the set phrases that we chose. Using an API to implement this feature ensures that the speech recognition code is robust and efficient as API's from large and reputable companies are thoroughly tested. The use of an API also improves the maintainability of the software. Since the implementation of the web speech functions has been abstracted away from the developer, the code is simpler, shorter and easier to understand. We are able to directly benefit one of our stakeholders, the IT staff (see section 1.3), through making the code clear and easy to understand.

To briefly summarise our research, we will make use of Mozilla Firefox's Web speech API to enable people with low vision to be able to navigate our system using their voice. In order to navigate the system the user will say one of the set phrases, after which the system will repeat the phrase that it has heard and ask the user to confirm. If the user happens to forget which phrase to say in order to perform a certain action they may say "help" in order to hear a list of all of the set phrases and their actions.

§2.2 Defining the structure of the solution

¶2.2.1 Entity relationship diagrams

Entity relationship diagram — *1st draft*

The acronym "cfg" denotes configuration.

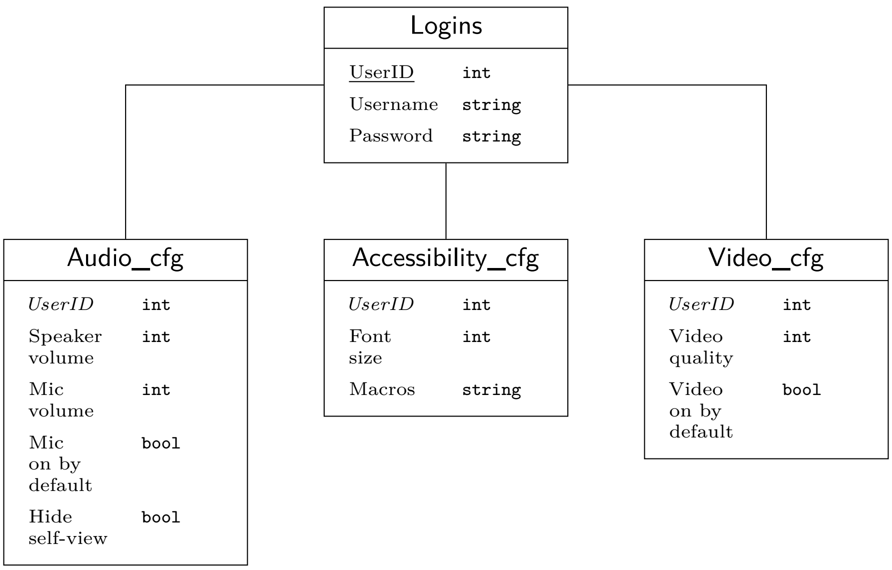


Figure 2.2: The 1st draft ER diagram.

Explaining and justifying the 1st ER diagram draft. This entity relationship (ER) diagram was my first attempt at designing the database for this project. It captures the general structure and idea that I had in mind for how the database of my system should work. Loosely there are 4 main categories of data, login data, audio configurations, video configurations and accessibility configurations, that would have to be stored in our database. Naturally I decided to split these 4 categories into 4 individual tables, such that each table should hold 1 category of data. This should make life simpler once I begin implementation, for example when I have to make a queries about a collection of related pieces of data I will only need to query the 1 table whose label should cover the data I need at that point. I make use of a unique numeric ID in order to identify each user and this is seen as the primary key in the table Logins. These IDs are then used to link each user account to their relevant configurations via the 3 tables ending in _cfg. Though this design choice was ok for a first draft I soon realised that there were obvious improvements that could be made. I present some of the flaws of our current design with an illustration. Suppose that you wish to find *all* the configurations that a specific user has tied to their account.

Algorithm Pseudo code for finding all configurations tied to a user.

```
function GET_ALL_CONFIGS(ID)
    User_ID ← ID
    Configs ← []                                                    ▷ Let Configs be an array.

    Configs.insert( Query_tbl(Audio_cfg, User_ID, Speaker volume) )
    Configs.insert( Query_tbl(Audio_cfg, User_ID, Mic volume) )
    Configs.insert( Query_tbl(Audio_cfg, User_ID, Mic on by default) )
    Configs.insert( Query_tbl(Audio_cfg, User_ID, Hide self-view) )
    ...

    return Configs
```

The algorithm above shows a sketch of the algorithm necessary to solve our problem in pseudo-code. Unfortunately with our database approach this request is tedious and inelegant. The function will call the `Query_tbl()` function 9 times! Not only will this function be slow, due to the repeated queries to an external database, but it also violates the DRY software development principle we are adhering to, as discussed in 1.2. We now propose a solution to this issue that allows us to use at most 2 calls to the `Query_tbl()` function.

Entity relationship diagram — *2nd draft*

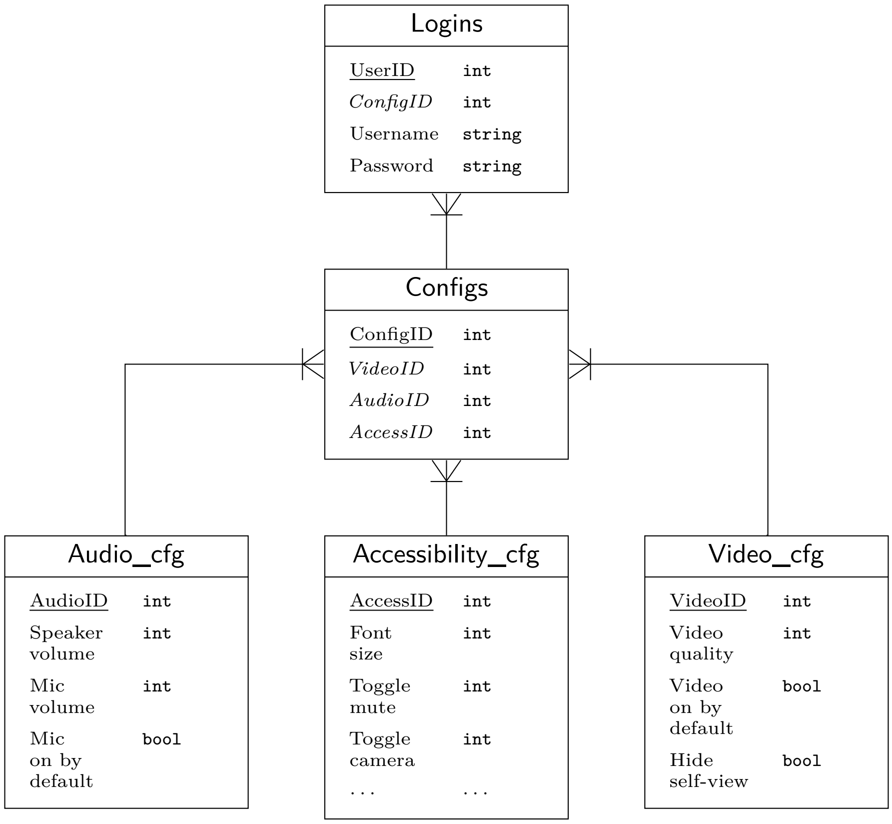


Figure 2.3: The 2nd draft ER diagram.

Explaining and justifying the 2nd draft ER diagram. The dots in the Accessibility_cfg table indicate that there are potentially more fields in this table.

The changes in the final ER diagram all revolve around the new ConfigID attribute. The user's complete configuration used to be comprised of numerous attributes that were stored across multiple tables. Now each collection of configuration attributes has been given it's own table in the Configs table. This table stores a number of IDs that each map to a collection of configuration attributes. In essence instead of treating each configuration attribute as it's own separate object, we now store all of a user's configurations together in one object, like how a bag stores a collection of items. This approach to our database design allows us to solve the problem we had in 2.2, now in order to request *all* of a user's configurations we may use the following two-liner;

```
Config_ID ← Query_tbl(Logins, UserID, ConfigID)
return Query_tbl(Configs, Config_ID)
```

Not only is this code much shorter and more clear than the code in the algorithm above but it will also run much faster, with just 2 calls to `Query_tbl()` . Moreover with this new approach we can still adhere to the DRY software development principle, since we are no longer forced to repeat code unnecessarily. The database is also in 3rd normal form enhancing the efficiency of our system as well as ensuring that data integrity is achieved. Another issue that this new design solves is duplicate configurations. For instance in figure 2.2 if 2 users happen to have the same configuration then this data will simply be duplicated in our table and take up unnecessary space in our table. However with the design in figure 2.3 if 2 users have for example the same audio settings then they can simply use the same audio ID, meaning that this specific audio configuration only needs to be stored once in our database. It is clear to see that with this new approach we not only save memory in the long term, at the cost of 1 extra table, but we also improve the performance of our system dramatically.

Although the database system will increase in size temporarily I argue that the size of the database will be smaller as compared to the draft database design as the number of users increases. All of our configuration options take a discrete set of values, for instance font size will only be able to take on an integer value $12 \leq f \leq 40$. For each configuration option let $c_1, c_2, \ldots, c_N$ be the of length N sequence of all the possible configurations for our system (note that this is allowed because all of configuration options are numerical, the `bool` data type can simply be represented by a number that is either 0 or 1). Denote c_i to be the i-th configuration and $|c_i| \in \mathbb{N}$ to be the number of possible options that this configuration can take on. Suppose that we have n users in our system, and each user will have a different set of configuration values. Define $X_{i,j}$ to be the value of the i-th configuration option for the j-th user. Let μ_i be the finite expected value of the i-th configuration. Then denote $\overline{X_i}$ to be the sample mean of the i-th configuration for all or our n users. From these definitions we have,

$$\overline{X_i} = \frac{1}{n}\sum_{j=1}^{n} X_{i,j} \tag{2.1}$$

Then by *Kolmogorov's strong law* [4] we have,

$$Pr\left(\lim_{n\to\infty} \overline{X_i} = \mu_i\right) = 1 \tag{2.2}$$

over all valid values of i. From the definition of a limit we can conclude that as n grows larger the sample mean converges almost surely to it's expected value. That means that as the number of users increases, the likelihood that users will choose the same configuration option increases significantly meaning the number of redundancies in our table increases significantly too. Whilst equation 2.2 can provide some motivation for the direction of our argument *Kolmogorov's strong law* cannot be applied because we can never have an infinite number of users in our system. Instead we will consider the configuration option c_m where this configuration option is able to take on the largest number of values.

Formally, we take a configuration option c_m such that $\max_{i=1}^{N} |c_i| = |c_m|$. We will then show that with a sufficiently large user base that users will inevitably start picking the same values for their configuration options. The options that are able to take on the largest number of values are the 2 volume options in Audio_cfg. We let users choose some integer volume $0 \leq v \leq 100$. By the *pigeonhole principle* once we have 102 users we must have at least 2 users who have chosen the same volume v. This is because we have 101 different options for the value that v could take on, and in the worst case scenario each user would take on a unique value of v. Above the threshold of 101 users every value of v would already have at least 1 user who has selected this value and hence the 102^{nd} user has to choose a volume value which someone else has already chosen. More generally starting from user 1, every 101^{th} user must choose a volume that has already been chosen before causing a new duplicate. This is seen by repeatedly applying the *pigeonhole principle*. As the number of users increase the number of duplicates for volume grows linearly. For instance if we have just $1011 = 101 \cdot 10 + 1$ users then we must have at least 10 duplicates for values of v. More worryingly since the volume example was the worst case scenario (because volume has the largest number of options that a user can choose) we can see that every other configuration option will also have to have ≥ 10 duplicates. We have 9 options total so across all tables there has to be at least 90 duplicates, and in fact the number of duplicates will be much greater in reality. The 4 boolean options will have duplicates every 2 users! Hence these options will at least 505 duplicates (from the calculation $2 \cdot 505 + 1 = 1011$). Multiplying by the 4 boolean options yields 2020 duplicates *minimum* over the 4 boolean options only! It is not hard to see that once the number of users starts to grow our database will rapidly fill up with duplicates resulting in catastrophically large database sizes and hence users will be unable to use our system as their data simply can no longer recorded in our database. With our final database design we can guarantee that our database won't have any duplicates saving me and my client from a multitude of economical and logistical issues in the future.

Entity relationship diagram — *3rd draft*

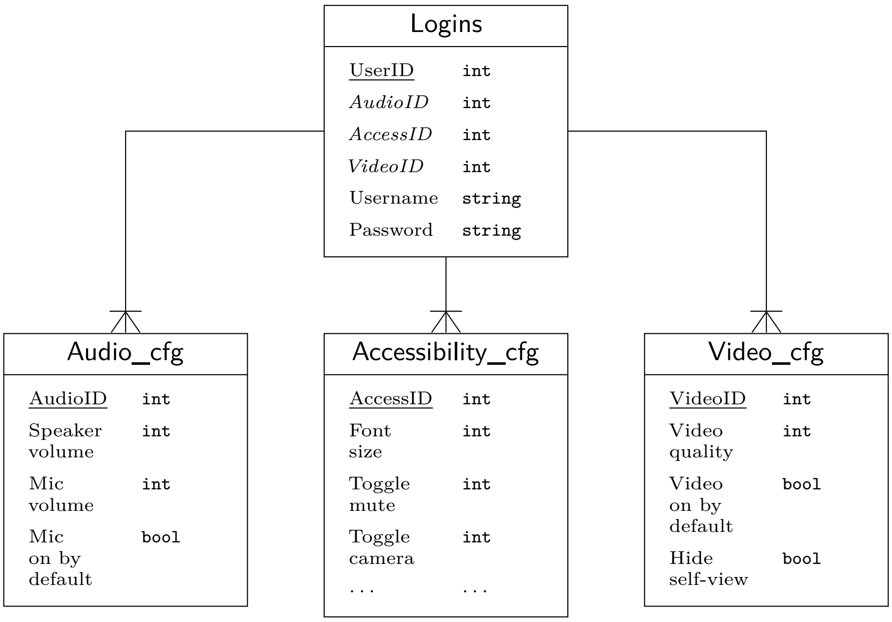


Figure 2.4: The 3rd ER diagram.

Explaining and justifying the 3rd ER diagram. We will first explain the shortcomings of the 2nd database design and then explain how the 3rd design was created in order to resolve these issues. The intention in creating the ConfigID key was to provide a clearer and more organised way to store data on the user's configuration options. In this way we wouldn't have to set up queries that ask for the value each option individually, we could instead simply query for the ID of the category of configuration we would like (audio, accessibility or video) and then use an object to store all the attributes in that category and their corresponding values. This means that we will reduce the number of queries that we would need to make to the database, improving the performance of our application. Additionally through employing this approach to the database design we make the configuration system more modular. Indeed if 2 users happen to have the same accessibility configurations then instead of storing a duplicate entry in our table the users can simply use the same AccessID.

Suppose that a user would like to change their configuration. How would we know whether this configuration is already taken, (so we should give this user the same ConfigID as the one already found in the database) or whether this configuration is new (so we should give this user a new unique ConfigID). The most obvious naive solution would be to iterate over all configurations checking whether or not configuration is the same as the one created by the user. Since different ConfigID's may have the same AudioID or the same VideoID, and etc [1], we will spend a lot of time performing redundant checks. Let's reinforce this argument using a concrete example, suppose that the 2 ConfigID's `001` and `002` are set up like so.

ConfigID	AudioID	VideoID	AccessID
001	521	126	078
002	521	126	024

Clearly the 2 configurations are exactly the same apart from their different accessibility settings. Now imagine that another user changes their configuration. We first check whether or not this user's configuration is the same as ConfigID `001`. We check all the attributes in Audio config `521` then all attributes in Video config `126` and then all the attributes in Accessibility config `078`. Suppose that the configurations happened to be different. We now have to check whether or not this user's configuration is the same as ConfigID `002`. We check again check all the attributes in Audio config `521`, then all the attributes in Video config `126` and then all the attributes in Accessibility config `024`. Now the problem is clear: we have checked over the same audio and video configuration twice! This is a huge waste of resources and the issue will become worse as more user configs are added to the database.

[1] Only 1 of the AccessID, AudioID, VideoID has to be different in order to create a new ConfigID.

The solution: to remove the ConfigID attribute. In studying this example we will find that the ConfigID attribute is unnecessary. Although it's addition was intended to provide clarity and organisation it ended up creating redundancies and needlessly complicating the system. Now in order to solve the problem of changing a user's configuration we can simply do checks against all audio configs, then all video configs and then all accessibility configs. Through this approach we know that each of the configs that we check is unique, and we ultimately avoid the downfall of the 2nd database draft, redundancies.

Entity relationship diagram — *4th draft*

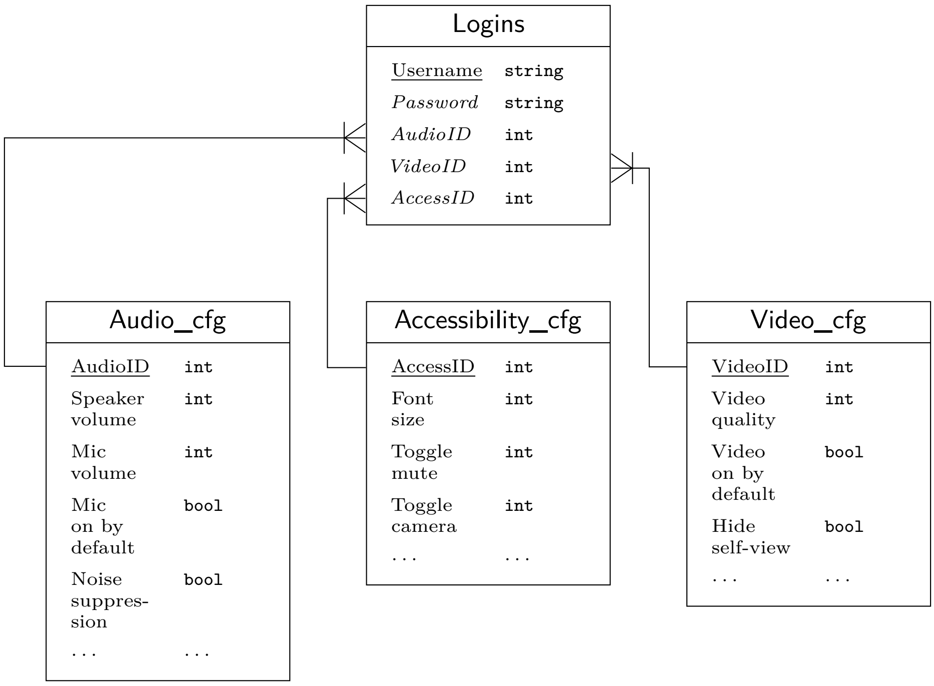


Figure 2.5: The 4th ER diagram.

Explaining and justifying the 4th ER diagram. In the 4th draft, the manner in which we have presented the database design has changed due to requests from my client. Each primary to foreign key relationship is now represented with an arrow that points directly to to the name of it's corresponding key. Moreover the logical relationships between attributes have been corrected. Finally the UserID attribute has been removed in place of a unique username to act as the primary key for the Logins table instead.

Note: Additionally during this time period my client asked me to change the database provider from MongoDB to Oracle Cloud.

¶2.2.2 Data flow diagrams

Proposed system level 0 DFD

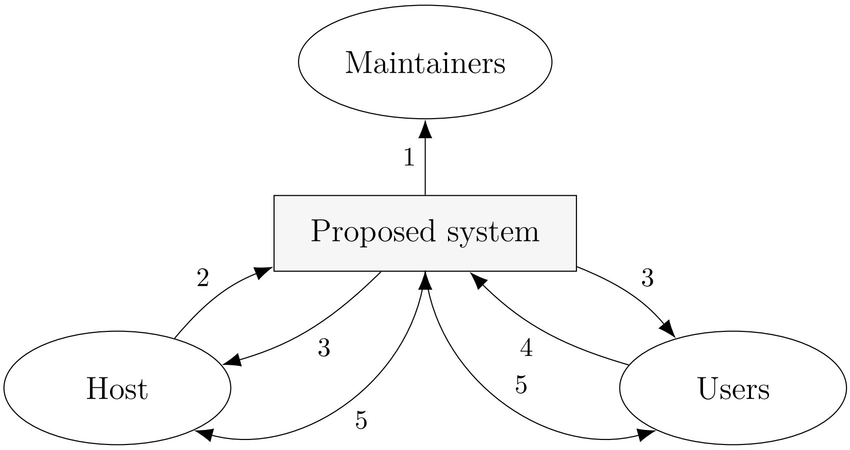


Figure 2.6: Proposed system level 0 DFD diagram.

An explanation of the DFD is found below. The number closest to each edge refers to the number in the left column of the explanation table below.

Table 2.1: Explanation of proposed system DFD.

Edge	Commentary
1	Maintainers receive and analyse information relating to the performance of the proposed system, and use the documentation provided with the system to suggest and make any necessary changes.
2	The host uses the proposed system in order to commence video conferences and to invite all those whom they would like to. Moreover the host will use the system to admit or remove people from the video conference, as they wish.
3	The system provides the host with live audio and video from all the other participants in the video conference, provided their microphone or camera are connected and that the user has activated their microphone or camera from their end.
4	The user connects to video conferences via a code that was given to them by the host of the meeting. They can also raise their hand and tailor their experience via the settings page.
5	Hosts and users are both able to communicate either via their microphones or via the built-in chat box on every video conference.

Justifying the DFD. The DFD above allows me and my client to be able to clearly understand the plans for the proposed system, in terms of some of it's main features and functionalities. Furthermore the DFD diagram can be compared to the previous system's DFD diagram to ensure that the new system is an adequate replacement, as discussed in 1.3.1. The usage of primitive shapes and clear labelling ensures that the diagram is easily digestible, such that even those who don't understand anything about software architecture will be able to grasp the structure of the proposed solution. Consequently these diagrams also work to improve the maintainability of the system, even new developers with minimal programming experience will have an intuitive understanding of how the proposed system works and it's overarching structure. Hence thanks to the creation of these diagrams we do not only equip the future maintainers of this system with all the necessary understanding that they require to make changes, but we also promote a design culture of clear and maintainable software. Then if the future maintainers of this software feel motivated to do the same we can ensure that our software will maximise it's longevity, through the constant innovation and improvement of our system namely via it's future maintainers.

Proposed system level 1 DFD

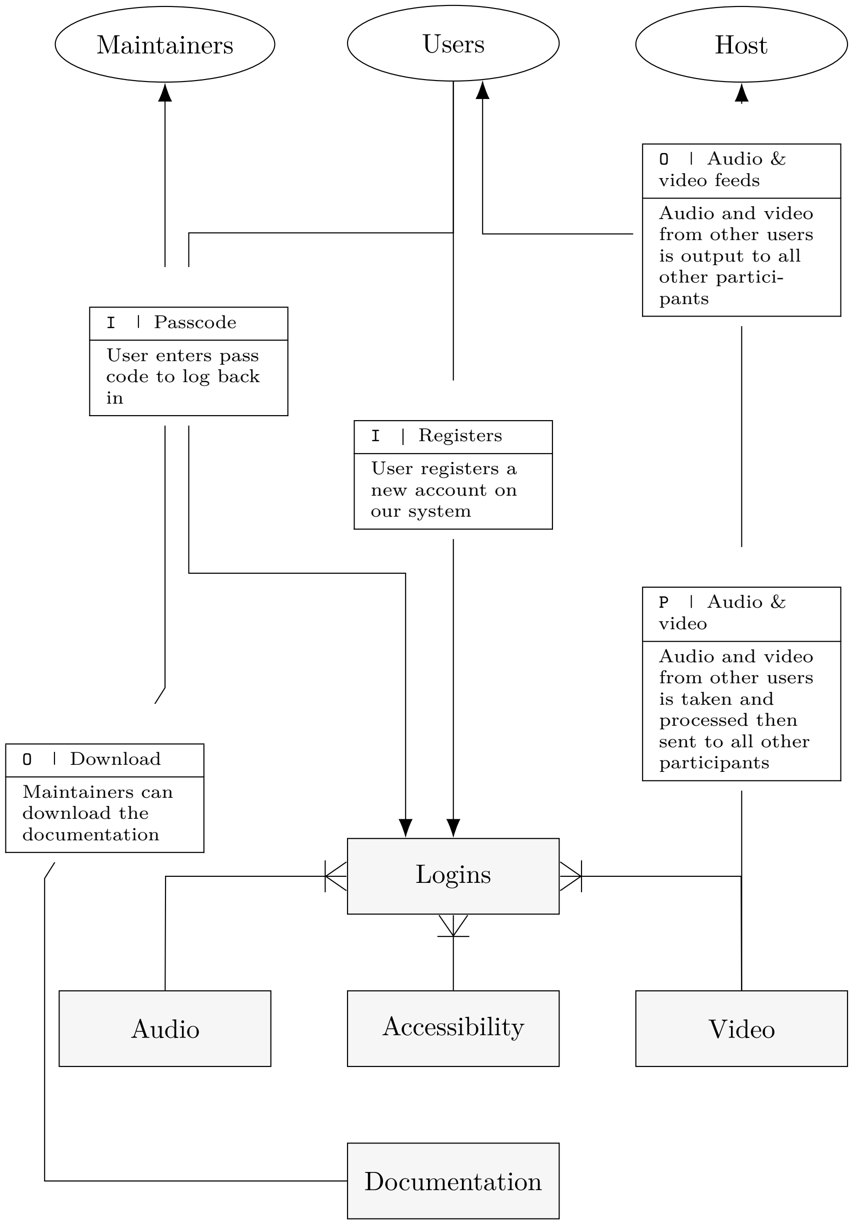


Figure 2.7: Proposed system level 1 DFD diagram.

The DFD is explained via the dialog boxes in the figure above.

Justifying the DFD. The level 1 data flow diagram exists to present a more in-depth look at the functionality of the proposed system, specifically in regards to the inputs and outputs of the system. Additionally the level 1 DFD diagram can help us to get a general understanding of the various inputs and outputs that our software will have to produce. Through these diagrams we can layout and clearly view the structure and design of our system holistically and see how each component of our system links to and integrates in with the other components. Moreover data flow diagrams can also help us in the testing our software. Through a study of the diagram we can notice that the figure 2.7 acts a guide for the expected inputs and outputs (I/O) of our system, which we can use to compare against the actual I/O of our system during the development process. Indeed the level 1 DFD coupled with the problem decomposition diagram serve as the "blueprints" of our system, visually illustrating the structure of our proposed system in a manner that is clear and simple to understand for both me and my client to refer to in the future. Finally we have seen that the level 1 DFD diagram provides a neat, cohesive visual representation of our system's architecture.

§2.3 Key algorithms

Note: *The boxed number in front of each subheading refers to the corresponding boxed number found in the system decomposition in 2.1.*

$\boxed{6}$ Account creation

Function name:	ADD_ACCOUNT(`string` Username, `string` Password)
Description:	This function will receive in a username and password and add these details into the login table, provided that the username is not taken. We assume that the username and password have already been properly validated via the algorithm in the previous subheading. It returns false if the username is taken and true if it is not.
Justification:	Users need to create an account in order to be able to save their configurations. For a detailed justification, refer to section 1.5

Algorithm 1 Pseudo code for creating a new user account.

```
function ADD_ACCOUNT(Username, Password)
    if Query_tbl(Logins, Username) ≠ null then        ▷ Cannot create account if username is taken.
        return False

    Insert_tbl(Logins, Username, Password, Default_cfg)   ▷ Set default settings for new user.

    return True
```

Iterative test plan: (These tests take place assuming that the logins table is empty each time)

Test type	Input	Expected Output	Justification
Normal	`(TerryTao, re%k91)`	`True`	The system should correctly be able to add a new user.
Erroneous	Call the function 2 times with this input `(TerryTao, re%k91)`	`False`	The system should not be able to add a user that already exists.

$\boxed{4}$ Validate username and password

Function name:	VALIDATE_ACCOUNT(`string` Username, `string` Password)
Description:	This function will receive in a username and password and check whether the given password is strong enough and whether or not the username is acceptable. A username is considered acceptable if it is at least 3 characters long, doesn't start with a special character, and only consists of the characters a-z, A-Z, 0-9, "_" or "-". Function should return true if the given username and password are acceptable.
Justification:	We should ensure that the user uses a strong password, and has a username that can be read. For instance if we had no restrictions on the username, the user could choose the username `" "` which is obviously not readable, as it only consists of white spaces.

Algorithm 2 Pseudo code for validating a username and password.

```
import zxcvbn
function VALIDATE_ACCOUNT(Username, Password)
    if Username.length < 3 then
        return false

    Pass_strength ← zxcvbn(Password)
    Legal_chars ← {'a', 'b', 'c', ...}                ▷ Set of acceptable characters

    if Pass_strength.score < 2 then                   ▷ Password is too weak
        return false

    for char in Username do
        if not char in Legal_chars then
            return false

    return true
```

Iterative test plan:

Test type	Input	Expected Output	Justification
Normal	`("David_Hilbert1", "i#1Go8")`	`True`	The function should return true if the username fits the criteria stated and if the password is strong enough.
Extreme	`("Dav", "i#1Go8")`	`True`	The function should still return true if the username given is 3 acceptable characters.
Erroneous	`("c", "123")`	`False`	The function should return false if the password is too weak and the username is not considered acceptable.

(5) Log user in

Function name:	LOG_USER_IN(`string` Username, `string` Password)
Description:	This function should take in a user's username and password and then load in their relevant configurations. We should check that the password given matches the password stored in the database. Their customised settings should all be correctly loaded into the system, once we have confirmed that the user has successfully logged in. The function should return true if the user has successfully logged in and false if they have not.
Justification:	The initiative to provide accessibility options and configurations was suggested by my client as a non-essential feature as was seen in section 1.4.5. Through providing our users with these options and configurations we enable them to customise their video conferencing experience so that their settings are tailor-made to suit their individual needs. We justify confirming that the user has logged in by noting that in order for the user to be able to access their configurations we must know this user's UserID. Without the UserID we will not be able to query the database and therefore we won't be able to load their configurations.

Algorithm 3 Pseudo code for logging a user in.

```
function LOG_USER_IN(Username, Password)
    if Query_tbl(Logins, Username) = null then                     ▷ User doesn't exist.
        return False

    if Query_tbl(Logins, Username, Password) ≠ Password then       ▷ Wrong password.
        return False

    UserID ← Query_tbl(Logins, Username, UserID)
    Video_ID ← Query_tbl(Logins, UserID, VideoID)
    Audio_ID ← Query_tbl(Logins, UserID, AudioID)
    Access_ID ← Query_tbl(Logins, UserID, AccessID)

    Set_Video(VideoID)
    Set_Audio(AudioID)
    Set_Access(AccessID)

    return True
```

Iterative test plan (These tests take place assuming that the logins table only has one account setup with the name `TestAcc1` and password `A2d£l!`):

Test type	Input	Expected Output	Justification
Normal	`(TestAcc1, A2d£l!)`	`True`	The system should function correctly when loading in a an actual user's configurations with correct information.
Extreme	`(TestAcc1, password)`	`False`	The system should recognise that the password inputted is incorrect and should not allow the user to log in.
Erroneous	`(kirishkiwi, password)`	`False`	The system should recognise that this user doesn't exist and should not allow them to log in.

16 Update user settings

Function name:	SET_USER_CFGS(`string` Username, `string` Category, `string` Option, `int` Value)
Description:	This function will update the user's current settings. If the given user doesn't exist the function will return false. It will save these changes to the database so that they can be retrieved later. If the specific configuration is already stored in the database than the user's config id will change to the id of the configuration already stored in the database. Otherwise a new configuration id will be created and assigned to the user.
Justification:	By allowing the user to be able to save their customised settings we save them a deal of time. Users won't have to repetitively re-configure their settings each time they log in, instead their settings are saved in our database to be loaded in. Since our target user base is the elderly it is also likely that their memory is poor, by enabling these ones to set their configurations once and then have these settings be saved automatically we also save them from having any issues remembering how to set up their config.

Algorithm 4 Pseudo code for updating a user's settings.

```
function SET_USER_CFGS(Username, Category, Option, Value)
    If Check_User_exists(Username) ≠ True: return False          ▷ User doesn't exist.

    UserID ← Query_tbl(Logins, Username, UserID)
    AudioID ← Query_tbl(Logins, UserID, AudioID)
    VideoID ← Query_tbl(Logins, UserID, VideoID)
    AccessID ← Query_tbl(Logins, UserID, AccessID)

    Audio_cfg ← Create_Audio_cfg(AudioID)          ▷ These are objects storing the user's current settings.
    Access_cfg ← Create_Access_cfg(AccessID)
    Video_cfg ← Create_Video_cfg(VideoID)

    If Category = "Audio": Config ← Audio_cfg          ▷ Config is the configuration object being updated.
    If Category = "Video": Config ← Video_cfg
    If Category = "Access": Config ← Access_cfg
    Config.Option ← Value

    if Config exists in DB then
        ConfigID ← Find_ConfigID(Config)
        return

    ConfigID ← New_ConfigID()          ▷ Create a new unique config id.
    return
```

Iterative test plan: (Assume that only the user "Warren2" exist in the database).

Test type	Input	Expected Output	Justification
Normal	`("Warren2", "Font size", 24)`	`None`	The system should be able to update a user's configuration without crashing.
Erroneous	`("Howard9", "Font size", 24)`	`False`	The function should return false if the user doesn't exist.

9 / 14 Create call

Function name:	CREATE_CALL()
Description:	This function will start a new call for the user. It will return a unique passcode that the user can share in order to invite others to join the call.
Justification:	The ability to be able to create a call is a necessity for this application. It is one of the primary requests from my client, and without it this software would not be a *video conferencing* system.

Note: This is a modified, pseudo-code version of the code found at the article https://www.100ms.live/blog/webrtc-video-call*. All credit goes to it's author, Karthikeyan S.*

Algorithm 5 Pseudo code for starting a new call.

```
import WebRTC
function CREATE_CALL
    Local_Stream ← GetUserMedia(video = true, audio = true)
    Connection ← RTCPeerConnection()

    Connection.addTrack(Local_Stream.get_track(), Local_Stream)   ▷ Add the local stream to the connection.
    Connection.createOffer()                                      ▷ Create a SDP offer. (See 1.4.6 for details.)
    Signal(Connection.icecandidate)                               ▷ Send the ICE candidates. (See 1.4.6 for details.)

    Try
        Signal(Connection.offer)                                  ▷ Send the offer to the signalling server.
    Catch(error)
        print("Error creating offer!", error)

    return New_Call_Code()
```

Iterative test plan:

- Ensure the function can be called without crashing.
- Verify whether a video call was setup correctly.
- Verify whether a unique code was generated.

Justification:

- Users should be able to start a video call without the system crashing.
- The function should create a new video call successfully.
- The function should return a unique call passcode in order to allow other users to join.

10 Join call

Function name:	JOIN_CALL(`int` Code)
Description:	This function will allow the user to join the video call associated with the code that they entered. If no such video call exists, it will return false. Otherwise if the call was joined successfully it will return true.
Justification:	Again in order for my client's primary request to be fulfilled, we must be able to have multiple users join a video call with each other.

Note: This is a modified, pseudo-code version of the code found at the article `https://www.100ms.live/blog/webrtc-video-call`. *All credit goes to it's author, Karthikeyan S.*

Algorithm 6 Pseudo code for joining a call.

```
import WebRTC
function JOIN_CALL(Code)
    If Call_Exists(Code) = False: return False

    Local_Stream ← GetUserMedia(video = true, audio = true)
    Connection ← RTCPeerConnection()

    Connection.addTrack(Local_Stream.get_track(), Local_Stream)    ▷ Add the local stream to the connection.
    Connection.createAnswer()                                      ▷ Create a SDP answer.
    Signal(Connection.icecandidate)                                ▷ Send the ICE candidates.

    Try
        Signal(Connection.answer)                                  ▷ Send the answer to the signalling server.
    Catch(error)
        print("Error creating answer!", error)

    return True
```

Iterative test plan:

- Ensure the function can be called without crashing.
- Verify whether a video call was joined correctly, provided a call with that code exists.
- Verify whether the function correctly returns false if a call doesn't exist with this code.

Justification:

- User should be able to join a call without the system crashing.
- Function should allow user to join a video call provided they enter the correct code and a call with that code exists.
- Function should not allow user to join call if no call corresponding to the code given exists.

Justify how these algorithms form a complete solution. Finally in relating each of these algorithms (with the numbering) to the system decomposition in 2.1 we have demonstrated fully how these algorithms form a complete solution to the problem at hand.

§2.4 Usability features

All website design mock ups were all created in Figma. These are just mock ups see chapter 3 for later iterations.

Design inspiration from:

https://www.youtube.com/watch?v=MnUWG3VLufI&t=245s&ab_channel=AlienaCai

https://www.awwwards.com/

Home page

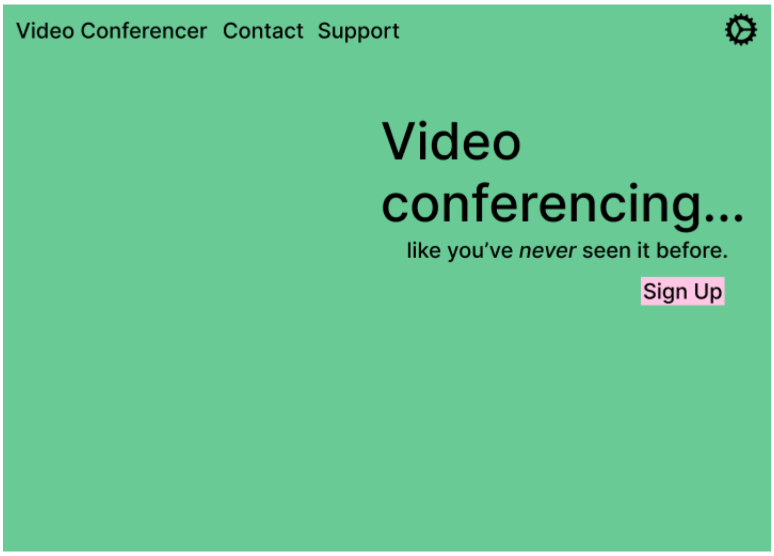


Figure 2.8: Mock up of the home page.

Description: The home page mock up is comprised of a navigation bar, a headline and sub headline along with a call to action button.

Justification: In UI/UX design it is often taught that the user first looks at the top of the web page. Since the home page should be the page that captures our audience's attention, we want design it such that anyone who looks at the page can tell what the product we are selling is within 5 seconds. We achieve this in our headline using a large and bold font. The call to action button is the "Sign Up" button and we want our users to click it, hence we have made use of our accent colour to make the button stand out from the rest of the page. Since our target users may be unfamiliar with technology it is important to ensure that the home page is designed to look professional, however it is just as equally as important to ensure that our page is easy to navigate. In this manner even our users who have little to no experience with technology will be able to use the system easily. Indeed this homepage design organises the content we would like to show in an organised and cohesive manner enabling such users to navigate our site as they so desire.

Log in/Sign up page

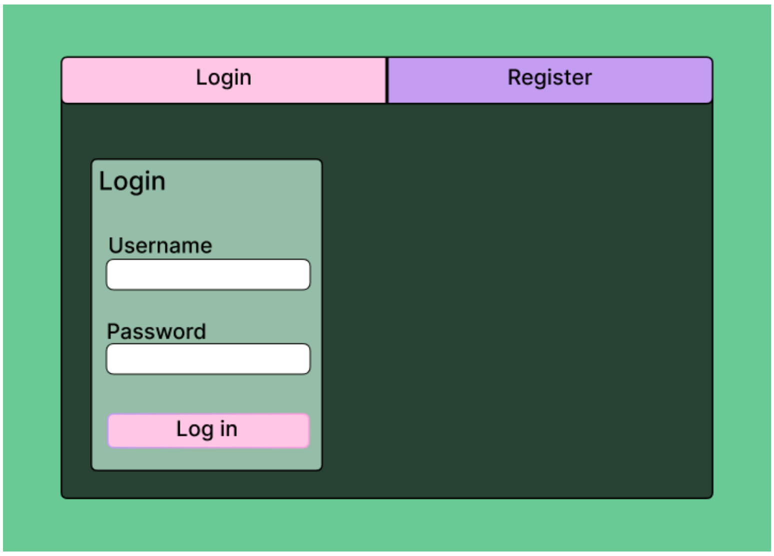
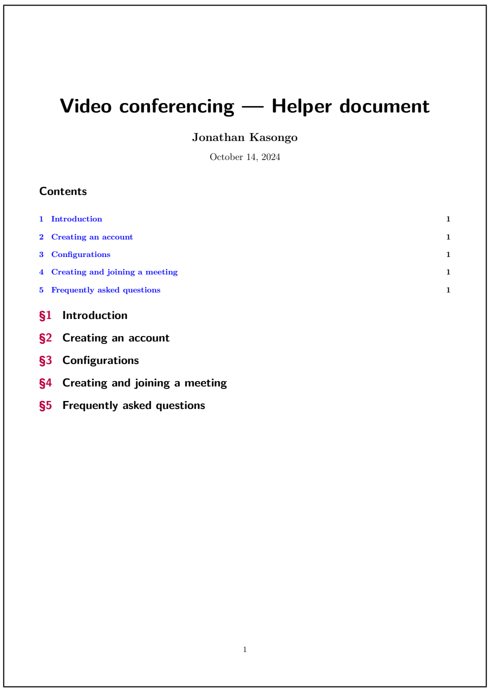


Figure 2.9: Mock up of the log in/sign up page.

Description: The log in/sign up page mock up is comprised of 1 main body part that is inside a rectangle with smoothed corners. The main body part is comprised of 2 tabs the login tab and the registration tab. First time users will create their account via the registration tab whilst returning users will log back in via the login tab. Finally users will be able to enter their username and password into the username and password fields respectively, then click "log in" to use our system.

Justification: Since our target user base is the elderly I chose to display the text in a large and readable font, because it is quite likely that a good number of our users will have trouble with their eyesight. Consequently this means that our users shouldn't have problems with reading the text on our page. In the design I tried to create a nice design based on the colouring scheme that was outlined in section 1.7.5. The design was chosen to create a professional look for the system as well as make the user feel calm and comfortable whilst on our website. In developing this calm theme throughout the design of our application we hope that the user will feel that our system is comfortable and reliable so that we hence provide a strong enough motivating factor to encourage the user to switch from the old system to our system.

Help PDF document


Video conferencing — Helper document

Jonathan Kasongo

October 14, 2024

Contents

1	Introduction	1
2	Creating an account	1
3	Configurations	1
4	Creating and joining a meeting	1
5	Frequently asked questions	1

§1 Introduction

§2 Creating an account

§3 Configurations

§4 Creating and joining a meeting

§5 Frequently asked questions

1

Figure 2.10: Mock up of the helper document.

Description: The figure above presents the mock up that was created for the helper document that will be created along with this system. The document follows roughly the same formatting as the one in this report, and has been split into 5 primary sections each providing a guide on how to use one portion of the software. Finally at the end of the document we add a frequently asked questions section (FAQ) to put answers to common questions that our user may have regarding how to use the system.

Justification: On the helper document we have chosen to stick with the style used in this report providing a consistent and familiar theme to our users. The blue section links provide a quick way for our users to be able to navigate to the sections that they specifically need. This document is intended to be read by our future users so it is imperative that the text be easy to read, as well as comprehensive and thorough, in order to answer any of our users' questions. Finally the text must be easy to read so that our target audience, the elderly, will be able to comprehend our text without issues, since it is not uncommon for a person's cognitive thinking capacity to decline as they become old.

Macros

Description: Users will be able to program set buttons to perform specific actions via their accessibility settings. For example the user may want to set the space bar to be the button that allows them to mute/unmute their microphone during a video call.

Justification: Considering our target demographic it is likely that some of our users will have trouble using a mouse to navigate our system due to conditions like Carpal tunnel. By providing another way to navigate the video-conferencing system we allow people suffering with all sorts of immobilising conditions to be able to navigate our system comfortably. This aligns with the motivations of our client who wishes to create an *accessible* video-conferencing system. Indeed by implementing features that allow users with mobility issues to be able to use our system comfortably we are directly improving the accessibility of our system, through increasing the number of people who could potentially use our system.

Video tutorial

Description: A video tutorial will be uploaded onto YouTube providing a guide on how to use our system effectively.

Justification: Although we already provided a helper document, we make efforts to ensure that the users who dislike reading or perhaps have medical conditions that make it challenging for them to read, are still able to access a simple guide on how to use the software properly. In doing so we prevent the need for our users to have to contact someone to support them when video-conferencing, instead we promote their independence in learning and understanding how to use new technologies.

§2.5 Key variables and data structures

¶2.5.1 Key variables

The following table outlines the key variables that will be used in our system along with a description of each and a justification for each.

Table 2.2: Key variables and data structures.

Variable	Data type	Description	Justification
Logins table			
Username	`string`	A unique string of length 3-20 characters that will be used to identify a user.	Unique usernames help other users to identify the the identities of the other participants who are in their call. Since they are unique they can also form a primary key as we are doing so in the Logins table.
Password	`string`	A string that will be entered by the user to log into their account, as a way for the system to confirm that it actually is the correct person logging into this account.	The usage of password protected accounts makes us compliant with the GDPR. Passwords are beneficial to the user as if they correctly use a strong password they should never have to worry about hackers accessing their account and all their private data.

Continued on next page

Table 2.2: Key variables and data structures. (Continued)

AudioID	`int`	An unique integer code that will be used to reference the user's personal audio configuration in the table Audio_cfg.	In using a code to represent the user's configuration we prevent redundancies from user's selecting the same configurations as discussed during the design of our database ERD. The usage of the integer data type will provide more than enough codes for all users as well as a simple way to generate a new unique code, adding 1 to the last generated code.
AccessID	`int`	An unique integer code that will be used to reference the user's personal accessibility configuration in the table Access_cfg.	The justification is the same as the one found in the AudioID row.
VideoID	`int`	An unique integer code that will be used to reference the user's personal video configuration in the table Video_cfg.	The justification is the same as the one found in the AudioID row.
Audio table			
Speaker_Volume	`int`	The speaker volume will be an integer in the range 0-100 where 0 is the lowest and 100 is the highest, indicating how loud the output volume should be.	This variable allows our user to be able to control and save the volume level that they want their video conference to default to each time they log in. This helps to avoids issues like the audio being too loud and disturbing people, and the audio being too quiet so that the user can't hear.
Mic_Volume	`int`	The microphone volume will be an integer in the range 0-100 where 0 is the lowest and 100 is the highest, indicating how loud the microphone volume should sound to others.	This variable will allow the user to save the volume of their microphone so that when they log into a new video call their microphone volume will be this value. By allowing users to save their microphone volume we prevent them from having to adjust the volume of their microphone manually every time they log onto a video call.

Continued on next page

Table 2.2: Key variables and data structures. (Continued)

Mic_On_Default	`bool`	True indicates that the user's microphone will be on as soon as they enter the video call, false indicates that the user's microphone will be off as soon as they enter the video call.	This variable would be adjusted depending on the occasion of the video call. For example if the user is attending a public conference then they may want to join with their microphone automatically off, so that they do not disturb others. However if the occasion is a company meeting they may wish to turn their microphone on automatically so that they will be able to participate in the meeting without facing the issue of speaking whilst they are muted.
Noise_Suppression	`bool`	True indicates that the browser will try to suppress background noise from the user's microphone, false indicates that the browser will not suppress the microphone's background noise.	If the user happens to attend the video call in a particularly noisy area, it may be difficult for others to be able to hear them clearly. By providing the option for our users to apply background noise suppression we allow users to be able to conduct video calls in more locations, improving the accessibility of our software
Video			
Video_Quality	`int`	An integer representing the video the user's desired video quality. More specifically the user's desired video resolution.	Since different users may have access to different internet connection speeds, our users should be able to control their video quality so as to be able to have the lowest latency video conferencing experience possible without major quality compromisation. Indeed it is through the Video_Quality variable that allows our users to be able to achieve this necessity.
Hide_Selfview	`bool`	True if the user would like to hide their own video, false if the user would like to see their own video feed.	Our users may want to be able to hide their own video feed for a multitude of reasons, they may find it distracting, it might be embarassing/awkward for them, and more. By giving the user the option to be able to hide their own video we provide the user with a more comfortable, natural video conferencing experience.

Continued on next page

Table 2.2: Key variables and data structures. (Continued)

Video_On_Default	`bool`	True if the user would like to join a video conference with their video camera on automatically, false if the user would like to join a video conference with their video camera off automatically.	This variable allows our users to be able to avoid repetitively changing their settings in each video call, for example if the user primarily uses video conferencing software to log onto team meetings at work where they need to have their camera on (company policy) they won't need to worry about forgetting to turn their camera on whilst they are interacting with their team members every time.
Access table			
Toggle_Mute	`int`	An integer representing the key code of the key that the user has set to toggle between muted and unmuted in a video call.	By allowing our users to set macros for commonly performed tasks, we improve the user experience by letting them choose the keys most comfortable to them to use as macros. This is especially helpful to our users who may suffer from mobility issues, where controlling a mouse may be a pain. We justify use of the integer data type by noticing that in all modern software, integer codes are used to reference each individual key. It has become standard to use the key codes layed out by the Win32API, so we will not take time to re-invent the wheel and simply adopt the already existing standards.
Toggle_Camera	`int`	An integer representing the key code of the key that the user has set to toggle their camera on and off in a video call.	Justification is the same as the one found in the Toggle_Mute row.
Toggle_Hand	`int`	An integer representing the key code of the key that the user has set to raise their virtual hand up or down in a video call.	Justification is the same as the one found in the Toggle_Mute row.
General			
Peer_Connection	`RTCPeerConnection`	The interface used in order to set up a peer to peer connection via WebRTC.	The usage of WebRTC's peer to peer connection object will provide an efficient and robust implementation of abstracted peer to peer connection. We know this because the WebRTC API has been thoroughly tested by many others seeing as the initial release was 13 years ago.

Continued on next page

Table 2.2: Key variables and data structures. (Continued)

Local_Stream	`MediaStream`	This variable will store and provide an abstracted interface for the user's video and/or audio data.	The usage of WebRTC's MediaStream object will provide the same benefits as was discussed above as well as integrate seamlessly with the rest of the WebRTC components of the system.

¶2.5.2 Data validation rules

The following table outlines the data validation rules that must be obeyed in our system for it to function as required, along with justification for each rule.

Table 2.3: Data validation rules.

Validation	Description	Justification
Data type	Each variable must use only 1 data type, data of the wrong type should never be assiged to any variable.	Data type validation is necessary for the system to function without runtime errors being raised. In ensuring that this validation rule is obeyed we guarantee that the system will never produce type errors. If the user encountered such an error whilst using the finished application then it could produce some unexpected behaviour, and the user may be tempted to move back to the previous system on the basis that it was more reliable.
Existence	If data must be found in a field, we check for it's existence. For instance each user account must have a unique username associated with it in the Logins table.	We need to have existence checks in place to avoid unexpected behaviour, and failures in our software, to illustrate suppose that we didn't use existence checks and our Logins table was now full of users that have blank usernames due to some bug elsewhere in the system. Now our database will quickly fill up with data that is useless because none of our users will be able to use the usernames that they chose in order to log in.
Uniqueness	Variables that act as primary keys must be unique.	Checking for uniqueness is vital to the functionality of our system. If 2 users get the same username, we would not only be wasting space in our database by storing a redundant entry but we would also experience issues when one of the users wants to log in, because we would never be able to know whether the password entered is correct.

Continued on next page

Table 2.3: Data validation rules. (Continued)

Range	All numerical variables n that are restricted to some interval $I = [l, r]$ must satisfy $n \in I$. All string typed variables s whose length is restricted to some lower and upper bound residing in interval $J = [a, b]$, where a is the minimum string length and b is the maximum string length, must satisfy $\|s\| \in J$, $\|s\| \in \mathbb{N}$.	For the numerical variable we must check that they satisfy the range requirement because without it errors and unexpected behaviour may arise, ruining the user experience. For the string typed variables these must satisfy their length requirements so as to not cause issues for the system or for other users. As an example suppose that we did not have the username length requirement then users could select a username of length 0, a username with no characters, meaning that the other users wouldn't be able to identify who has joined their video conference. On the flip side a user could pick a username of length 100,000 characters, which is obviously unnecessary for a username, this username would then use up a large amount of the storage available in our database.

Table 2.4: Validation justifications.

Variable	Validation	Justification
Username	Must be a `string`, of length 3-20 characters.	Users shouldn't be able to have usernames as other data types because a username is used to identify users as humans. Humans do not refer to each other using numbers so it wouldn't make sense to have the username be an `int` data type. The length check is there to ensure that users do not choose usernames that are too short, for example people do not usually have 1 or 2 letter names, similarly people do not usually have 20 or 30 letter names. If we didn't have a restriction on the number of characters one could use for their username then they could potentially pick an extremely long username (for instance what if a user picked a username with a million characters?) and this would take up a lot of memory within our user account database.
Password	A `string` of length 7-30 characters.	User passwords are strings to allow users to use numbers and characters in their password, thus strengthening the safety of user accounts as this allows for many more options when it comes to password selection. The justification for the restriction of length is similar to the one mentioned above.

Continued on next page

Table 2.4: Validation justifications. (Continued)

Video quality	An `int` q satisfying the condition $360 \leq q \leq 8192$	It is a universal standard to use integers to represent video quality. 360p is the lowest video resolution that one can consider to be clear video, and in our current day the highest resolution screens are 8K screens which can have resolutions of 8192p.
AudioID, AccessID, VideoID	An `int` n such that the condition $0 \leq n \leq 10^9$ is satisfied.	We use an `int` data type because all of the ID's were defined to be integers in table 2.2.
Speaker_Volume, Mic_Volume	An `int` v such that the condition $0 \leq v \leq 100$ is satisfied.	Volume has to be an integer so that we can compare values. For instance it easy to see that the volume of 75 is louder than a volume of 50, however it is not easy to tell whether or not `"pipj"` is a greater volume value than `"goood"`.
Mic_On_Default, Noise_Suppression, Hide_Selfview, Video_On_Default	A `bool`.	All of these variable will be in either on or off states, this is exactly what a boolean does.
Toggle_Mute, Toggle_Camera, Toggle_Hand	An `int` c satisfying the condition $8 \leq c \leq 222$	In Javascript the integer key codes range from 8 to 222.*

* *Source:* `https://www.cambiaresearch.com/articles/15/javascript-char-codes-key-codes`

§2.6 Testing

The iterative test data along with it's justification for each key algorithm is given underneath it's pseudocode see section 2.3.

¶2.6.1 Alpha and beta testing

Alpha testing is done by the software developers and their team, before the launch of the system. In carrying out alpha testing we can ensure that our system fully meets the needs of our client and end users as well as verify that our system has no hidden bugs. Moreover by thoroughly testing our software during alpha and beta testing we reduce the risk of our software failing during it's actual release by gaining feedback from our testers and verifying that our software has no security vulnerabilities and/or legal compliance issues.

Beta testing is carried out by a group of potential end users, before the launch of the system. These tests simulate a more realistic usage of our system, so the beta testers may end up discovering more bugs or inadequacies with the system after alpha testing since users often try things that the developers have not though of. Additionally by testing with a group of potential end users we gain insight on how the software performs under increased loads and stresses, then use this data to justify changes and enhancements to our system.

¶2.6.2 Development test plan

Let username be U and password be P, for typographical reasons.

Table 2.5: Development test plan.

Test #	Description	Test data	Expected outcome	Justification

Continued on next page

Table 2.5: Development test plan. (Continued)

1.1	Create an account. *Normal*	(U, P) = `(Thomas12, u#4t3&$̂)`	The system should correctly add a new user account, provided that password is strong enough.	Users must be able to make an account in order to save their configuration data. Their username also helps others to identify them. This test contributes to to completion of success criteria 5.
1.2	Create an account. *Erroneous*	(U, P) = `(Thomas12, 123)`	The system should not add a new user account, provided that the password is not strong enough.	Users shouldnt be able to make an account if their password is too weak. This not only allows us to comply with the relevant GDPR regulations but also ensures that our user's data is secure. This test contributes to to completion of success criteria 5.
2.1	Log in. *Normal*	Assuming the relevant account exists. (U, P) = `(Thomas12, u#4t3&$̂)`	The system should correctly allow a user to log in to their account, provided that the correct password is entered.	Once a user has created an account and setup all of their configurations they should be able to access their account by logging in with the correct details. This test contributes to to completion of success criteria 5.
2.2	Log in. *Erroneous*	Assuming the relevant account exists. (U, P) = `(Thomas12, abc)`	The system should not allow a user to log in to their account, provided that the wrong password is entered.	If a user enters the wrong password they shouldn't be able to log in to the system, this measure is in place to protect our user's data. This test contributes to to completion of success criteria 5.
3.1	Log out. *Normal*	Assuming the user is logged in. Navigate to account profile, click the log out button.	Users should be able to log out of their account once they are finished using our application.	Users should be able to log out of their account so that malicious internet users wouldn't be able to access their account whilst they are away from their computer. This test contributes to to completion of success criteria 5.
4.1	Type in the chat box, *Normal*	Assuming that a conference is running. Message = `Hello!`	The system should allow users to be able to type in the chat box provided in the video conference.	This feature will enable users to be able to communicate with each other in the case that they do not have a microphone. This test the implementation of essential feature 6. (See section 1.5)

Continued on next page

Table 2.5: Development test plan. (Continued)

5.1	Access documentation. *Normal*	Click on the "Docs" button.	The documentation PDF should open in a new tab.	This is an open source project that will be passed off to other developers once complete, it is therefore necessary to ensure that the future developers are able to fully understand the system's architecture and it's code. By opening the documentation in a new tab we allow our users to be able to easily access and read through the system documentation in their own web browser. This tests the implementation of essential feature 8. (See section 1.5)
6.1	Create video call. *Normal*	Navigate to the create call page. Click on the create call button.	Video call started successfully.	Since this is a video conferencing software it is vital that the software is able to allow users to video conference, it's intended purpose. This test will ensure that users will have an enjoyable user experience whilst video conferencing. This test contributes to the completion of success criteria 7 and 8.
7.1	Join video call *Normal*	Assuming a video conference exists with the passcode: `041058`. Enter the passcode `041058`.	Video call joined successfully.	Users should be able to join a video call provided that one exists and that they have entered the correct code. This test contributes to success criteria 7 and 8.
7.2	Join video call *Erroneous*	Assuming a video conference exists with the passcode: `041058`. Enter the passcode `123456`.	Video call not joined. Prompt displaying "Invalid passcode!".	Users shouldn't be able to join a video conference if they have entered the wrong passcode. This test contributes to success criteria 7 and 8.

¶2.6.3 Post-development testing

Once the software development is finished I will ask a small group of potential end users to carry out the following tasks, and report how easy/difficult the task was to complete.

Test 1

Task: Create an account.

Test type: Normal.

1. Navigate to the sign up page.
2. Enter an adequate username into the username field.
3. Enter the password `if5#@Jal` into the password field.
4. Click on the sign up button.

Expected outcome: User account created successfully.

Test type: Extreme.

1. Repeat the steps except use the password `AopS12v`.

Expected outcome: User account created successfully.

Test type: Erroneous.

1. Repeat the steps except use the password `123`.

Expected outcome: User account not created, display message saying password is too weak.

Test 2

Task: Log in to your account.

Test type: Normal.

1. Navigate to the log in page.
2. Enter your actual username into the username field.
3. Enter your actual password into the password field.
4. Click on the log in button.

Expected outcome: User logged in successfully.

Test type: Erroneous.

1. Repeat the steps except using the wrong password.

Expected outcome: User not logged in, message displaying "Wrong username or password!".

Test 3

Task: Log out.

Test type: Normal.

1. Navigate to your account profile.
2. Click the log out button.

Expected outcome: User logged out.

Test 4

Task: Change settings.

Test type: Normal.

1. Navigate to the settings page.
2. Change some settings and record these changes.
3. Log out of your account (refer to Test 3).
4. Log back into your account (refer to Test 2).
5. Navigate back to the settings page.
6. Verify whether the settings there are the same as the ones you recorded in step 2.

Expected outcome: User settings changed successfully.

Test 5

This test relates to the usability feature: "Video tutorial". Refer to section 2.4 for details.

Task: Access and watch the software video tutorial.

Test type: Normal.

1. Navigate to the home page of our system.
2. Click on the help button.
3. Click play on the video and watch the video tutorial.

Expected outcome: Video tutorial begins playing in the page.

Test 6

This test relates to the usability feature: "Help PDF document". Refer to section 2.4 for details.

Task: Access and read the help PDF document.

Test type: Normal.

1. Navigate to the home page of our system.
2. Click on the help button.
3. Scroll down and click on the help document link.
4. Read the PDF document that was downloaded.

Expected outcome: The help PDF document should be downloaded and opened in a new tab.

Test 7

Task: Start a video call.

Test type: Normal.

1. Connect a webcamera and microphone to your device (if necessary).
2. Navigate to the create call page.
3. Click on the create call button.

Expected outcome: Unique call code generated successfully and displayed to the user. User is able to view themselves in the video call.

Test 8

Task: Join a video call.

This test requires 2 people. Both users should have a camera and microphone connected to their device. Both users should have a stable internet connection satisfying the requirements listed in section 1.7.9

Test type: Normal.

1. Have person 1 complete the steps outlined in Test 5.
2. Person 1 should share the unique call code with person 2.
3. Person 2 should then navigate to the join call page and enter the code that they were given.
4. Person 2 should click on the join call button.

Expected outcome: Both users successfully joined the video call and can clearly see and hear each other.

After the user has completed a task in the test plan we ask our user to fill out the following feedback form.

Table 2.6: Post-development feedback form.

No	Question	Rating (1-10)	Additional comments
1	How easy was the task to complete?	10 being very easy, 1 being very difficult	
2	How easy was it to navigate the site?	10 being very easy, 1 being very difficult	
3	How would you rate the design of the site?	10 being very good, 1 being very poor	
4	How would you rate the user experience of the site?	10 being very good, 1 being very poor	
5	Any additional comments or suggestions	N/A	

Justification: This suite of tests adequately tests the functionality of all of the main features of this system. In asking a diverse group of potential end users to carry out the above tasks we gain insight as to how real people learn to navigate and use the system. Moreover we can analyse the user feedback to identify the strengths and weaknesses of our system and implement any needed changes based on the feedback received. Hence the test serves to confirm whether or not our system works as intended as well as inform us on ways in which we can improve the system. It is imperative to have potential users test our system because these users may potentially try things that I would have never have thought about. In receiveing user feedback after getting people to complete the post-development tests we gain valuable insight into the how our end users will navigate around and use the system.

Development

§3.1 Development overview and sub-tasks

We apply the principle of decomposition to our development plan by splitting up the development of the system into a number of smaller sub-tasks. Each sub-task relates to it's relevant circled number in our problem breakdown (see section 2.1).

In doing this we **clearly** relate each iteration to a part of the problem breakdown.

Table 3.1: Development sub-tasks.

Number	Sub-task
1	Design home page
1.1	Design help page
1.2	Design login page
1.3	Document the backend code throughout
2	Allow users to create their account
3	Implement login system with password resets
...	...
A	Deploy website

Finally note that all the code for this project is open source and freely available for you to view on github: `https://github.com/zzzNathan/Video-Conferencer`.

We may shorten code snippets using ellipsis though the entire code at such point in development is available by finding the correct commit on Github. This commit number as well a link are both provided at the end of each iteration.

§3.2 Iteration 1

During this iteration I wanted to finish up designs for a few of the pages that will appear on our website. This should clear up sub-task 1 and 1.2.

Home page

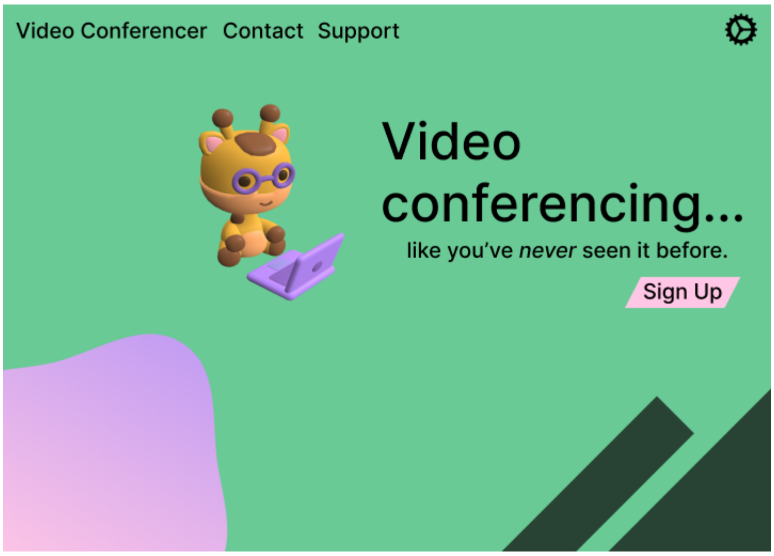


Figure 3.1: 2nd Mock up of the home page.

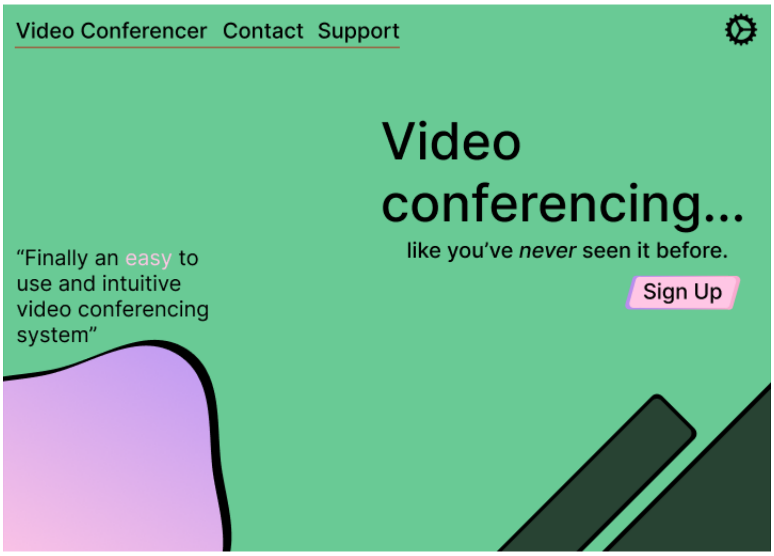


Figure 3.2: 3rd Mock up of the home page.

Description: I started by cleaning up the mock design for our home page. The first mock up was a very basic outline for how I wanted to style the page, so in the next couple iterations I polished the design and added more designs to fill the web page. These additions made the home page feel more complete and less bare-bones, consequently making the webpage feel more professional for the user.

Explanation and justification: I created mock ups of the home page and all other web pages before actually coding them up. Designing mock ups helped to give me a sense of direction in the process of creating a webpage design in HTML and CSS. If I were to try to create the design on the fly whilst writing code I would have no reference to use and my design would end up being lackluster and poorly thought out. By purely focusing on the page layout and design first, I can create a page that I am happy with in Figma and then focus on converting this design into HTML and CSS code using my mock ups as a reference as to what the end product should look like. Indeed by starting with designing mock ups, it gives me something to aim for when writing up the code to produce our home page. Then by iteratively improving upon the design mock ups I created I enhance the look of the final home page design, in turn creating a superior user experience.

Description: I transfer my designs into HTML and CSS. However after some reseach I come to the conclusion that I'd prefer to write sass and compile this into CSS instead, because of it's superior syntax.

Explanation and justification: The next step was to transfer this design into HTML and CSS. Since my designs made use of a collection of frequently used colours I concluded that it would be useful to be able to use these colours as defined variables instead of having to write out their hex colour codes each time (see section 1.7.5). However the native CSS syntax looks something like this. (Taken from `https://www.w3schools.com/css/css3_variables.asp`)

```
:root {
  --blue: #1e90ff;
  --white: #ffffff;
}
```

I personally found this syntax particularly disgusting, after doing some research I came across this stack overflow answer `https://stackoverflow.com/a/1877358`. After digging into exactly what "sass" and "less" were I decided to use sass as an extension of my CSS because I liked the look of it's syntax better. Sass (**S**yntactically **A**wesome **S**tylesheet) is a CSS extension language that provides a new syntax for writing CSS as well as a number of features that prevent repetition in the code like variables, functions and more. The sass code is saved in a `.sass` file then one can compile the sass code into CSS by using the command `sass <sass file name> <output css file name>`.

After blocking in the basic elements of our webpage the HTML and sass code looked like this.

Prototype 1

main.html

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href='https://fonts.googleapis.com/css?family=Inter' rel='stylesheet'>
  <link rel="stylesheet" href="styles.css">
  <title>Video-Conferencer</title>
</head>

<body>
  <div class="Headlines">

    <div class="Main_Headline">
      Video conferencing...
    </div>

    <div class="Sub_Headline">
      like you've never seen before.
    </div>

  </div>
</body>

</html>
```

styles.sass

```
// Colour definitions
$Col_Main:       #69ca95
$Col_Secondary:  #284333
```

```
$Col_Tertiary:   #9a6442
$Col_Accent:     #ffc6e5
$Col_AccentDark: #c49df2

// Setting font and background colour
body
  background-color: $Col_Main
  font-family:      'Inter', sans-serif

// Styling for the main headline
.Main_Headline
  width:        701px
  height:       238px
  font-size:    96px
  ...

// Styling for the sub headline
.Sub_Headline
  width:        603px
  height:       101px
  font-weight: 500
  ...
```

This code produced our 1st prototype.

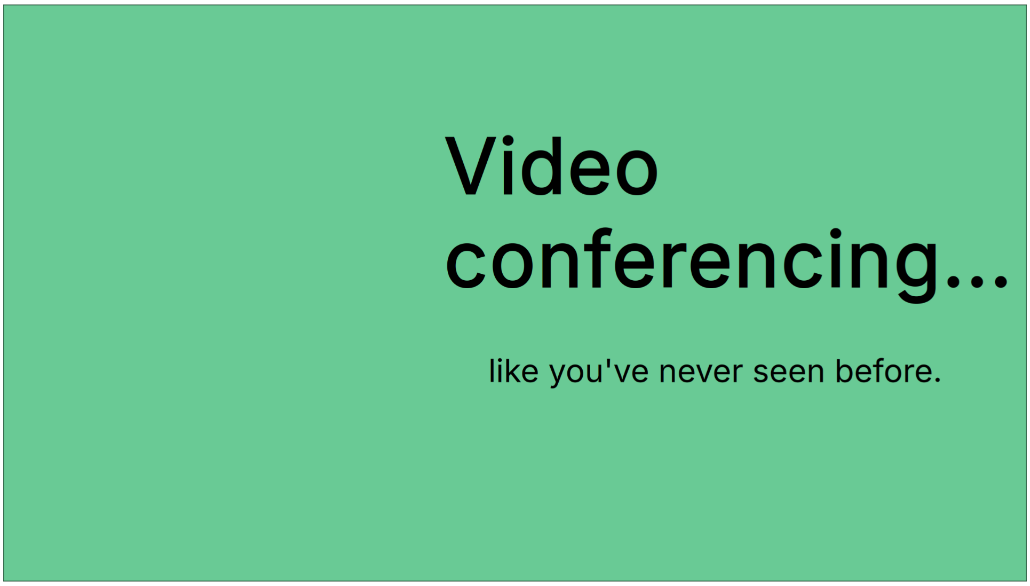


Figure 3.3: Home page prototype 1

Client Feedback:

- The headline and sub-headline are too far apart.
- The main headline takes up too much space on the page.
- The "never" isn't italicised.

Prototype 2

Description: I pivot to using vector graphics in order to position the elements on our webpage.

Explanation and justification: After writing the code for the first prototype I soon realised that actually writing the text for the main headlines and manually positioning it would be a very tedious task. So I decided to instead export the headline as well as the graphic designs as `.svg` files from Figma. This means I wouldn't have to worry about carefully positioning and spacing the headlines individually and I could instead just position them as 1 vector image. The `.svg` image format was chosen because vector images never lose their resolution even when scaled, you will still be able to see clean and crisp images no matter how much you zoom in, something which isn't achievable with the `.jpg` or the `.png` formats. In the rare case that `.svg` files are not supported for the user's browser [1] we also provide fallback `.png` image files for these users. This ensures that users will be able to see the page's content no matter their choice of browser, hence improving the robustness of our system.

Our code now looks like this.
main.html

```html
...

<body>

  <div id="Headline_container">
    <object id="Headline" data="Images/Headline_text.svg" type="image/svg+xml">
      <img src="Images/Headline_text.png">
    </object>

    <object id="Sign_up" data="Images/Sign_up.svg" type="image/svg+xml">
      <img src="Images/Sign_up.png">
    </object>
  </div>

  <div id="Bot_left_graphic_container">
    <object id="Bot_left_graphic" data="Images/Bot_left_graphic.svg" type="image/svg+xml">
      <img src="Images/Bot_left_graphic.png">
    </object>
  </div>

  <div id="Bot_right_graphic_container">
    <object id="Bot_right_graphic" data="Images/Bot_right_graphic.svg" type="image/svg+xml">
      <img src="Images/Bot_right_graphic.png">
    </object>
  </div>

</body>

</html>
```

styles.sass

```sass
...

// Setting font and background colour
body
  height:           100vh
  margin:           0
  background-color: $Col_Main
  ...

// Positioning the headline svg
#Headline_container
  position: relative
  top:      10%
```

[1] Vector images are supported by all browser except Internet Explorer 8 and below, and Android 2.3 browser and below.

```
  left:     45%

// Positioning the signup svg
#Sign_up
  position: absolute
  top:      75%
  left:     35%

// Positioning the bottom left graphic svg
#Bot_left_graphic_container
  position: relative
  top:      0%

// Positioning the bottom right graphic svg
#Bot_right_graphic_container
  position: relative
  left:     50%
  bottom:   75%
```

This code produced our 2nd prototype.

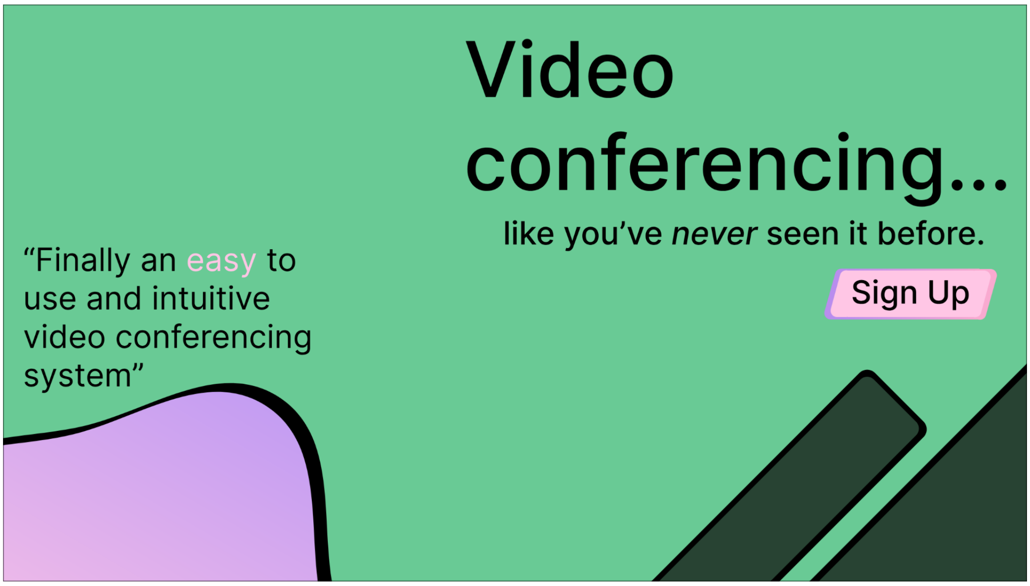


Figure 3.4: Home page prototype 2

Client feedback:

- We are still lacking a navigation bar at the top.
- All the elements feel as though they take up too much space on the page.

Prototype 3

Description: I redesign the entire look and aesthetic of our system.

Explanation and justification: After reviewing the design holistically, I felt that I could have done a better job designing the page to contribute to creating a professional looking system, so I re-did my designs in Figma. The graphic designs on our previous prototype designs were nice but they felt "too much", so I decided I should tone back the designs and focus on simplicity. Indeed our target user base is comprised of elderly ones who might not appreciate such a bright and vibrant design, the design would have not only been overwhelming for the average user but also would have made our system seem unprofessional and childish. This is seen in the complete lack of cohesion between the various graphics on that page. The dark green triangle graphic creates an unwanted contrast between it and the rest of the page to the point that it becomes the centre of attention and one of the first things that will catch the user's eye. Furthermore the choice for the background colour was too bright and vibrant and didn't allow other elements to stand out as much as they should.

In designing the revised prototype I took inspiration from the design of the Zen browser home-page `https://zen-browser.app/`. They simply have a navigation bar, a main and sub headline, a call to action button and some testimonials along the bottom. Here is the newest design. (Of course after making such significant changes I contacted my client and asked for permission to change to this new design and he agreed).

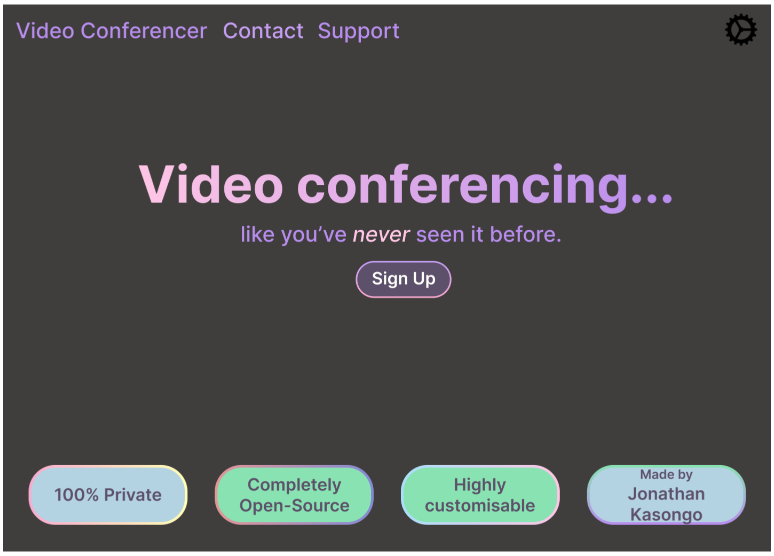


Figure 3.5: 4th Mock up of the home page.

Now in relation to the point *"the elements feel as though they take up too much space on the page"* I found that an appropriate solution was to simply scale every element on our page to 75%. This then made the page look closer to the one design in Figma. Whilst writing the HTML and sass code I made some minor design changes based on what looked the most appealing. The bottom bar was dropped from the final design because it felt tacky compared to rest of the website. I instead replaced it with the quote that sat on the bottom left of our original design (look at prototype 2). Finally I gave the settings icon the same purple colour as the rest of the navigation bar. Here is the code of the 3rd prototype.

main.html

```
  ...
  <link rel="stylesheet" href=
    "https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  ...
<body>

  <ul id="Navbar">

    <li id="Logo_container">
      <object id="Logo" data="Images/Logo.svg" type="image/svg+xml">
        <img src="Images/Logo.png">
      </object>
    </li>

    ...

    <li id="Settings">
```

```
      <i class="glyphicon glyphicon-cog"></i>
    </li>
  </ul>

  <div id="Main">
    <object id="Headline" data="Images/Headline.svg" type="image/svg+xml">
      <img src="Images/Headline.png">
    </object>

    <object id="Sign_up" data="Images/Sign_up.svg" type="image/svg+xml">
      <img src="Images/Sign_up.png">
    </object>
  </div>

  <object id="Quote" data="Images/Quote.svg" type="image/svg+xml">
    <img src="Images/Quote.png">
  </object>

</body>

</html>
```

styles.sass

```
// Colour definitions
...

// Setting font and background colour
body
  height:           100vh
  margin:           0
  background-color: $Col_Main
  ...

// Positioning the headline
#Headline
  display:      block
  margin-left:  auto
  margin-right: auto
  ...

// Scaling the sign up button
#Sign_up
  margin-top: -3vh
  scale:      75%

#Navbar
  list-style-type: none
  margin:          0
  padding:         0
  ...

// Styling the navbar elements
#Settings
  float: right
  scale: 300%
  color: $Col_Secondary
  ...
```

```

#Contact_container
  margin-right: 0%
  scale:        75%
  padding:      2vh

#Logo_container
  margin-left:  -3%
  margin-right: -5%
  scale:        75%
  ...

#Support_container
  margin-left:  -3%
  margin-right: 45%
  scale:        75%
  ...

// Positioning the call to action button
#Main
  display:         flex
  align-items:     center
  justify-content: center
  flex-direction:  column

// Positioning the quote
#Quote
  position: relative
  left:     0%
  top:      20%
  ...
```

This code produced our 3rd prototype.

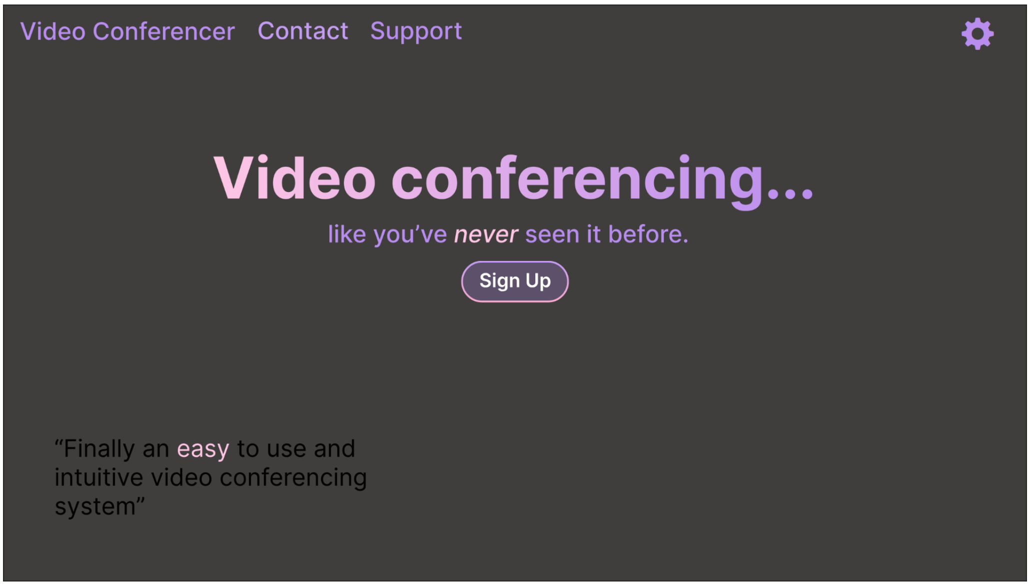


Figure 3.6: Home page prototype 3

Client feedback:

- Everything looks great!

Refinements

Description: I reduce the usage of vector graphics in our design, add a gradient to the background, add a new navigation tab and more. I conclude by cleaning up some of our CSS code.

Explanation and justification: Not much has changed in terms of the UI of our home page. I added a soft gradient to out previously plain gray background as well as a new navigation tab "Docs". This is where users will click in order to read the documentation for our back end. I added in 2 more quotes to the bottom of the page, in a card format using a `.svg` graphic. I also changed the navbar to make use of text instead of `.svg` images because they are easier to align and will help the page load faster aswell. Then as a nice touch I added a simple hover transformation to the sign up button that makes the whole button a gradient. Finally I cleaned up the code and made more use of CSS's viewport and `margin: auto` properties, since this would ensure that our users would have the same UI experience no matter the size of their device. Here is what the refined code looks like:

main.html

```
...
<body>

  <ul id="Navbar">

    <li class="Navbar_item">
      Video-Conferencer
    </li>

    ...

    <li class="Navbar_item">
      Docs
    </li>

    <li class="Last_Navbar_item">
      <i class="glyphicon glyphicon-cog"></i>
    </li>
  </ul>

  <div id="Main">

    <object id="Headline" data="Images/Headline.svg" type="image/svg+xml">
      <img src="Images/Headline.png">
    </object>

    <button class="Sign_up"> <b> Sign up </b> </button>

  </div>

  <object id="Quote" data="Images/Quote_cards.svg" type="image/svg+xml">
    <img src="Images/Quote_cards.png">
  </object>

</body>

</html>
```

style_main.sass

```
@import url('https://fonts.googleapis.com/css?family=Inter')
@import url('https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css')
```

```

// Colour definitions
...

// Setting font and background colour
body
  height:           100vh
  margin:           0
  background-image: linear-gradient($Col_Main, $Col_Gradient)
  ...

// Positioning the headline
#Headline
  display:      block
  margin-left:  auto
  margin-right: auto
  ...

// Styling the sign up button
.Sign_up
  border-radius:    700px
  color:            white
  background-color: $Col_Button_BG
  border:           none
  ...

// Creates border and hover effect
.Sign_up::after
  content:          ''
  position:         absolute
  background-image: linear-gradient(to bottom, $Col_Button_Top, $Col_Button_Bot)
  z-index:          -1
  ...

.Sign_up:hover
  z-index: 0

// Styling the navigation bar
#Navbar
  list-style-type: none
  margin:          0
  padding:         0
  ...

.Navbar_item
  color:       $Col_Secondary
  padding:     2vh
  font-size:   2.3vw
  white-space: nowrap

.Last_Navbar_item
  @extend      .Navbar_item
  margin-left: auto

// Positioning the call to action button
#Main
  display:         flex
  align-items:     center
```

```
  justify-content: center
  ...

// Positioning the quote
#Quote
  position:    fixed
  bottom:      -27px
  left:        50%
  ...
```

This code produced this page:

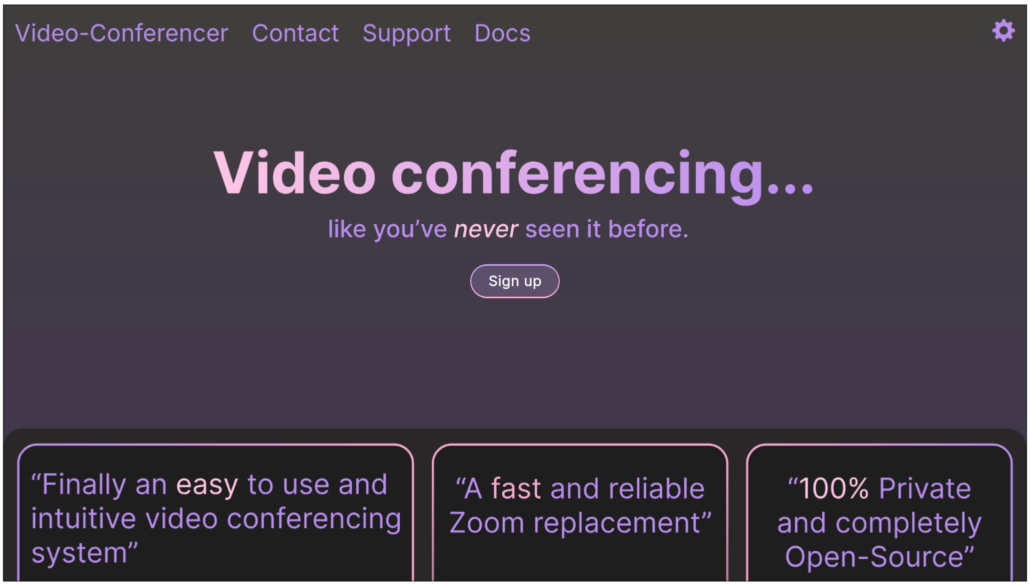


Figure 3.7: Refined home page.

Development sub-task (1) ✔

Login page

Description: I re-design the login page to fit with the new theme, and implement my design in HTML and sass.

Explanation and justification: Clearly the login page needed to be re-designed also in order to properly match the theme of the home page. Here is the revised design for our login page. Inspiration was drawn from Spline's login UI https://spline.design/.

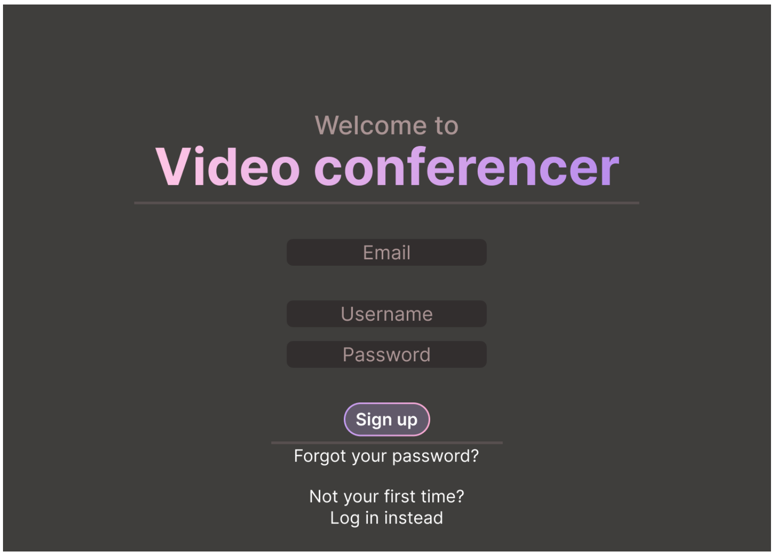


Figure 3.8: 2nd Mock up of the login page.

This revised version fits much nicer with the home page. During the creation of this page however I realised that I had not yet discussed what we would do if a user had forgotten their password. In such a case we could employ the standard technique of sending the user a reset password link to their email address and I have included this in our mock up. This idea will have to be added as a development sub-task and we will tackle it's implementation in a future iteration.

The next step is to transfer the design in code, then link our home page and login page together. During the programming of the home page I realised that I would be using the same sign-up button design as the one designed on the home page. Recalling that I would apply the DRY ("**D**on't **r**epeat **y**ourself") principle whilst I wrote my code (see section 1.2), I decided to put the button code in it's own file `button_styling.sass` and simply have the other `.sass` files import from the button styling file whenever needed. Now that I had gained some experience with translating my designs into HTML and CSS I was able to make some changes to the design on the fly to improve upon the figma mock up design.

login.html

```
...
<body>

  <object id="Headline" data="Images/Welcome.svg" type="image/svg+xml">
    <img src="Images/Welcome.png">
  </object>

  <center>
    <b id="Login_Prompt"> Log in instead? Click here </b>
  </center>

  <br><br>

  <center>

    <div id="Login">

      <form>
        <input type="text" placeholder="Email" name="email" required>       <br><br><br>
        <input type="text" placeholder="Username" name="username" required> <br><br>
        <input type="password" placeholder="Password" name="password" required autofocus>
        <br> <br> <br>

```

```
        <button class="Button"> <b> Sign up </b> </button> <br><br>

        <div class="Rule"> </div> <br>

        <p style="color:white; font-size: 20px"> Forgot your password? </p>
      </form>
    ...
```

styles_login.sass

```
@use './button_styling.sass'
@import url('https://fonts.googleapis.com/css?family=Inter')
@import url('https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css')

// Colour definitions
...

// Setting font and background colour
body
  height:           100vh
  margin:           0
  background-color: $Col_Main
  ...

.Rule
  width: 40%
  border-bottom: 4px solid $Col_Rule

// Positioning the headline
#Headline
  display:      block
  margin-left:  auto
  margin-right: auto
  ...

// Positioning the login box
#Login
  disply:       block
  margin-left:  auto
  margin-right: auto

// Styling the input field
input
  border-radius:    12px
  width:            281px
  height:           37.5px
  ...

#Login_Prompt
  color:     white
  font-size: 20px
```

This code produced our first prototype:

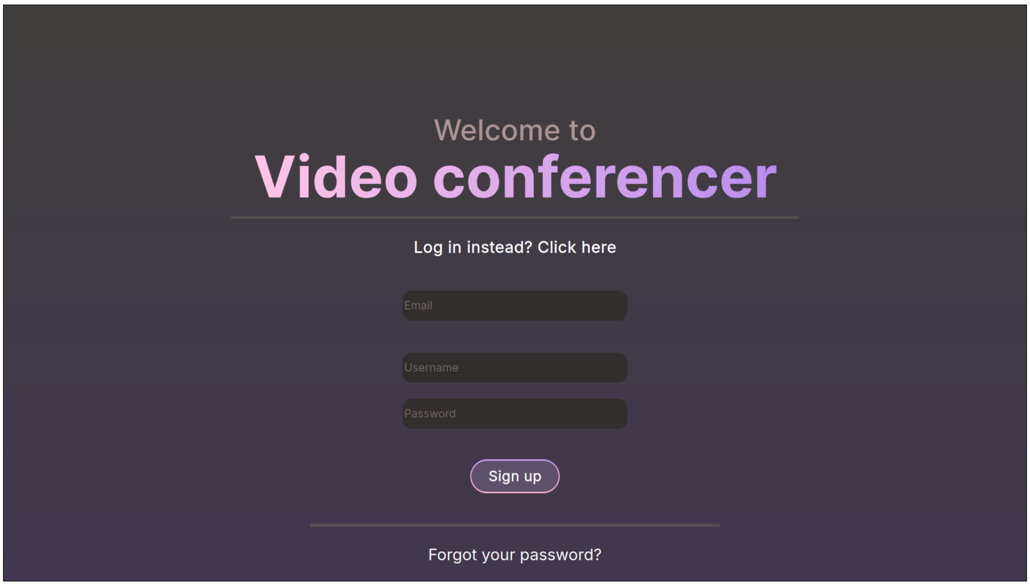


Figure 3.9: Login UI prototype 1

Client feedback:

- Looks great!

Development sub-task (1.2) ✔

The current file structure of our project is as follows:

```
src
├── Images
├── main.html
├── login.html
├── styles_main.sass
├── styles_login.sass
├── styles_main.css
├── styles_login.css
└── button_styling.sass
```

Look at Github commit `54e06ac60f0876086c259f29ce23eae89ff4aad6` to see the state of the project after our first iteration.

Alternatively click the following link to view the state of the project at this point in development: `VideoConferencer-Iteration1`.

§3.3 Iteration 2

During this iteration I wanted to design and implement some of the underlying infastructure behind our application and complete development subtasks (2) and (A).

Description: After doing some of research into free web deployment services I settled on using Vercel in order to deploy my website.

Explanation and justification: Here are a few of the other options I looked into and why I rejected them.

Service	Explanation
Railway.app	You can only use $5.00 worth of resources on the free tier, once this is depleted the project won't be up anymore
render.com	Projects on the free tier are deleted automatically after 1 month

On the other hand Vercel doesn't have any of these drawbacks, so my site will be able to stay up permanently so long as some resource limits are not crossed.

Description: In order to use Vercel to deploy my web application I had to use a web framework. I chose to use React with Vite and rewrote my existing codebase to work with this framework.

Explanation and justification: Initially I intended to try and stay away from all of these web-frameworks and just stick to plain old HTML and CSS (or sass in our case), however an issue arose when I wanted to redirect users to our registration page once they clicked the sign up button on our home page. For whatever reason the redirection would work when I hosted the application on my own machine, however when I would deploy to vercel the redirection would either, simply not change page or display a 404 code not found error.

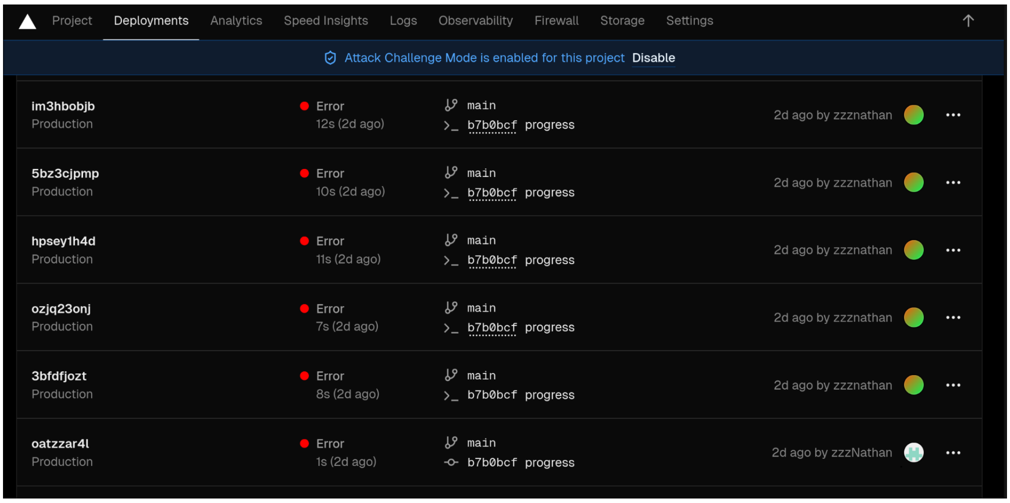


Figure 3.10: Errors when deploying to Vercel.

Eventually after a lot of trial and error and some research, I found out that using a router with react would allow us to redirect the user to different pages once deployed. I then took some time to migrate our codebase to JSX to work with the react framework. We will omit the code for the re-written webpages since it is pretty similar to the snippets shown in Iteration 1. Here is what the routing logic looks like.

main.jsx

```
import { createRoot } from "react-dom/client"
import { createBrowserRouter, RouterProvider } from "react-router-dom"
import Home from "./components/Home.jsx"
import Registration from "./components/Registration.jsx"

// Path is an extension that goes after our URL,
// once this extension is written the corresponding
// React element will be loaded and rendered.
const router = createBrowserRouter([
  {
    path: "/",
    element: <Home />
  },
  {
```

```
    path: "/registration",
    element: <Registration />
  }
])

createRoot(document.getElementById('root')).render(
  <RouterProvider router={router} />
)
```

Description: I used the Clerk user management platform to create a user account registration form.

Explanation and justification: I originally wanted to do this task manually, creating my own table and then querying this table in order to add new users and log users in. However after doing some reading and looking over Stack Overflow posts like this one https://stackoverflow.com/questions/46819734/how-to-check-username-and-password-matches-the-database-values I realised that many issues can arise when developers try to implement these systems themselves. In order to avoid this plethora of issues I instead chose to use the Clerk.com API [2] for user authentication. Not only was it much easier to implement but the usage of this API also ensures that I wont have to worry about any security issues, since the API has been thoroughly tested. Here's the implementation.

Registration.jsx

```
import { ReactTyped } from "react-typed"
import { ClerkProvider, SignedOut, SignUp } from "@clerk/clerk-react"
import { dark } from "@clerk/themes"
import Navbar from "./Navbar"
import "../styles/Registration.sass"

const PUBLISHABLE_KEY = import.meta.env.VITE_CLERK_PUBLISHABLE_KEY
...

function Registration () {
  return (
  <> <Navbar />
    <Headline />

    <ClerkProvider publishableKey={PUBLISHABLE_KEY}>
      <SignedOut>
        <center> <SignUp appearance={dark}/> </center>
      </SignedOut>
    </ClerkProvider>
  </>
  )
}

export default Registration
```

Here is what the code renders to:

[2] I originally came across this API watching Theo T3.gg's React tutorial.

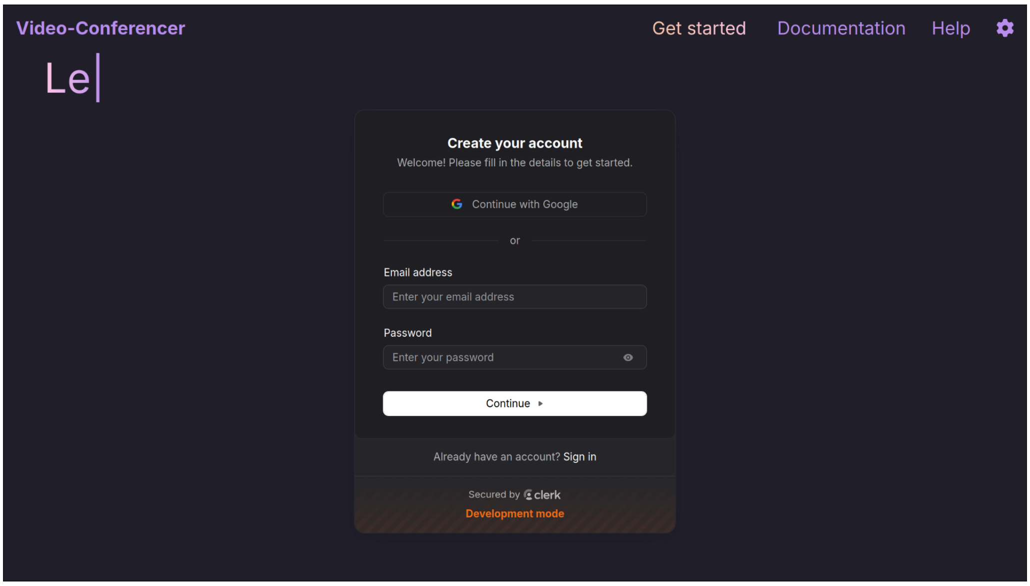


Figure 3.11: Rendering the registration page.

Tests:

- Site home page loads correctly ✔
- Correctly redirected to registration page ✔
- Users can create a new account ✖

Evidence: https://youtu.be/tHUQeArAv-g

The reason why the registration page wasn't rendering was because I made use of Clerk's `useUser()` function in my `<Navbar />` component outside of a ClerkProvider component. As per the documentation *"The `<ClerkProvider>` component is required to integrate Clerk into your React application, providing session and user context to Clerk's hooks and components."* [3] This makes sense since the Clerk SDK would need to have context about the current user logged in order to get the information returned in the `useUser()` hook, like the `user.id` and the `user.username` .

This issue was fixed by simply wrapping my HTML code in the `<ClerkProvider>` component whenever a page made usage of the `<Navbar />` . All the tests now successfully pass.

Tests:

- Site home page loads correctly ✔
- Correctly redirected to registration page ✔
- Users can create a new account ✔

Evidence: https://youtu.be/vpuVXP1-RTw

You can try these features out yourself at https://video-conferencer.vercel.app/registration. (May not work if development is ongoing).

Development sub-task (2) ✔

Development sub-task (A) ✔

The current file structure of our project is as follows: (Ignoring configuration files)

[3] Source: https://clerk.com/docs/components/clerk-provider

```
Video-Conferncer
├── index.html
└── src
    ├── main.jsx
    ├── assets
    │   └── Quote_Cards.svg
    ├── components
    │   ├── Button.jsx
    │   ├── Home.jsx
    │   ├── Navbar.jsx
    │   └── Registration.jsx
    └── styles
        ├── Button.sass
        ├── Home.sass
        ├── Navbar.sass
        └── Registration.sass
```

Look at Github commit `243560c390c019ab39903850ef3dba99ed98df18` to see the state of the project after our first iteration.

Alternatively click the following link to view the state of the project at this point in development: `VideoConferencer-Iteration2`.

¶3.3.1 Iteration 3

During this iteration I wanted to further develop the backend infastructure of our application, and start to sketch out the actual implementation for including video conferencing in our application. I also improve the code and touch up some of the designs, and use Clerk to rewrite the login page.

Description: I rewrite the login page using clerk, and make minor improvements to our landing page.

Explanation and justification: Our users can now register an account with Clerk, so they will also have to login using the Clerk SDK aswell. During this time I also improved the landing page design. These improvements contribute to the professional look and feel of our website and enhance the user experience.

Login.jsx

```
...
// Makes the typing headline animation
function Headline ()
{
  return (
    <ReactTyped
      className={"Headline"}
      strings={[
        "Welcome back :)",
        "Log back in here,",
      ]}
      typeSpeed={120}
      startDelay={30}
      loop
    >
    </ReactTyped>
  )
}

// Renders the login page
function Login () {
  return (
  <>
    <ClerkProvider publishableKey={PUBLISHABLE_KEY}>
      <Navbar />
      <Headline /> <br/>
      <SignedOut> <center>

        <SignIn
          signUpUrl="/registration"
          forceRedirectUrl="/home"
          appearance={{
            baseTheme: dark,
            variables: {spacingUnit: "2vh"}
          }}
        />

      </center> </SignedOut>
    </ClerkProvider>
  </>
  )
}

export default Login
```

Here's what the code renders to:

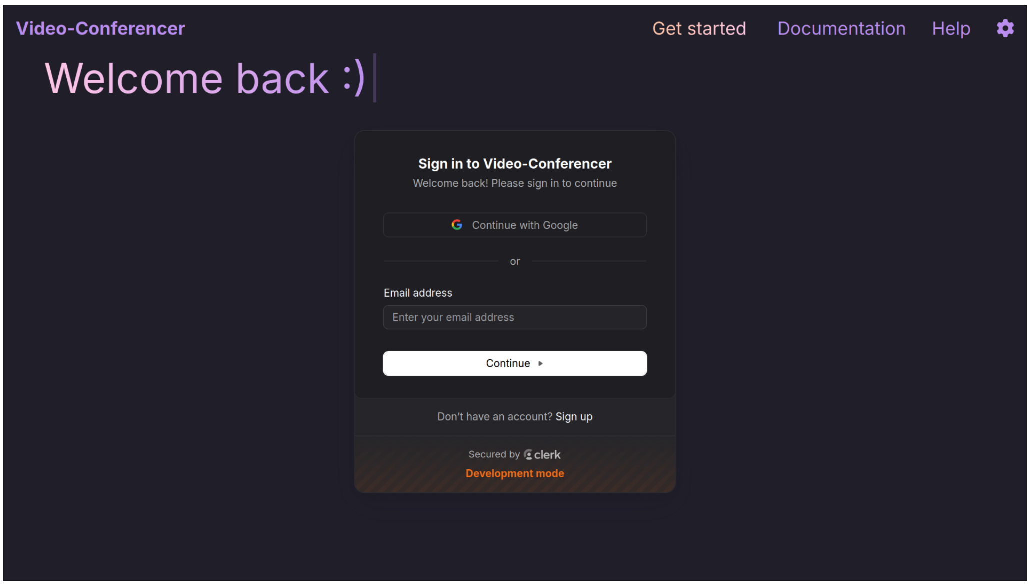


Figure 3.12: Rendering the login page.

Description: I design and implement the page that users will see after logging in to their account.

Explanation and justification: In order to start, join video conferences and change video configuration settings, the user must have some way to be able to access these features. The UI should allow our users to easily navigate the site, providing a good user experience whilst still looking professional. I began by making a quick sketch of the design I had in mind.

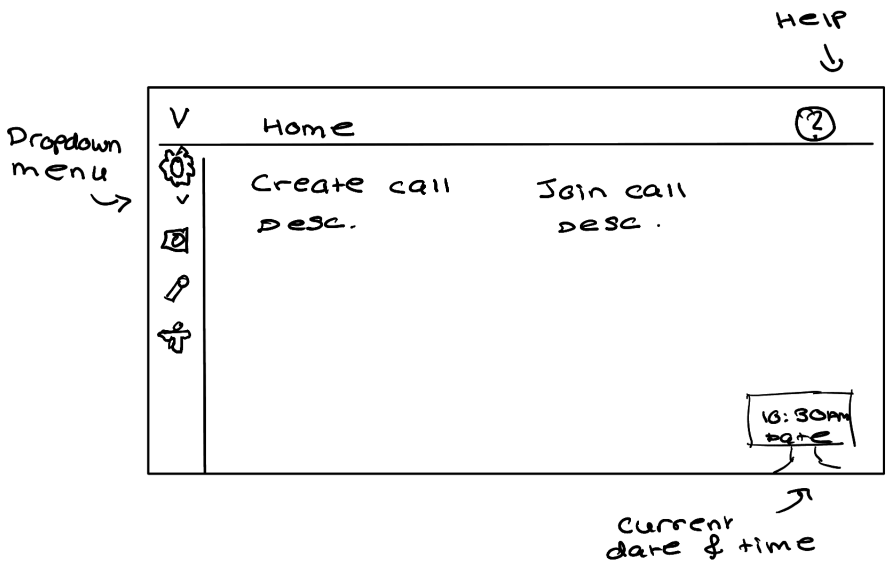


Figure 3.13: Sketching out the design.

From here it was pretty simple to write out the code to make this UI.

Confusingly from here on we call this page our home page and the page previous home page our landing page in our code base. In this document we will use the following names. We will call the page that the user first sees our 'home' page, we will call the page that the user first sees after logging in our 'main' page.

Home.jsx

```
import "../styles/Home.sass"
import Clock from "./Clock.jsx"
```

```
...

function Options () {
  const Create_Desc = <div className="Desc">
    Create a new video call, then invite others
    to join by giving them the call code given
    to you once you start the call.
  </div>

  const Join_Desc = <div className="Desc">
    Enter the code given to you to join the
    video call.
  </div>

  return (
    <ul className="Options">
      <li> Create call <br/> {Create_Desc} </li>
      <li> Join call   <br/> {Join_Desc}   </li>
    </ul>
  )
}

function Home () {
  return (
    <> <Top_Bar />
    <Side_Bar />
    <Options />

    <Clock />
    </>
  )
}

export default Home
```

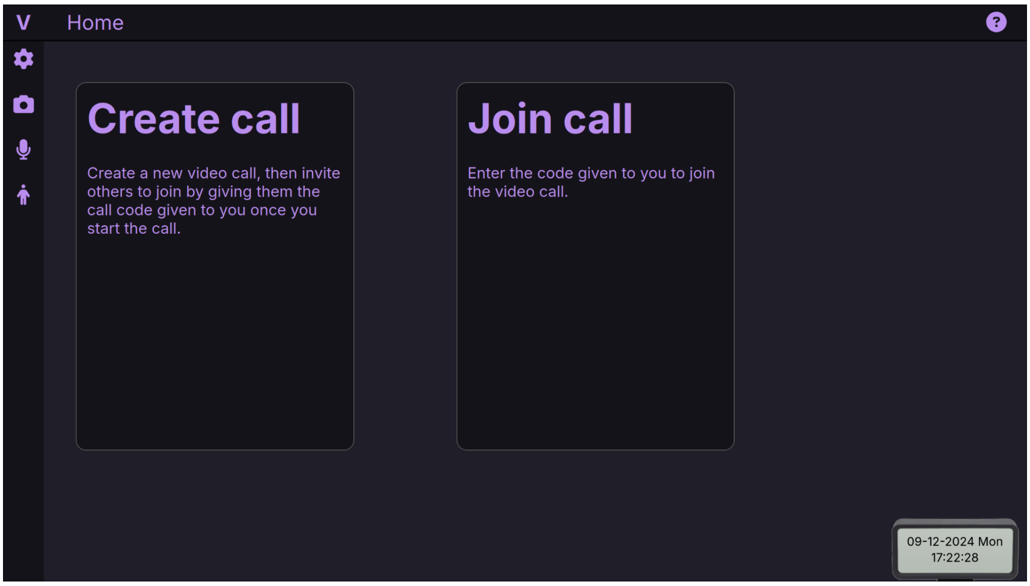


Figure 3.14: Implementing the code for the sketch.

Description: I chose to use the GetStream React library to implement video conferencing in my application.

Explanation: As detailed in my research, I first intended to use the WebRTC framework provided by Google to write the video conferencing logic. However after some research into what it would take for me to implement such a system I realised that actually implementing the logic would take me an unreasonable amount of time. Taking into consideration the fact that I would have to understand and correctly implement the sending of SDP protocol offers, connecting each user to a STUN server, whilst ensuring that each user has access to the other user's ICE candidates, the task felt as though it would take me a number of weeks to implement such a system to allow 2 users to video conference properly, let alone have n users video conference simaltaenously. So instead I chose to use a popular industry standard video confercing library called GetStream.

The first step to have users be able to video conference was to have our users be authenticated as users on the GetStream API. According to the documentation in order to do this each user must be given a unique JWT (JSON web token) token via GetStream's `client.generateUserToken()` function. In the documentation it explicitly says *"Tokens need to be generated server-side, ... You need to implement the token provider in your own application, this is usually an HTTP endpoint."* [4] In order to run the backend code I would need a server or some other place to be able to host my code. After reading through some forums online I came across Cloudflare's Worker service. This service allows users to be able to run server side code remotely through one of Cloudflare's many servers across the globe, with pretty generous free tier limits. I quickly set up an account with Cloudflare and began to implement my HTTP API.

/backend/stream-token-provider/src/index.js

```
...

// An HTTP endpoint that allows POST requests
// with a user id given in the request. Will return
// a unique user token that can be used to begin
// video conferencing
async function Provide_Token(request, env)
{
  const { method } = request;

  // Ensure that the request is a POST
  if (method !== 'POST')
    return new Response('Method not allowed', { status: 405 });

  try {
    const { User_Id } = await request.json();

    // Ensure that a valid user id is actually provided
    if (!User_Id || User_Id.length != USER_ID_LENGTH)
      return new Response('Bad Request: Proper userId is required', { status: 400 });

    const apiKey = env.STREAM_API_KEY;
    const secret = env.STREAM_API_SECRET;

    const token = await Generate_Token(User_Id, apiKey, secret);

    return new Response(JSON.stringify({ token }), {
      headers: { "Content-Type": "application/json" },
    });

  } catch (error) {
      return new Response(`Error: ${error.message}`, { status: 500 });
  }
}
...
```

Description: I fix a bug with button misalignment on mobile.

[4]Source: https://getstream.io/video/docs/api/authentication/

Explanation: Whilst testing out how my website works on different devices, I found that although the page looks fine on desktop, on mobile the buttons on the home page seem to be misaligned.

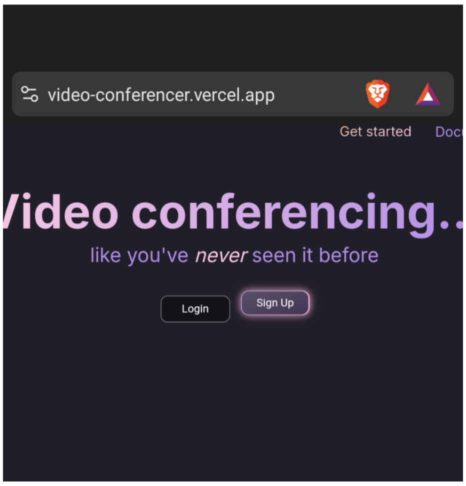


Figure 3.15: Buttons being misaligned on mobile

In order to fix this I simply changed the sass of the login button to use a pseudo element for it's border, so that it is created in the same way the sign up button is created. By doing it this way we can be 100% sure that both buttons will now be the same exact size, and that hence we will have no more alignment issues.

Landing.sass

```
...
.Login_Button
  border-radius:    0.9vw
  color:            white
  background-color: $Col_Black
  ...

.Login_Button::after
  content:          ''
  position:         absolute
  height:           107%
  ...

.Login_Button:hover
  scale: 104%
...
```

These changes fixed our issues with button misalignment.

Code listings

§A.1 Front end

/src/main.jsx

```
import { createRoot } from "react-dom/client"
import { createBrowserRouter, RouterProvider } from "react-router-dom"
import Landing from "./components/Landing.jsx"
import Registration from "./components/Registration.jsx"
import Login from "./components/Login.jsx"
import Home from "./components/Home.jsx"

// Path is an extension that goes after our URL,
// once this extension is written the corresponding
// React element will be loaded and rendered.
const router = createBrowserRouter([
  {
    path: "/",
    element: <Landing />
  },
  {
    path: "/registration",
    element: <Registration />
  },
  {
    path: "/login",
    element: <Login />
  },
  {
    path: "/home",
    element: <Home />
  }
])

createRoot(document.getElementById('root')).render(
  <RouterProvider router={router} />
)
```

/src/Button.jsx

```
import "../styles/Button.sass"

// Simple sign up button component
function Button()
{
  return (
    <button class="Button"> Sign Up </button>
  )
}

export default Button
```

/src/Clock.jsx

```
import "../styles/Clock.sass"
import { useEffect, useState } from "react"

// Gets current date and time and continously updates it every second
function Clock ()
{
  const [currentDateTime, setCurrentDateTime] = useState("")

  const updateDateTime = () => {
    const now = new Date()

    // Get components of the date
    const day    = String(now.getDate()).padStart(2, "0")
    const month  = String(now.getMonth() + 1).padStart(2, "0")
    const year   = now.getFullYear()
    const hour   = String(now.getHours()).padStart(2, '0')
    const minute = String(now.getMinutes()).padStart(2, '0')
    const second = String(now.getSeconds()).padStart(2, '0')

    // Get the day name
    const options = { weekday: "long" }
    const dayName = now.toLocaleDateString("en-US", options).substring(0,3)

    // Format the date string
    setCurrentDateTime(`${day}-${month}-${year} ${dayName}\n
                        ${hour}:${minute}:${second}`)
  };

  useEffect(() => {
    updateDateTime()

    // Update every second
    const intervalId = setInterval(updateDateTime, 1000)

    // Cleanup interval on unmount
    return () => clearInterval(intervalId)
  }, []);

  // Toggle the opacity of the clock when the user clicks on it
  const [opacity, setOpacity] = useState(1)
  const Toggle_Opacity = () => {
    setOpacity(prevOpacity => (prevOpacity === 0.1 ? 1 : 0.1))
  }

  // Render the clock
  return (
    <div
      className="Clock"
      onClick={Toggle_Opacity}
      style={{opacity: opacity}}>

      {currentDateTime}

    </div>
  )
}
```

```
export default Clock
```

/src/Landing.jsx

```
import { ClerkProvider } from "@clerk/clerk-react"
import Button from "./Button"
import Navbar from "./Navbar"
import Quotes from "../assets/Quote_Cards.svg"
import "../styles/Landing.sass"

const PUBLISHABLE_KEY = import.meta.env.VITE_CLERK_PUBLISHABLE_KEY

// Code to render the landing page of our site
function Landing () {
  return (
    <>
    <ClerkProvider publishableKey={PUBLISHABLE_KEY}>
      <Navbar />

      <div className="Main_Graphic">
        <h1 className="Main_Headline"> Video conferencing... </h1>

        <h2 className="Sub_Headline"> like you've <i>
        <span className="Pink_Text"> never </span> </i> seen it before </h2>
      </div>

      <div className="Buttons">
        <a href="/login"><button className="Login_Button"> Login </button></a>
        <a href="/registration"><Button /></a>
      </div>

      <img className="Quote_Card" src={Quotes} />
    </ClerkProvider>
    </>
  )
}

export default Landing
```

/src/Navbar.jsx

```
import { useUser } from '@clerk/clerk-react'
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome"
import { faCog } from "@fortawesome/free-solid-svg-icons"
import "../styles/Navbar.sass"

// Makes the navigation bar design a component must be inside
// a clerk provider
function Navbar()
{
  // If user isn't signed in then they can be redirected
  // to the registration page
  const { isSignedIn } = useUser()
  var link = "/registration"
  var text = "Get started"

  // If user is signed in then they can go straight
  // to the home page
```

```
  if (isSignedIn)
  {
    link = "/home"
    text = "Home"
  }

  return (
    <nav class="Navbar">
      <ul class="Navbar_List">
        <li class="Navbar_Item"> <b>  Video-Conferencer </b> </li>
        <li class="Navbar_Item_Left"> <a href={link}> <span class="Gradient"> {text}
          </span> </a> </li>
        <li class="Navbar_Item">      Documentation          </li>
        <li class="Navbar_Item">      Help                   </li>
        <li class="Navbar_Item"> <FontAwesomeIcon icon={faCog} /> </li>
      </ul>
    </nav>
  )
}

export default Navbar
```

§A.2 Backend

/backend/stream-token-provider/index.js

```
import { StreamClient } from "@stream-io/node-sdk";

const USER_ID_LENGTH = 32

// Code to generate unique user GetStream token
async function Generate_Token(User_Id, Api_Key, Secret)
{
  const client = new StreamClient(Api_Key, Secret);

  // Create user
  const newUser = { id: User_Id };
  await client.upsertUsers([newUser]);

  // Generate token
  const token = client.generateUserToken({ user_id: User_Id });

  return token;
}

// An HTTP endpoint that allows POST requests
// with a user id given in the request. Will return
// a unique user token that can be used to begin
// video conferencing
async function Provide_Token(request, env)
{
  const { method } = request;

  // Ensure that the request is a POST
  if (method !== 'POST')
    return new Response('Method not allowed', { status: 405 });
```

```

  try {
    const { User_Id } = await request.json();

    // Ensure that a valid user id is actually provided
    if (!User_Id || User_Id.length != USER_ID_LENGTH)
      return new Response('Bad Request: Proper userId is required', { status: 400 });

    const apiKey = env.STREAM_API_KEY;
    const secret = env.STREAM_API_SECRET;

    const token = await Generate_Token(User_Id, apiKey, secret);

    return new Response(JSON.stringify({ token }), {
      headers: { "Content-Type": "application/json" },
    });

  } catch (error) {
      return new Response(`Error: ${error.message}`, { status: 500 });
  }
}

export default
{
  async fetch(request, env)
  {
    return Provide_Token(request, env);
  }
};
```

Bibliography

[1] Corinna M. Bauer et al. “Multimodal MR-imaging reveals large-scale structural and functional connectivity changes in profound early blindness”. In: *PLOS ONE* (2017). URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0173064.

[2] *Bitwarden*. Version 2024.6.3. URL: https://bitwarden.com/.

[3] Alexander Chernev, Ulf Böckenholt, and Joseph Goodman. “Choice overload: A conceptual review and meta-analysis”. In: *Journal of Consumer Psychology* 25 (2 2015). ISSN: 1057-7408. URL: https://myscp.onlinelibrary.wiley.com/doi/10.1016/j.jcps.2014.08.002.

[4] William Feller. *An Introduction to Probability Theory and Its Applications*. 3rd ed. Vol. 1. Wiley, 1968.

[5] “How Much Internet Does Video Conferencing Need?” In: *Internet Equity Initiative* (2020). URL: https://internetequity.cs.uchicago.edu/data-story/how-much-internet-does-video-conferencing-need/.

[6] Zane Ma et al. “Understanding the Mirai Botnet”. In: *26th USENIX Security Symposium* (2017). URL: https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-antonakakis.pdf.

[7] Fitzpatrick D Purves D Augustine GJ et al. *Neuroscience*. 2001. URL: https://www.ncbi.nlm.nih.gov/books/NBK10924/.

[8] *Regulation 2016/679 General Data Protection Regulation*. 2016. URL: https://gdpr-info.eu/.

[9] *RTP Payload Format for VP9*. IETF, 2021. URL: https://datatracker.ietf.org/doc/draft-ietf-payload-vp9/.

[10] *Skype*. Version 8.119.0.201. URL: https://www.skype.com/en/.

[11] *UK home broadband performance*. Tech. rep. Ofcom, 2023. URL: https://www.ofcom.org.uk/phones-and-broadband/coverage-and-speeds/home-broadband-performance-march-2023/.

[12] Nicole VanKim and Toben Nelson. “Vigorous Physical Activity, Mental Health, Perceived Stress, and Socializing Among College Students”. In: *American Journal of Health Promotion* (2013).

[13] Daniel Lowe Wheeler. “zxcvbn: Low-Budget Password Strength Estimation”. In: *25th USENIX Security Symposium* (2016). URL: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_wheeler.pdf.

[14] *Zoom*. Version 6.1.1. URL: https://zoom.us/.