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

Section 1

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 realtime. 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 so on. 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 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 opputunities 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 oppurtunities 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 aswell. 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 resedentials 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 [8]. 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"* [4]. Even when taking the worst case of 2

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

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 feasability 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 negligable 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 videoconference 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 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 codebase 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 guarentees clear and concise code. One feature of the problem that could benefit from pattern recognition is the username and passcode system. A detailed justificaiton for this feature is provided in section 1.5 under the subheading *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 passcode to enter a video conference. Instead of writing 2 seperate 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_passcode()` 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 codebases.

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. 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 [1]. When humans have too many options we become unable to make descisions, 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 appropiately 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 convienient 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. Algorithmic thinking is not just convienient 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. If 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.

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 videoconferences. The new system would help the videoconferences 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.

Why is it appropiate to their needs? The proposed solution would be appropiate for my client as it would provide a smoother and simpler system in order to host videoconferences 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 appropiate 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 appropiate to their needs.

How will they use it? These users will make use of the proposed solution by joining videoconferences that they were invited to and interacting with them in a manner that is convienient for them. They will be able to interact with the videoconference 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.

Why is it appropiate 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 oppurtunity to improve their social lifes as they will be able to chat withtheir loved ones and friends over their web browsers easily. Not only will the final solution make their lives more convinient but it will also provide the user with the oppurtunity to improve their mental health through socialising and engaging in conversation with others. [5]

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.

Why is it appropiate to their needs? The solution will be appropiate 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

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.

Continued on next page

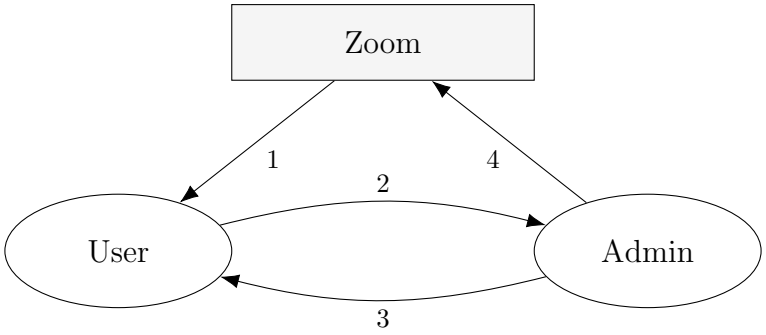

Figure 1.1: Zoom DFD diagram.

Table 1.1: Explanation of DFD. (Continued)

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 videoconference when appropiate. Admin can also interact with the app in order to manage the video conference. They can do things like spotlight speakers, screenshare images/video and mute or kick participants.

1.4 Research

We begin our research by observing solutions to similar problems. Then justifications of suitable approaches are given based on the existing solutions.

1.4.1 Zoom

Accessibility features

Zoom is a popular closed-source video chat application, developed by Zoom video communications. From their website [10] 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

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.

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 profiency in spoken english.

Zoom proposes a solution to the problem of exculsivity 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 descision 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.

1.4.2 Skype

Accessibility features

Skype is a proprietary messaging and video chat application developed by Skype technologies a subsidiary of Microsoft [7]. 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

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.

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."* [2] 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.

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 qualititative 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.

1.4.3 Interview

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 accidently 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?
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 inter-

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

act 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 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. Furthermore I believe that security should also be very important when designing the final solution. This should be a requirement because we don't want any of our users to have their important data stolen during their video conference.

Bearing this in mind we should also think about suitable approaches to implementing a system with these features. 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. 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 VP9 packetizers [6] 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. 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 compnay uses *"AES-CBC 256 bit encryption for the data encryption and PBKDF2 SHA-256 to derive one's encryption key."* [3] 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. Fortunately for our case we will not have to consider implementing these algotrithms because WebRTC provides built-in encryption.

Another concern pertaining to passwords is our method for checking a potential password's strength. I propose the usage of the `zxcvbn` algorithm [9] 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 codebase 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. [4] `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 substrings of the input password and classifies each substring 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 substrings with a frequency ranked dictionary, if there are symbols in p it creates extra strings making leet substitutions [5] 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 substring. Finally the last phase, the search looks for an optimal sequence s of length $|s|$, from a set S of all the matches found in p such that: the sequence contiguously covers the entirety of p and this sequence s

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

[4] Note that Daniel Wheeler explains his algorithm at a much more in-depth level in his paper, which may be found in the citation above, if you wish to learn more.

[5] Leet substutions are symbols that are used to replace characters like the word `@pp1e` → `apple`.

has a minimal number of guesses m_G over all matches in the sequence m. More formally the algorithm computes

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

with dynamic programming, 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. Having now seen how the algorithm works the advantages of using this model are numerous. It implements 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.

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 prioritse 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. Lastly 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.

1.5 Essential features & their justifications

Real-time audio/video feeds

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. 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.

Raising one's virtual hand

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 appropiate. 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.

Designation of a host

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. 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 in other video chat applications more specifically, on Zoom (e.g. Zoombombing), and in order to provide the best experience for our users it is vital that events like this aren't allowed to occur.

Usernames and passwords to join a call

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 passcode that they will be able to share with anyone else whom they would like to invite to the video conference. The justification of this feature is clearly sufficient because it was explicitly requested during my interview with the client. Moreover without a passcode 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 that was discussed in the previous paragraph.

Simple and user friendly UI

The UI should be designed in such a way that anyone could be able to understand how to use the application. This feature aligns with the idea to have the system be accessible, through ensuring that all users are able to navigate the interface without asking 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. This feature is fully justified since my client requested this as an "essential" feature during our interview in section 1.4.

1.6 Limitations

Limitation 1:

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. 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. [6] 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, alienating 34% of the world's population as they will be completely unable to use the system.

Limitation 2:

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. 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 userbase. 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.

Limitation 3:

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. 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 scalabilty of the system would be limited. After a certain amount of data is transfered 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 limita-

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

tion 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.

Limitation 4:
If this system is to be an actual web app then I would have to host it on a dedicated server, then purchase a domain name for my web app. The dedicated server would host the files needed for the web app to run, and the domain name would have to be purchased from a domain registrar so that whenever people type this domain name they are directed to the IP address of your server. Unfortunately both of these things require money to be spent whilst my client would prefer to have spent no money on this system. If I really want to fulfill the client's requests then I would have to use localhost in order to host the website, and this may cause complications in the future, like where should the database be hosted? These concerns fully justify this limitation because it could cause great inconveniences in the development of the system.

1.7 Client requirements

1.7.1 Data capture

A microphone and video camera will be used in order to get video and audio footage from each user. This will enable the users to communicate with one another and will allow my client to be able to host video conferences on the system. Justification for audio and video were provided in 1.2, so here we will justify the use of cameras and microphones. The approach of using microphones and webcams to capture video and audio is fully justified by the fact that there exists no suitable alternatives to these technologies. For video the only other known alternative to a camera is to use a photogram. A photogram is a black and white image *"made by placing objects between a light sensitive paper or film and light source."* [3]. This is unsuitable for video as if we want to achieve a standard frame rate of 30 FPS the users would have to buy a large quantity of photographic paper, costing the user a lot of money. Furthermore photograms take roughly 1 minute to develop, this means that the user would have to properly setup, develop and share photograms with others at a rate of 30 times per second, whilst waiting 1 minute for each photogram to develop. These reasons provide sufficient justification as for why cameras are the only suitable option. As for audio my independent research on this topic indicated that we have not yet found alternatives to microphones for capturing sound.

1.7.2 Data verification and validation

In this system data will be verified upon the creation of a new account. Users will prompted to enter their desired password twice in order to ensure that the user would really like this to be their password and that they have spelt it correctly. Moreover these passwords will be checked to ensure that they are of suitable strength. Justification for this feature was provided in 1.5 under *Usernames and passwords to join a call* and a potential implementation was discussed in 1.4.

1.7.3 Data processing

In order to send video and audio data to others quickly the system will have to compress both. This will be managed by the codecs that we choose to use in the beginning of the peer to peer connection. This will be handled by the WebRTC API once the video call starts. 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 than we should develop it in such a way that there are minimal requirements to use the application smoothly and compression will certainly reduce the system requirements needed to run this application.

1.7.4 Outputs

Outputs like the video and audio from other participants will be displayed on the screen and played out of the user's speakers respectively. 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. 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

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.

1.7.6 Security Issues

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 completly secure during video calls.

1.7.7 Backing up data

Data backups will be made nightly (every 24 hours) to ensure that none of our user's data is lost.

1.7.8 Software requirements

1.7.9 Hardware requirements

1.8 Success criteria

Section 2

Design

Bibliography

[1] Chernev Alexander, Böckenholt Ulf, and Goodman Joseph. "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.

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

[3] The Editors of Encyclopaedia. "Britannica - Photogram". In: *Encyclopaedia Britannica* (2008). URL: https://www.britannica.com/technology/photogram-photographic-print.

[4] "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/.

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

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

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

[8] *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/.

[9] 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.

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