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

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 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 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 [10]. 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 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

[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 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 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 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 1.4.3. 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 [2]. 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 itself. 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. 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.

§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 $\sim$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. [11]

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

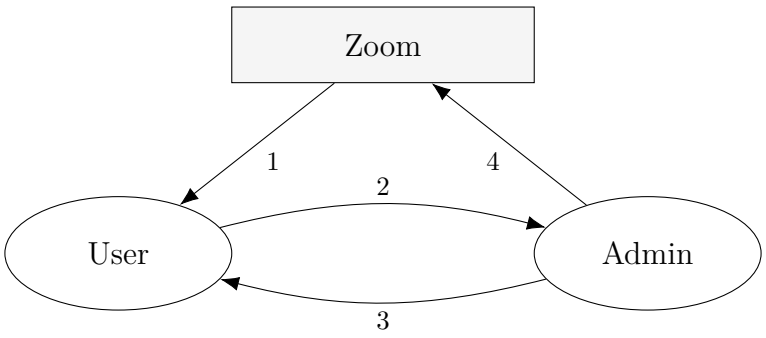

Figure 1.1: Zoom DFD diagram.

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 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.

Justifying the DFD. This level 0 DFD allows me to preseent 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 systems' 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

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 [13] 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 [9]. 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 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?

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

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. 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.

¶1.4.4 Further research to examine suitable approaches

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 [8] 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 [1]. 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. 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 algotrithms because WebRTC provides built-in encryption.

For this application's database we need a system that satifies 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 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 userbase the database could start charging me or my client whilst we weren't aware, this kind of mistake could be economically catastophic 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 guarentee. 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 aswell. 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.

Another concern pertaining to passwords is our method for checking a potential password's strength. I propose the usage of the `zxcvbn` algorithm [12] 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. `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 [4] 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

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

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

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 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 contigously cover the *entirety* of p. The search looks for an optimal match-sequence $s_1, s_2, ...$ of length $|s|$, 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;

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

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 reprecussions 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 clasical example is distorted character recognition,

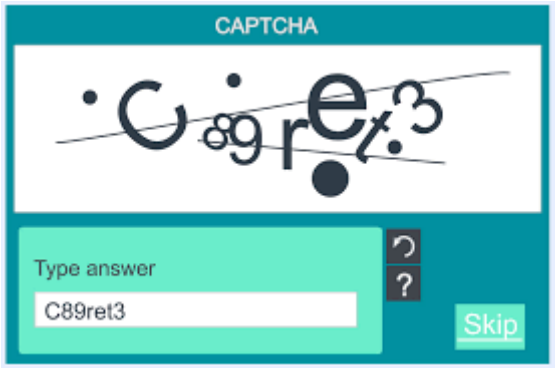

Figure 1.2: A classic CAPTCHA example.

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 descision 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.

Conclusions

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. 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 & 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 appropiate.

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 passcode 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 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" problems that were discussed in the previous paragraph.

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 particpant 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.

§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. [5] 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 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

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

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 1.4.4.

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 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 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 "invisisble" such that it looks like this • • • • • whilst they are typing. An attacker knowing this could then install a keylogger 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

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. Data will also have to be verified before it is put into the database. We should ensure that all data that is being entered into the database is of the correct data type and that `NULL` values aren't permitted if a particular attribute is mandatory. The verification of data entering our database is justified because it prevents a

host of problems arising later on in the development of the system. For example if we don't verify that the data in the username column is only of the string data type, it could be possible that someone's username is a `NULL` field. In this case we will never be able to log this user in because their account isn't accessed by their username meaning that we will never be able to verify that the password that they have inputted is the correct password that is stored in the database.

¶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 video/audio codecs that we choose to use in the beginning of the peer to peer connection. This is handled by the WebRTC API. 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. 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 clealy 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

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. 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. Finally the user's will be prompted to use a strong password that meets a set of requirements, they will also recieve suggestions on how to improve the strength of their password via the `zxcvbn` module.

¶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, as per the request of my client in 1.4.3. 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

This system will be hosted on Google Firebase a cloud hosted, web-app development platform. As discussed in paragraph 4 of 1.4.4, 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 webapps 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 1.4.4 but I ultimately settled on MongoDB because of it's lack of limitations and it's satisfactory storage limits. 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

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: 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

These requirements are based on the old systems requirements. [6] 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.

¶1.7.10 Summary of requirements

Table 1.4: Summary of requirements.

Requirement	Proposed solution	Alternative solution
Data capture	A videocamera 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 promted 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 codebase 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.

Continued on next page

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

Table 1.4: Summary of requirements. (Continued)

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 backend infrastructure. That is a competent programmer should be able to read the codebase 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 1.4.3. 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 malicous 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 [7]. 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 frontend 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.

¶1.8.2 Quantitative criteria

4 - The video call latency should not exceed 150 ms.

Explanation: The delay between a packet of data being sent the packet being recieved 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 occuring in real-time as explicitly requested in 1.4.3. The old system recommended a latency of *"150 ms or less"* [7] so we will seek to match or better this value.

5 - 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 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 1.4.3 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.

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

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.

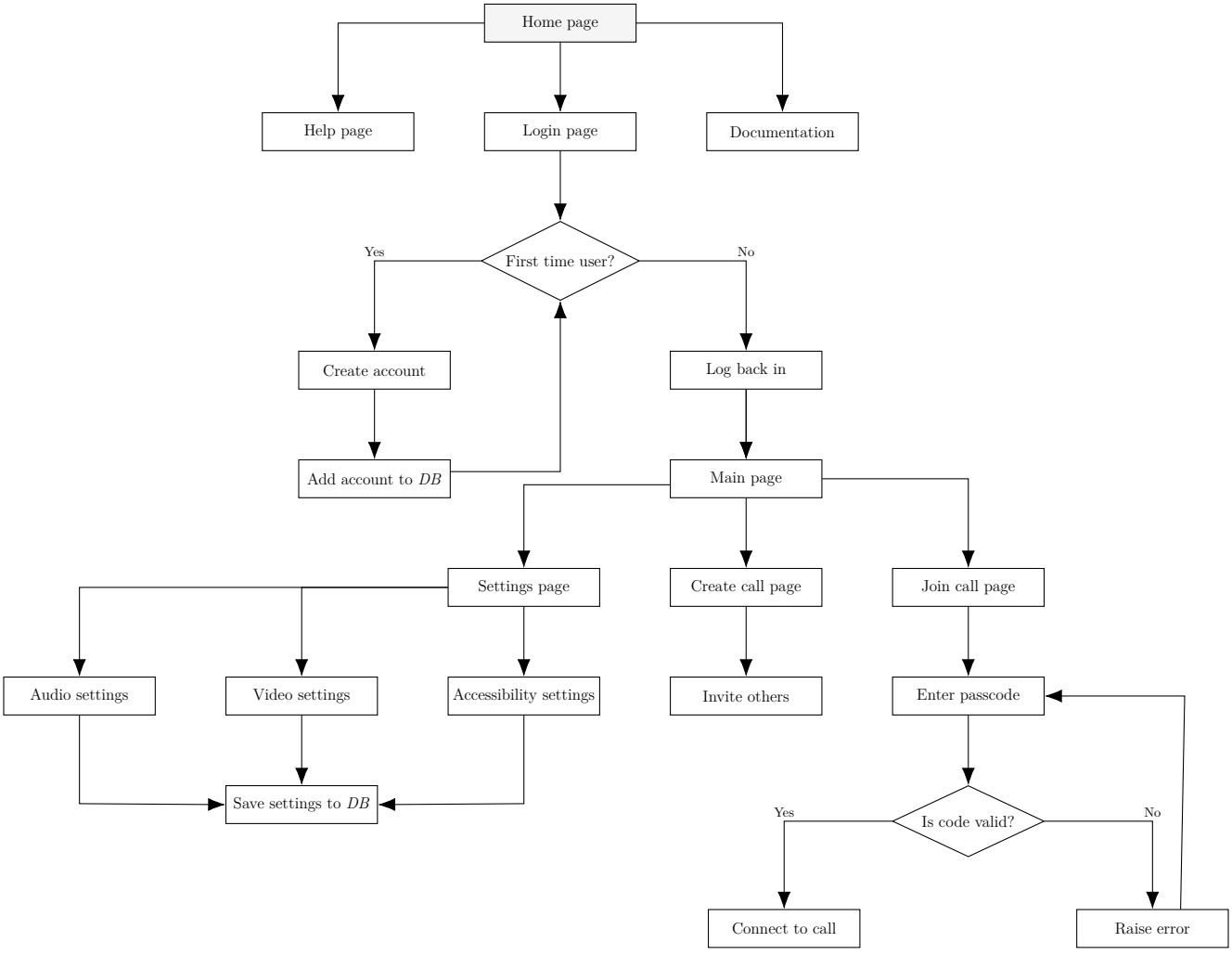

Figure 2.1: Decomposition of the problem.

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 primarly 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 userbase, 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 seperate 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 descision 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 descision 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 1.4.3.

¶2.1.1 Defining the structure of the solution

Entity relationship diagram — *Draft*

The acronym "cfg" denotes configuration.

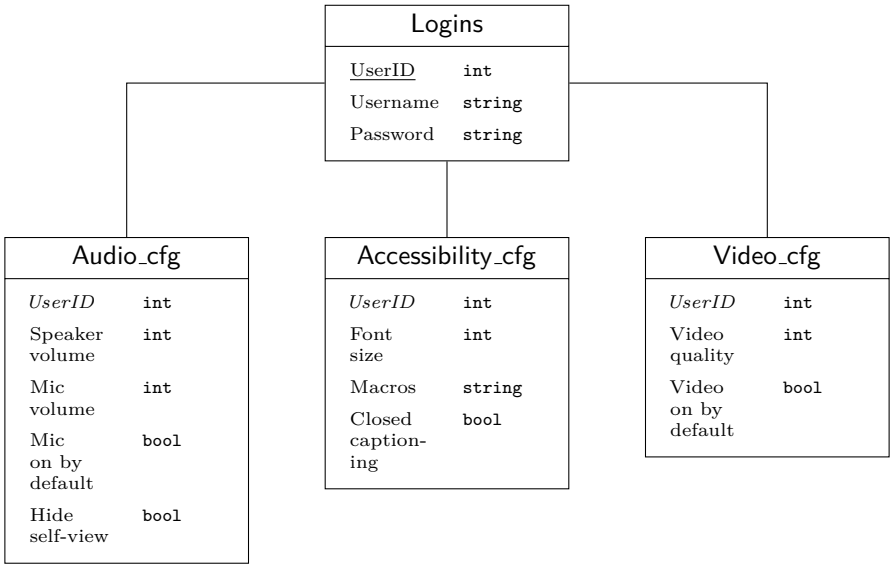

Figure 2.2: The draft ER diagram.

Explaining and justifying the 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 1 Pseudo code for finding all configurations tied to a user.

```
1:  function GET_ALL_CONFIGS(ID)
2:      User_ID ← ID
3:      Configs ← {}                                  ▷ Let Configs be an array.
4:
5:      Configs.insert( Query_tbl(Audio_cfg, User_ID, Speaker volume) )
6:      Configs.insert( Query_tbl(Audio_cfg, User_ID, Mic volume) )
7:      Configs.insert( Query_tbl(Audio_cfg, User_ID, Mic on by default) )
8:      Configs.insert( Query_tbl(Audio_cfg, User_ID, Hide self-view) )
9:      ...
10:
11:     return Configs
```

Algorithm 1 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 — *Final*

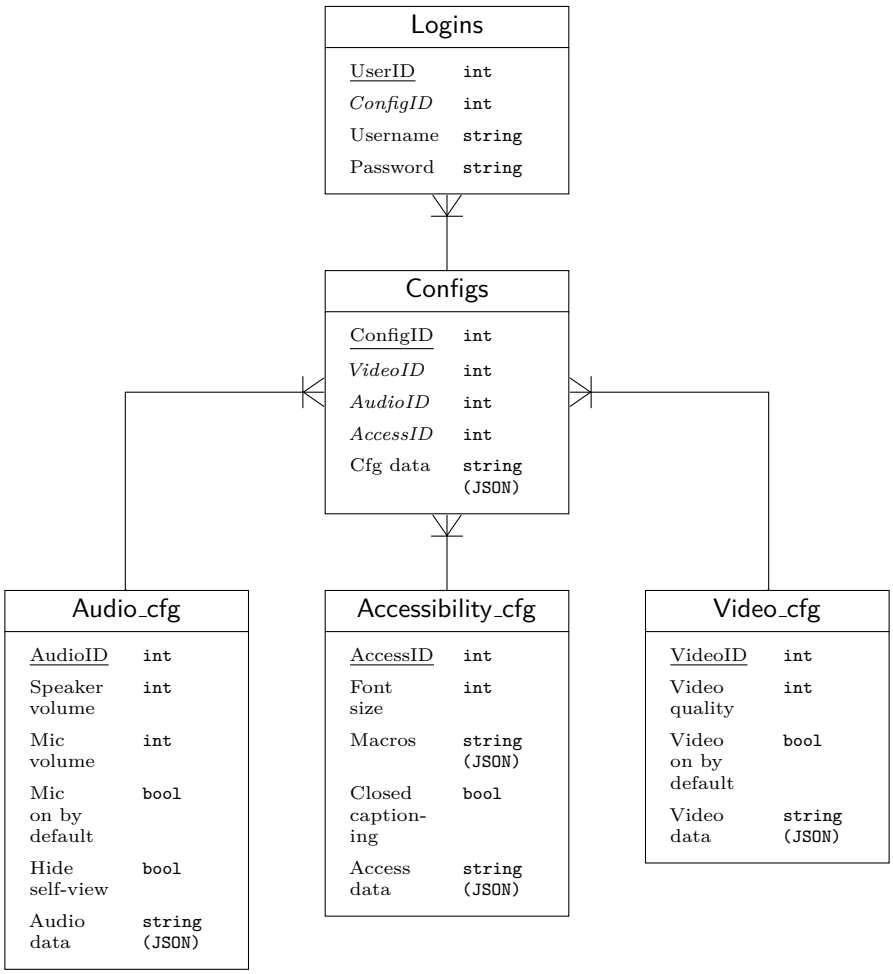

Figure 2.3: The final ER diagram.

Explaining and justifying the final ER diagram. As a pre-requisite to the discussion of our final ER diagram, we will first explain exactly what a JSON string is and why we chose to use it here. A JavaScript object notation (JSON) string is a string of key value pairs similar to a hashmap or a dictionary. As we can see in the name the format of the string is based on JavaScript's syntax. One advantage of this format is that we can quickly turn it into a JavaScript object with JavaScript's built-in `JSON.parse()` function. JSON syntax is straightforward and easy for developers to read and understand;

```
// 32 is the key code for space, 13 is the key code for enter.
var Access_data = JSON.parse('{
  "Font size": 18,
  "Macros": {
    "Unmute": 32,
    "Cam off": 13
  },
  "Closed captioning": false
}');
```

the example above presents how the JSON string in the Access data attribute may look. Thanks to the JSON syntax our data is clear and easily understood, this enables developers to be able to look at the data that is being stored in these JavaScript objects, so that they can have an easier time debugging any unintentional results or errors that may occur in our database system. Hence the application of JSON strings provides a neat and structured format to store data as well as improves the maintainability of the database portion of our system. Moreover 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.

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 string, more precisely a JSON string, that maps each configuration attribute to it's corresponding value. In essence instead of treating each configuration attribute as it's own seperate 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, Cfg data)
```

Not only is this code much shorter and more clear than the code in algorithm 1 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, for each configuration option let $X_{i,1}, X_{i,2}, \ldots, X_{i,n}$ a sequence of n independent discrete uniform random variables, where $X_{i,j}$ denotes 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. Denote $\overline{X_i}$ to be the sample mean of the i-th configuration for all or our n users. By *Kolmogorov's strong law* [4] we have,

$$Pr\left(\lim_{n\to\infty} \overline{X_i} = \mu_i\right) = 1 \tag{2.1}$$

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. Whilst equation 2.1 can provide some motivation for the direction of our arguement *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 userbase 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.

Proposed system level 0 DFD

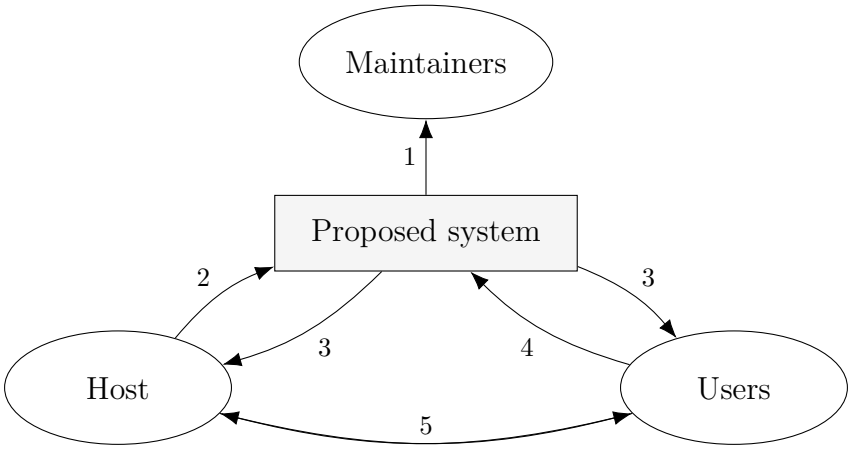

Figure 2.4: Proposed system level 0 DFD diagram.

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 recieve 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 maintable 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

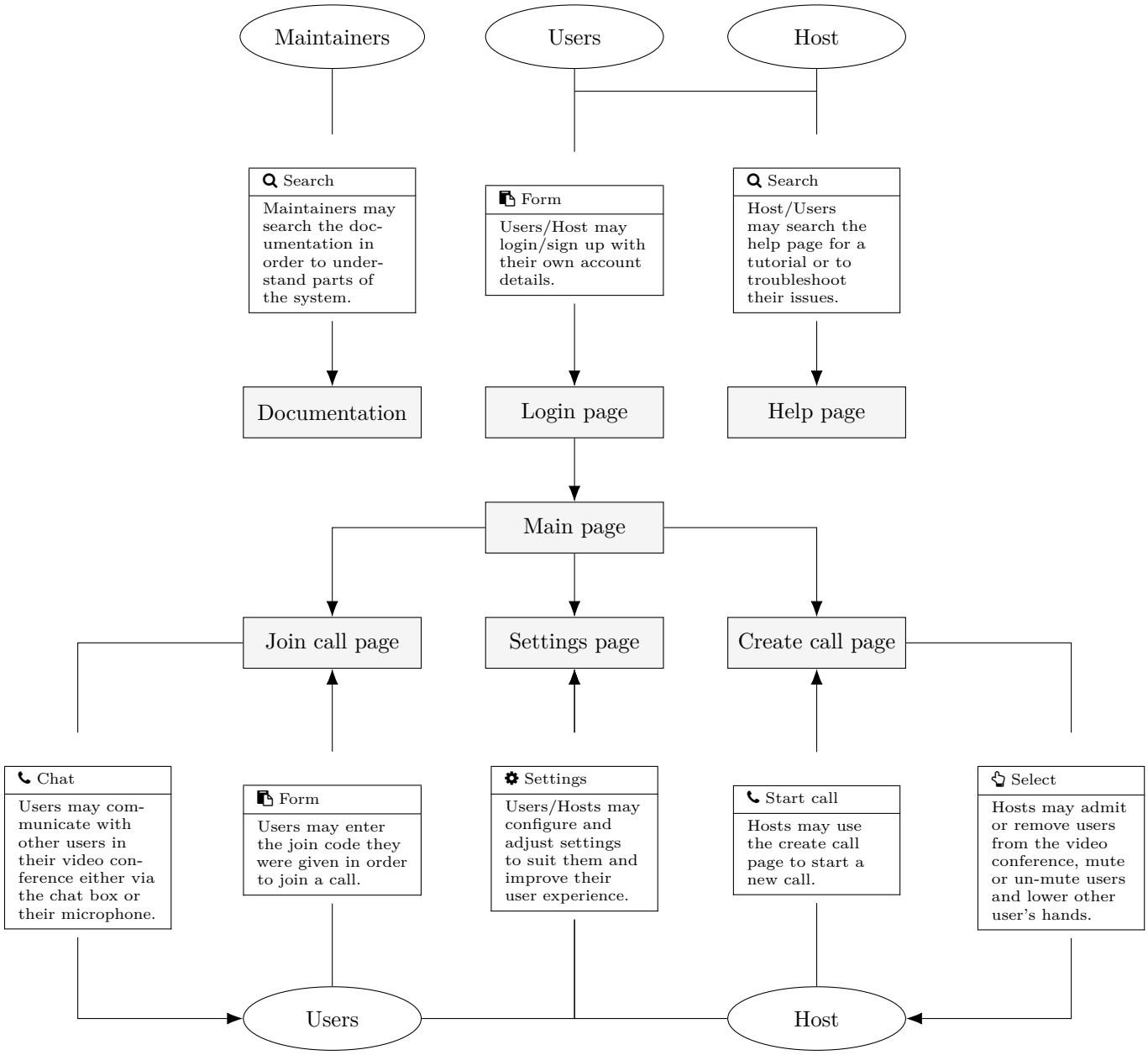

Figure 2.5: Proposed system level 1 DFD diagram.

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. 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 the other components. Finally we can see that the level 1 DFD diagram provides a neat, cohesive visual representation of our system's architecture.

¶2.1.2 Proposed algorithms

Bibliography

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

[2] 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.

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

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

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

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

[10] *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/.

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

[12] 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.

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