School of Electronic Engineering
and Computer Science

Final Report

Programme of study:
BSc Computer Science

Project Title:
Dictate – Video-conferencing for small meetings and groups

Supervisor:
Huan Zhang

Student Name:
Yaqoob Maniar

Final Year
Undergraduate Project 2023/24

Date: 29th April 2024

Abstract

With advancements in technology, the usage of video-conference systems in businesses and in particular meetings has rapidly grown. Meetings have the potential to be highly productive mediums whilst, conversely, also have the potential to be the means to waste a company's time and resources.

Dictate is a browser application created to investigate the effect collaborative tools have on the productivity and enjoyment of video-conference meetings. Implementing live group notes and meeting recordings paired with team calendars; a variety of tools were designed, implemented and later tested.

Feedback suggested advancements in video-conference technologies would push more individuals to choose a video-conference in the future and showed a positive correlation between the advancements in UI/UX design and user satisfaction.

Contents

Table Of Tables

Table Of Figures

Chapter 1: **Introduction**

1.1 Background

One of the most inhibiting forces to organisational effectiveness is a lack of effective communication (Lutgen-Sandvik, 2010, cited in Ahyia Adu-Oppong & Agyin-Birikorang, 2014). For this reason communication skills are consistently one of the most valued skill sets sought by employers (Choren, 2015). For this same reason there is a significant importance on meetings (the chosen communication medium of the professional world). Employees average 6 hours per week in meetings whilst managers bolster averages of 23 hours per week (Rogelberg, Scott & Kello, 2007). They are from the most important ways a company organises and orchestrates itself with estimates indicating that top managers spend as much as 60-75% of their time in face-to-face or telephone meetings (Kloppenborg & Petrick, 1999; Fulk & Collins-Jarvis, 2001).

With advancements in technology, the usage of video-conference systems in businesses and in particular meetings has rapidly grown. A report conducted by Dialpad (2022) found that 36.98% of employees spent between 4-12 hours in video-conference meetings a week while the majority (46.15%) spent 4 hours or less in the activity. In a study conducted by Denstadli, Julsrud & Hjorthol (2011) they found that 68% of business travellers had access to video-conference technology. The global video conferencing market size was valued at $7.02 billion in 2022 and is projected to more than double from $7.76 billion in 2023 to $17.05 billion by 2030 (Fortune Business Insights, 2022).

1.2 Problem Statement

While some meetings are highly productive and valued by attendees, a substantial number are not - with estimates as high as 41.9% (Schell, 2010, cited in Kauffeld & Lehmann-Willenbrock, 2012). Companies and business individuals reduce this by selecting the most optimal meeting method available which often turns away from the video-conference medium. This is seen in the study conducted by Denstadli, Julsrud & Hjorthol (2011) where they found 66% of answers as to why they had arranged a face-to-face meeting as opposed to a video-conference was due to the poor suitability.

1.3 Aim

The aim of this project is to research and develop a video-conferencing platform tailored toward small meetings in the professional setting. Key features desired in these environments include group note-taking, recordings and individual and group calendars which are either not found in or poorly optimised in the current tools available. The project aims to bring together these features in a single, easy-to-use platform focussing on usability, stability and responsiveness.

1.4 Objectives

- To investigate current research regarding the effectiveness of video-conference technologies in the professional setting.
- To research and compare existing video-conference technologies.
- To conduct a survey regarding the use of video-conference technologies.
- To develop a video-conference system suitable for small groups and meetings.
- To implement suitable technologies for the video-conference system including recordings, group notes and calendars.
- To validate the video-conference system via various testing methods.
- To evaluate the video-conference system in light of user feedback.

1.5 Research Questions

- How can the improvement of existing video-conference technologies be achieved?
- Would advancements in video-conference technologies push more users to select a video-conference meeting over a face-to-face-meeting?
- How can the implementation of video/audio recordings as well as live group note taking be carried out?
- How does UI/UX design impact user experience of video-conference meetings and does it improve meeting effectiveness?
- How can group teams and calendars affect the user experience of video-conference systems?

Chapter 2: **Background Research**

There is a plethora of research pertaining to meetings and the use of video-conferencing in the business environment as well as many documented efforts towards the creation of improved video-conference systems. This chapter presents an analysis of meetings and video-conference technologies followed by an analysis of popular video-conferencing systems in the market.

2.1 Literature Review

Meetings are defined as a gathering of three or more people who agree to assemble for a purpose ostensibly related to the functioning of an organisation or group (Schwartzman, 1989). Their mastery is vital for organisations to succeed with Adriel Aseniero et al. (2020) finding that ineffective meetings are bad for morale and productivity, they negatively impact employees' health and are directly linked to organisations' wasted time and resources. On the other hand, studies showed that improved internal communication can increase organisational productivity by as much as 25% (McKinsey Global Institute, 2012).

In order to investigate how meetings can be optimised, a study conducted across 1,411 business travellers in Oslo (Denstadli, Julsrud & Hjorthol, 2011) had the aim of investigating the use of video-conference systems and face-to-face mediums in meetings. Both video-conference and face-to-face meetings were most commonly found to be used for project work whereas video-conferences were mainly used at the intra-organisational level (70% of the time) unlike face-to-face meetings which were more widely dispersed across customers, suppliers and regional bodies. Such findings show that video-conferencing is best suited and most effective between teams and groups where formality and building new relationships is not as valued as the ability to easily connect and effectively communicate desires. This can be seen with the findings of the study contrasting planning times between the mediums (see Table 1); the results show video-conferencing to be more suitable for flexible and short-notice meetings.

Planning Time	VC (%)	FTF (%)
The same day	5	–
1-3 days before	18	5
3-7 days before	26	14
1-2 weeks before	20	22
2-4 weeks before	14	24
1 month or longer before	17	35
Total	100	100

Table 1: Planning time for the most recent video-conference (VC) and face-to-face (FTF) meeting (Denstadli, Julsrud & Hjorthol, 2011)

Denstadli, Julsrud & Hjorthol concluded with a proposed graph plotting when best suited to choose a video-conference over a face-to-face meeting in order to optimise efficiency and productivity (see Figure 1). Their findings prove that video-conferencing is more suitable over face-to-face meetings when tasks are unambiguous and explicit (such as daily stand-ups and weekly updates).

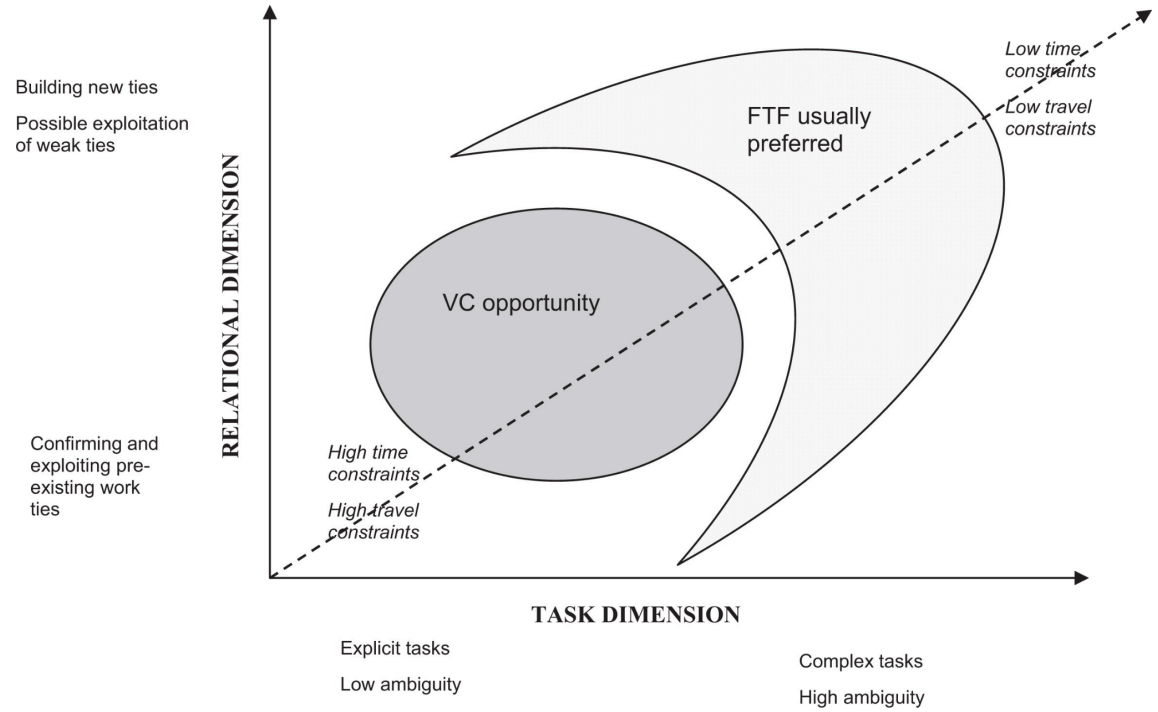


Figure 1: Relational and task-related dimensions that motivate choices for video-conferencing (VC) and face-to-face (FTF) meetings (Denstadli, Julsrud & Hjorthol, 2011)

In a report by Dialpad (2022), respondents were asked “Which features were missing from your video-conference solution?”. Answers varied with more than 30% saying they desired to meet without downloading software (e.g. browser support) and more than 20% remarking the need for meeting recordings. When asked about users’ biggest “pain point” regarding virtual meetings the majority replied with technical problems: more than 50% citing audio issues and more than 30% citing video issues. This shows one of the largest inhibitors of video-conferencing is not the software technology but rather hardware restrictions which the software relies upon.

Results from a study conducted by Cohen et al. (2011) showed a negative relationship between size and meeting quality with previous researchers adding that as group size increases, member cohesion and intimacy decreases (Fisher, 1953; Lundgren & Bogart, 1974; Seashore, 1954). These findings support the notion that the smaller a meeting is, the more likely it is to be effective.

Adriel Aseniero et al. (2020) produced ‘MeetCues’ as a support add-on to video-conference technologies; their aim was to improve the meeting experience by adding engagement features such as a commenting system and informal reactions such as ‘like’ and ‘clarify’. The project was met with success according to the recorded feedback and proves how UI/UX choices (among other things) can positively impact user engagement and meeting effectiveness.

The research project Colab had the long-term goal to build computer tools to make meetings more effective (Stefik et al., 1987). They introduced a concept named WYSIWIS (what you see is what I see - pronounced “whizzy whiz”) which refers to all participants seeing the same display, giving the impression that all members are interacting with shared and tangible objects. Originating from the chalkboard, the concept of a shared screen to visualise ideas has a large role to play in both face-to-face and virtual meetings.

2.2 Existing Applications

The following is an analysis of 3 of the top 4 video-conference software technologies ordered by market share: Zoom, Microsoft Teams and Google Meet - all together holding 88.1% of the video-conference market (Statista, 2023).

2.2.1 Summary

A summary of the findings is presented below highlighting key features and limitations of each technology.

Software	Key Features	Limitations
Zoom	● Meeting security and management ● Available as a desktop application ● Collaborative whiteboard templates ● More than 2500 third-party app integrations	● Calendar system requires 3rd party integration ● Overwhelming and complicated UI/UX
Microsoft Teams	● Strong team functionality ● Available as a desktop application ● Collaborative whiteboard templates ● Integration with Microsoft 365 ● More than 2000 third-party app integrations	● Recordings not available on all plans ● Resource demanding
Google Meet	● Simple and intuitive interface design ● Integration with Google Workspace	● Lack of teams ● Whiteboard is used in separate tab ● Limited collaborative tools

	• Limited number of 3rd party app integrations

Table 2: Comparison of 3 of top 4 video-conference software technologies (Statista, 2023) highlighting software features and limitations

2.2.2 Zoom

Zoom is an all-in-one collaboration platform providing team chat, meetings, whiteboard and workspace solutions to help hybrid teams collaborate (Zoom, 2024). Founded in 2011, their system provides an easy way to access video-conference meetings focusing on security and reliability. Zoom hosts can password protect, set up authentication profile restrictions, and add waiting rooms to their calls (Zapier, 2023).

The drawbacks of the system include having a clunky and confusing UI/UX design, negatively impacting user experience. Their implementation of calendars can also be seen as a drawback, requiring third-party integration such as Google or Microsoft 365 without providing an option for users who would prefer a stand-alone solution.

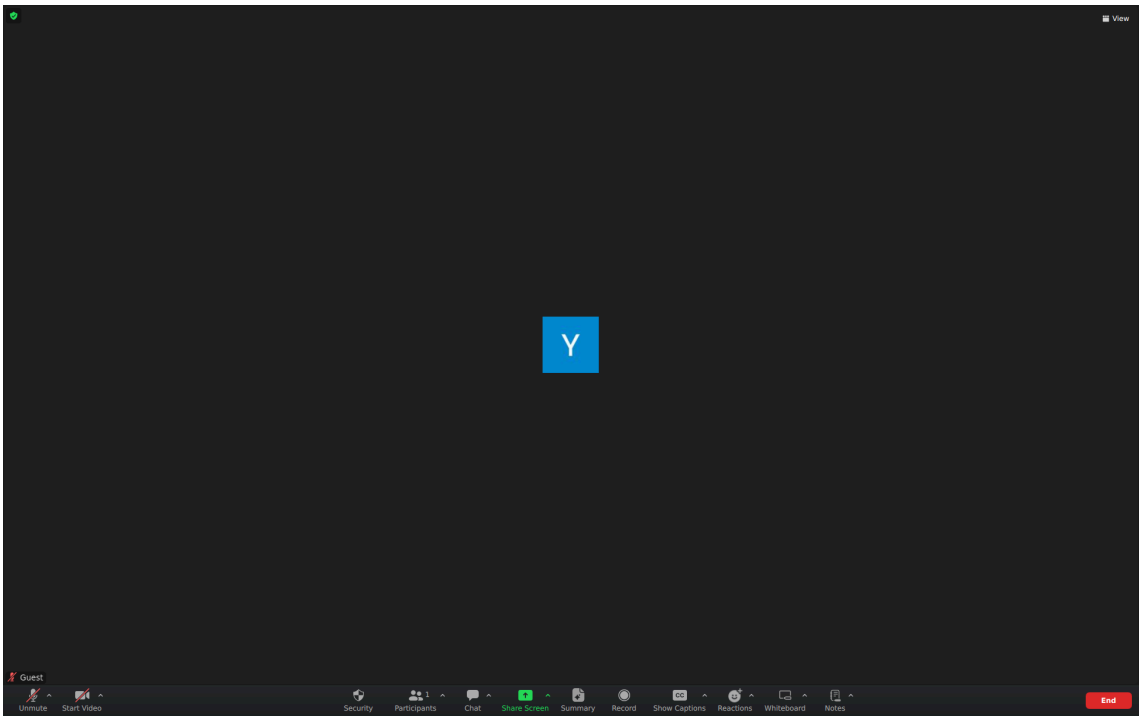


Figure 2: Screenshot of Zoom video-conference meeting

2.2.3 Microsoft Teams

Microsoft Teams is a messaging application for organisations, a workspace for real-time collaboration and communication (Microsoft, 2024). The focus on collaboration is seen in the teams functionality which allows creating teams and channels, allowing the sharing of files and messages. In being a Microsoft product there is also a strong integration with Microsoft 365 apps such as Word, PowerPoint, etc. which allows file sharing inside and outside of meetings.

Drawbacks of the system include its high resource requirements (i.e. RAM) as well as the requirement of using a Microsoft email address to access the system. It is also less capable at coping with larger meeting sizes due to having to provide access to its collaboration tools simultaneously (Zapier, 2022).

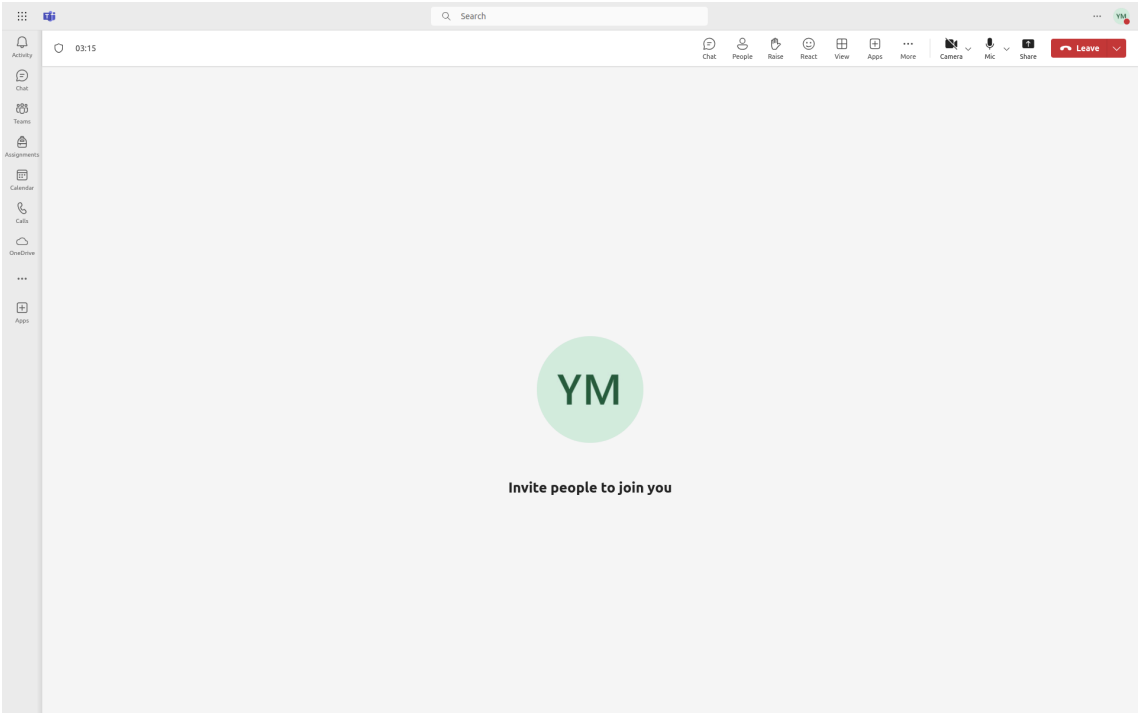


Figure 3: Screenshot of Microsoft Teams video-conference meeting

2.2.4 Google Meet

Google Meet provides secure, easy-to-use video calls and meetings for everyone, on any device (Google, 2024). From the standout features of the platform is its simplicity and intuitiveness, a minimalistic whilst understandable UI/UX design positively affects end users. Its seamless integration with the Google Workspace allows users to attach their Google Calendar to the system as well as share Google Docs or Google Slides in their meetings.

From the negatives of the platform is its limitations in collaborative tools, relying on other Google products with only a handful of third-party tools. Whilst users can work on a shared Google Doc and be present in a meeting in the same tab, the option is not available for whiteboards (named Jamboards). Also the absence of the team feature restricts end users in organising their calendars and creating privacy restrictions when sharing resources.

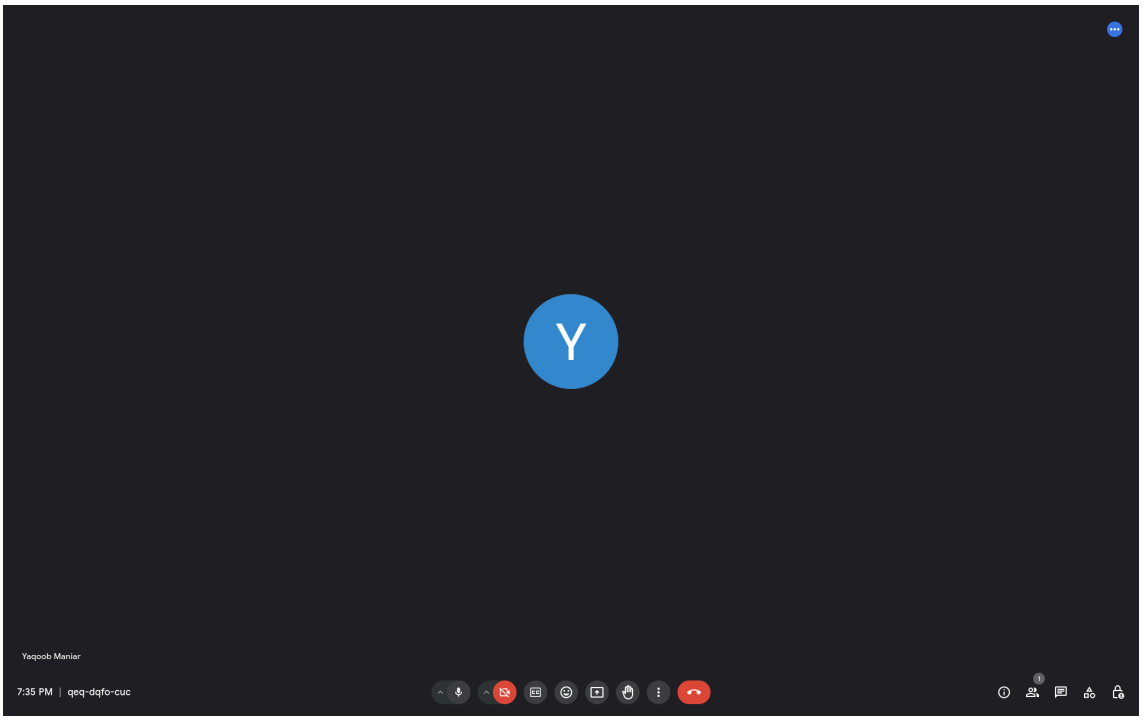


Figure 4: Screenshot of Google Meet video-conference meeting

Chapter 3: **Requirements Analysis**

This chapter presents the research and elicitation of function and non-functional requirements for the proposed video-conference system. The specification guided the following processes of design, implementation and later evaluation.

3.1 Requirements Research

3.1.1 Survey

A requirements survey was conducted aiming to identify video-conferencing habits and opinions across a range of ages and occupations in a local population. A mixture of closed and open ended questions were asked in order to obtain a mixture of both statistical and qualitative results. A full list of the questionnaire accompanied with its results can be found in Appendix A.

3.1.2 Analysis of Results

Key findings from the survey included 75% of participants currently accessing video-conference technologies as an app whilst 56% would prefer to access the technology in the browser. Other findings included 87.4% of average meeting sizes were between 1 - 20 members and 93.7% of average meeting lengths were under 2 hours long. Most problems raised in relation to current video-conference technologies were pertaining to technical issues such as video/audio lag and, accordingly, most improvements users wanted to see in a video-conferencing application were regarding stability and stream quality.

3.2 Requirements Elicitation

The following functional and non-functional requirements were elicited using the aforementioned literature review, similar systems comparison and public survey.

3.2.1 Functional Requirements

1. The system should allow users to create an account and log in. They should be able to update their profile information and delete their account upon request.
2. The system should allow for the creation of video-conference meetings by guest and logged-in users. Said users should be able to join video-conference meetings by an id.
3. The system should allow for users to access group notes inside a video-conference meeting. Said notes should be able to be interacted with by all participants simultaneously, being able to see live changes.
4. The system should allow for users to record video-conference meetings.
5. The system should allow logged-in users to access a meeting’s notes and recordings after said meeting has ended.

6. The system should allow for the creation of teams and for logged-in users to be able to join/leave said teams.
7. The system should allow for users to interact with shared calendars for each team they are a member of. Users should be able to schedule meetings for a specific calendar and join a scheduled meeting upon request. These scheduled meetings should be private.

3.2.2 Non-Functional Requirements

1. The system should be made available as a website accessed by a browser.
2. The system should have an availability of 99.9% (~1.44 minutes a day of downtime)
3. The system should not take up more than 100MB of RAM per open window.
4. The system should cater for video-conferences of sizes up to 20 members.
5. The system should allow video-conferences to last for up to 2 hours long.
6. The system should not stop/interrupt a video-conference upon the loss of connection of one of the participants.
7. The system should limit the lag of meeting participants' audio and video up to a 1 second delay.
8. The system should allow for recordings to be viewed inside the browser.

Chapter 4: **Design**

This chapter presents the design of the system to be created. Based upon the previous chapter, use cases, user flows and wireframes have been produced as a template for the system. Following these high-level diagrams is the system design accompanied with appropriate software diagrams.

4.1 User Flow

4.1.1 Use Cases

An overview of the use cases describing the video-conferencing system can be seen in Figure 5, after which follows a more detailed description presenting key use case specifications - the full list of which can be found in Appendix B.

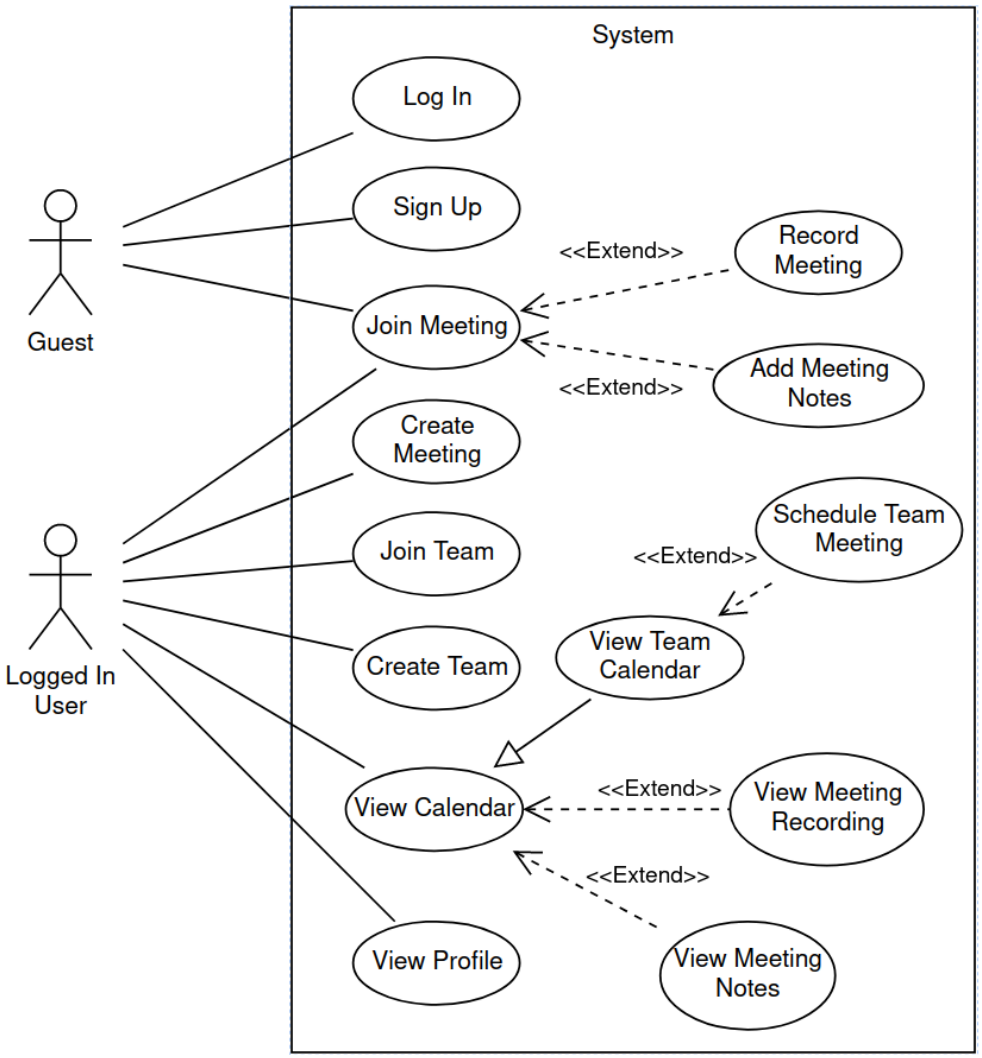


Figure 5: Use Case Diagram

Use Case	Join Meeting
Description	The Join Meeting Use Case provides the functionality for Guest Users and Logged in Users to join an active video-conference session
Actors	Guest, Logged In User
Basic Flow	1. User clicks Join Meeting
	2. User enters Meeting Id
	3. System adds User to video-conference
Alternative Flows	None
Exception Flows	2.1. User enters invalid Meeting Id
	2.2. System informs User of invalid Meeting Id and asks to try again
	2.3. Use Case returns to step 2
Special Requirements	1. Guest Users should be prompted to enter their name to be displayed in the meeting
Pre-Conditions	1. The User must allow browser access to camera and microphone
	2. The meeting must not have reached its capacity (30 members)
Post-Conditions	1. Users should be displayed in the meeting with either the name attached to their account (Logged In Users) or with the name inputted on entry (Guest Users)
Extension Points	1. Logged In Users are able to record the meeting by clicking the record button
	2. Users are able to access group notes by clicking the notes button.
	3. Users are able to leave the meeting

Table 3: Use Case Specification: 'Join Meeting'

Use Case	View Calendar
Description	The View Calendar Use Case allows Logged In Users to view their personal calendar, allowing them to view scheduled meetings, schedule new meetings and view meeting recordings and notes
Actors	Logged In User
Basic Flow	1. Logged In User clicks My Calendars
	2. System navigates Logged In User to Calendars Page
Alternative Flows	None
Exception Flows	None
Special Requirements	1. The calendars page should display the current month when first loading

Pre-Conditions	None
Post-Conditions	1. The Logged In User should be able to see all scheduled meetings in their calendar
	2. The Logged In User should be able to navigate to different months of the year (past and future)
Extension Points	1. The Logged In User is be able to navigate to a calendar specific to a team, this calendar will only show scheduled meetings and resources for that team
	2. The Logged In User is be able to schedule a meeting for a specific team (see Schedule Team Meeting Use Case)
	3. The Logged In User is be able to delete a scheduled meeting they created
	4. The Logged In User is be able to see a past meetings' recordings and notes (see View Meeting Files Use Case)

Table 4: Use Case Specification: 'View Calendar'

4.1.2 User Flow Diagrams

User flow diagrams describing team and calendar use cases are presented below, for a full list of the user flow diagrams see Appendix C.

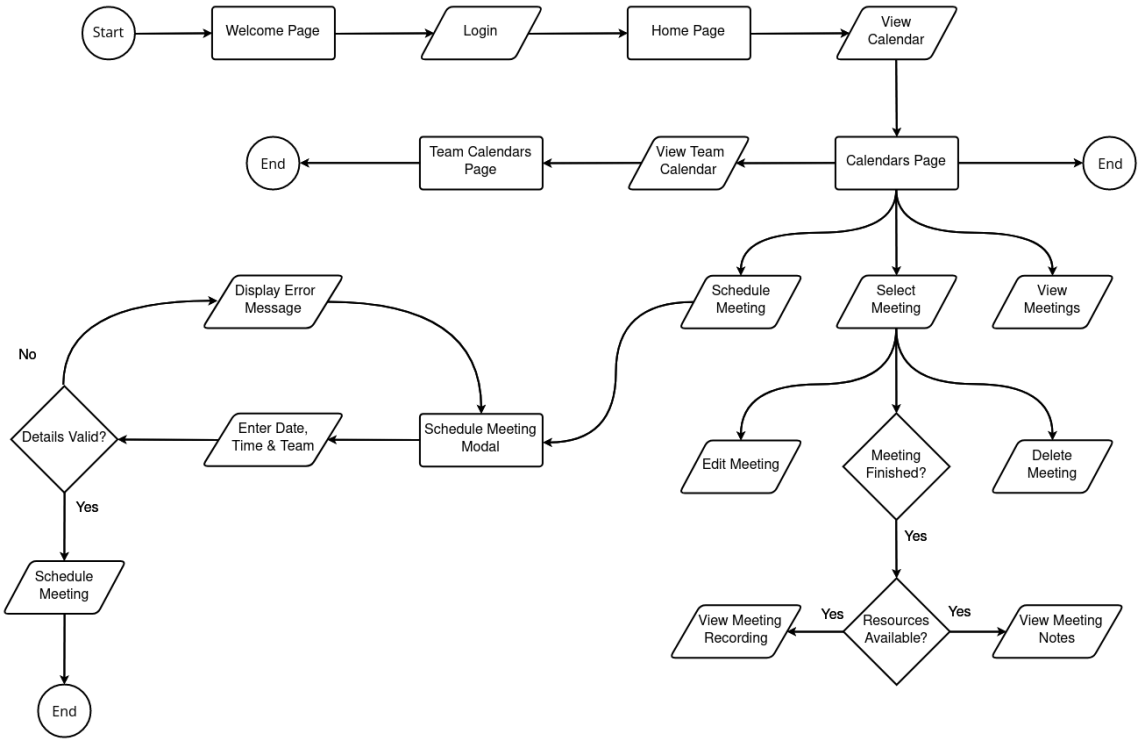


Figure 6: User Flow Diagram: 'View Calendars'

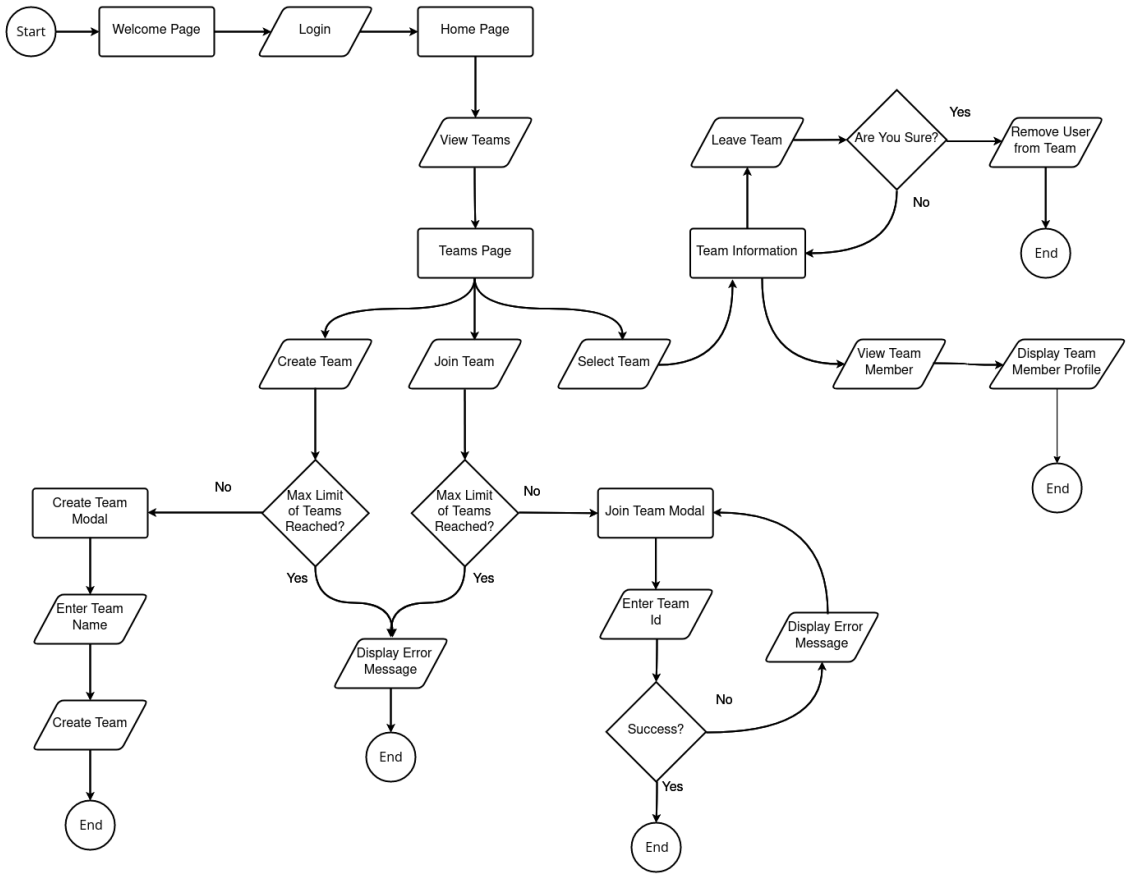


Figure 7: User Flow Diagram: 'Join Team'

4.1.3 Page Designs

Wireframes were developed describing the system which were subsequently labelled highlighting key design choices (a full list of which can be found in Appendix D). The focus of the templates was to incorporate a minimal and intuitive design to aid the end-user experience; focus was given to using symbols and icons over text based prompts and usage of whitespace effectively in order to achieve this.

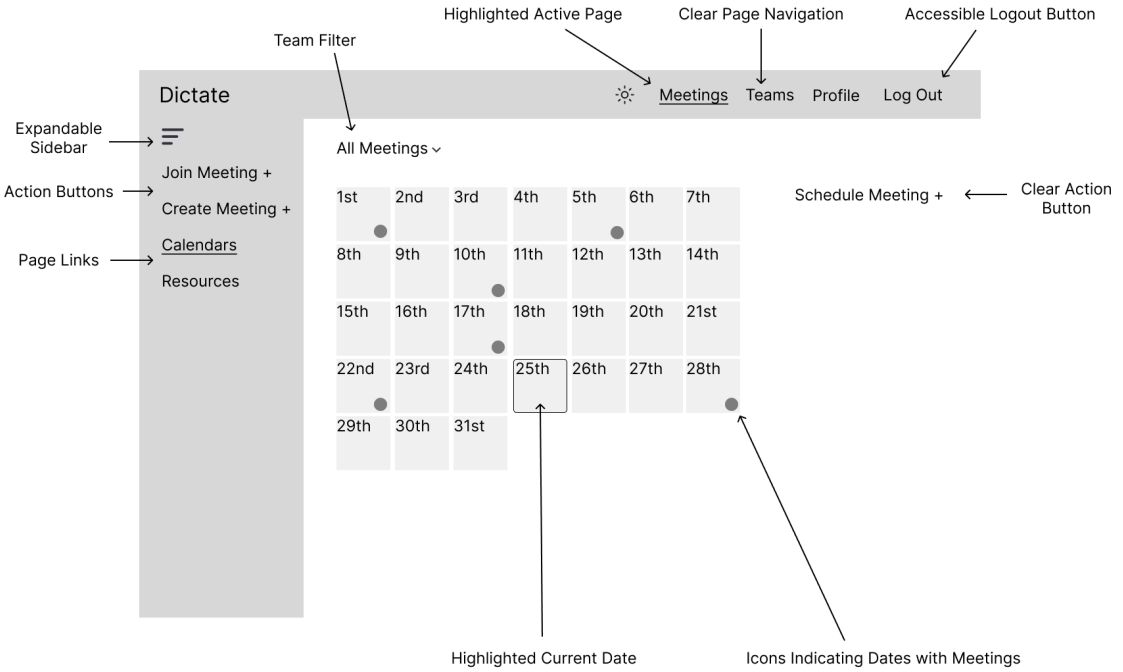


Figure 8: Wireframe Diagram: 'Calendars'

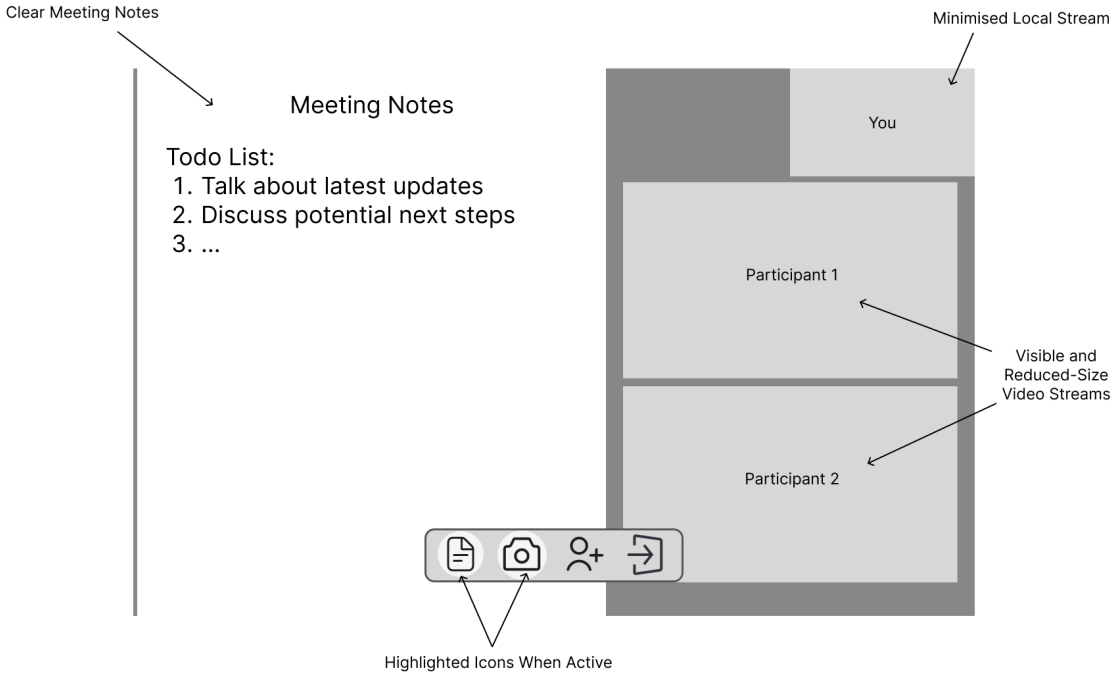


Figure 9: Wireframe Diagram: 'Meeting'

4.2 System Architecture

Software designs were built upon the aforementioned high-level designs which are detailed in this section accompanied by an explanation for design choice. For a complete list of system architecture diagrams refer to Appendix E.

4.2.1 Overview

An online video-conferencing system requires two main processes in addition to that of a regular web server in order to function: a procedure for the clients to initialise their connection between themselves and a procedure through which they can stream video and audio to each other. From this, a traditional web server and relational database were chosen as a foundation on top of which a websocket server and STUN server were added in order to provide the functionality for the first process. After the clients had initialised connection, a peer-to-peer topology using WebRTC was chosen to enable the clients to stream video and audio between themselves. To incorporate the video recording functionality a task queue which communicates with a worker was added, storing the videos in the cloud to later be accessed by the client.

The design choice was made to containerise each individual process (namely the servers and worker) in order to add abstraction, security and reliability to the system. If a process fails it can simply be restarted and if a different implementation was chosen, no code other than that said process would need to be changed.

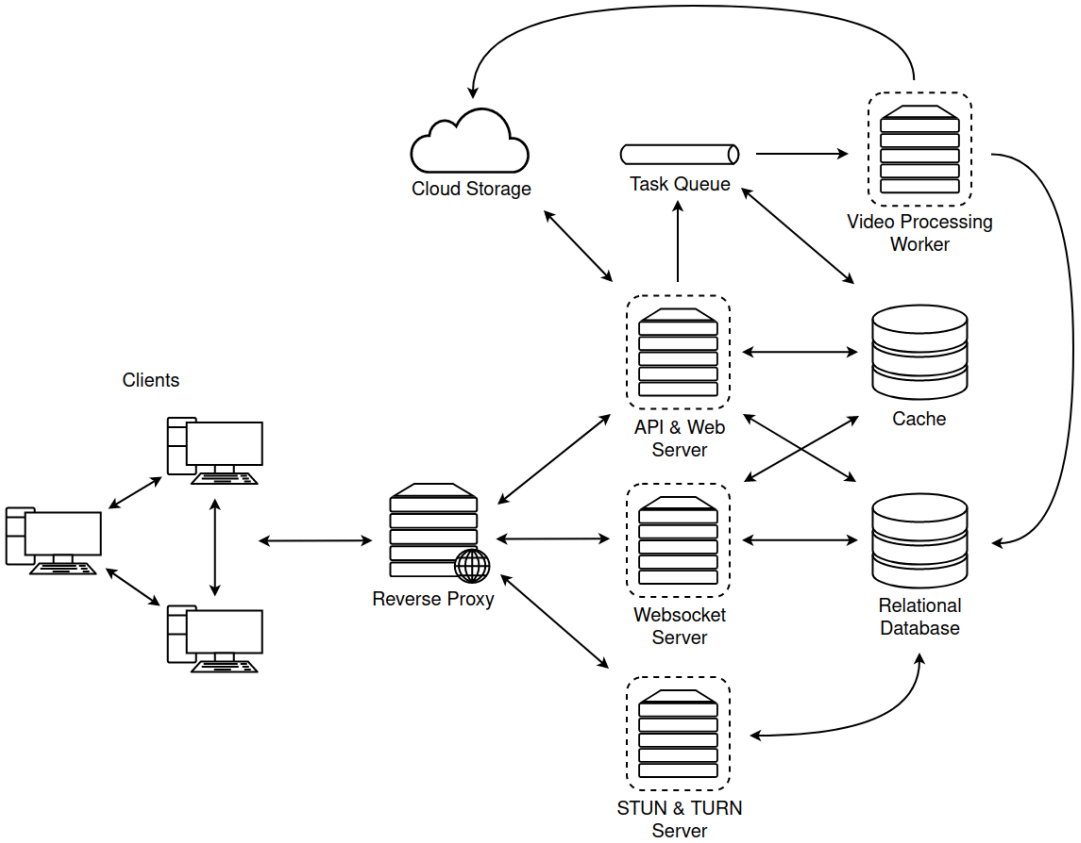


Figure 10: System Architecture (summarised)

Another reason for the choice of a microservice architecture as opposed to a monolithic architecture is the flexibility it allows in scaling individual processes as opposed to others. This was an important design choice for the system as having many concurrent meetings at a single time would increase load on websocket connections whilst API requests would stay relatively constant (by using browser caching) and so being able to specifically scale the websocket process is important.

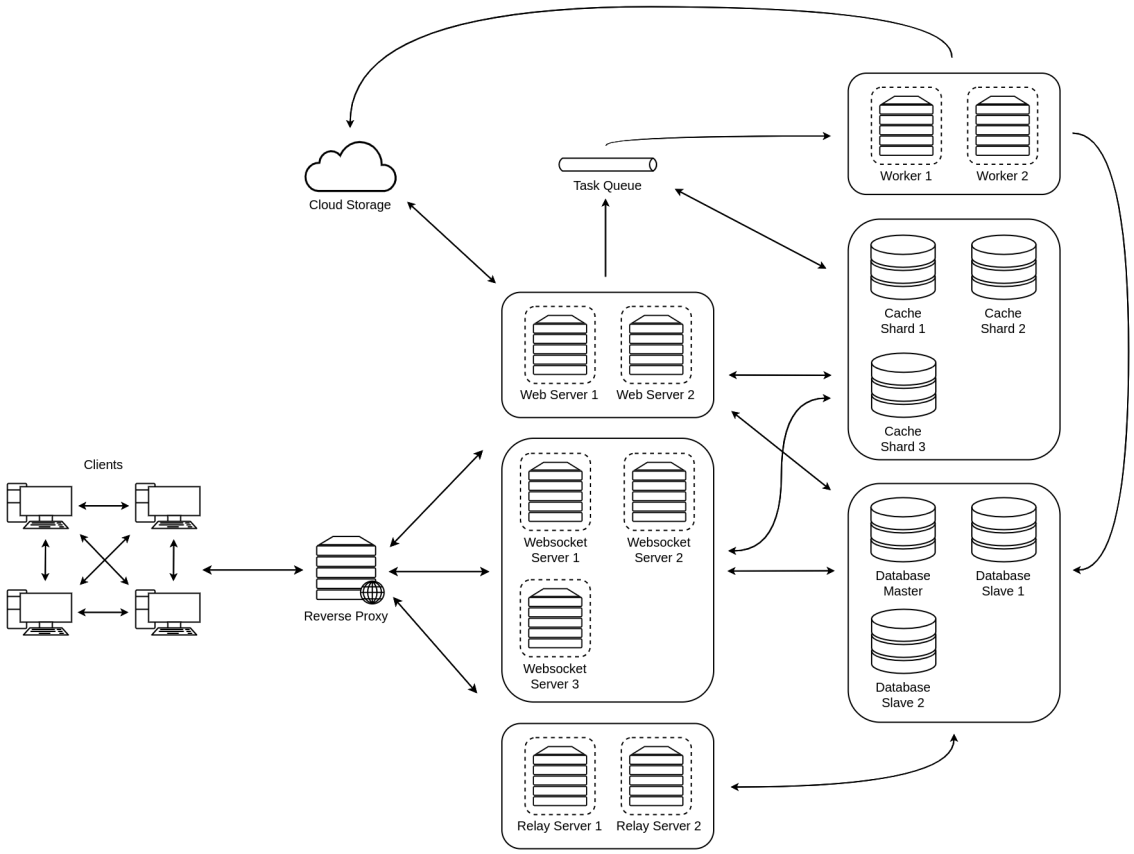


Figure 11: System Architecture (scaled-up)

4.2.2 Connection Topologies

A mesh topology was chosen to enable clients to communicate between themselves inside the browser. The choice was made between mesh, SFU (Selective Forwarding Unit) and MCU (Multipoint Control Unit) topologies (see figures a,b,c). Both MCU and SFU topologies require a ‘middleman’ media server although implemented in different ways: an MCU transforms all streams into a single output stream whilst an SFU simply relays all streams to each participant.

Both MCU and SFU topologies provide the flexibility to transform incoming streams before sending them out such as by managing bandwidth and changing codecs. They also are preferred when recording is desired as a functionality since there is direct control on how the video is rendered (since we create the video server-side as opposed to client-side for peer-to-peer). Another benefit of using said topologies is their efficiency; as the number of participants grows, the number of upstream connections in the network grows at a constant rate (each client has one upstream to the server), as opposed to the mesh

topology where if a participant is added, every peer needs to create an new upstream to that new participant in addition to their previous upstream connections.

The design choice of using a mesh topology was made due to the speed and security of connections. Participants communicate directly with each other without the need for an intermediary server (which may reside in the opposite geographical location to the direction of the call) and the connection is secured from both ends of the call where stream encoding and decoding takes place. In addition the tools made available to implement a mesh topology do not require advanced domain knowledge like that of MCU and SFU topologies which require a media server (being out of the scope of this project).

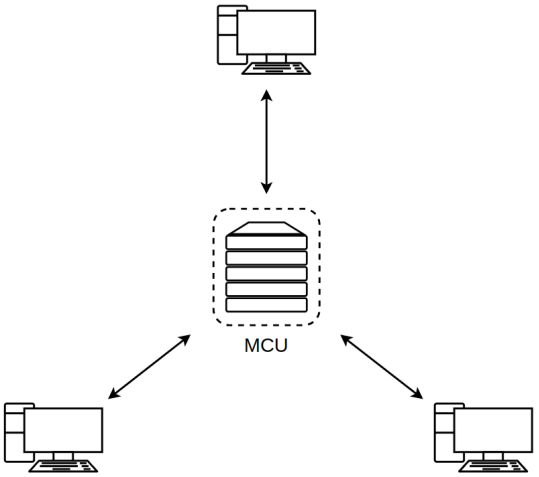


Figure 12: MCU Topology

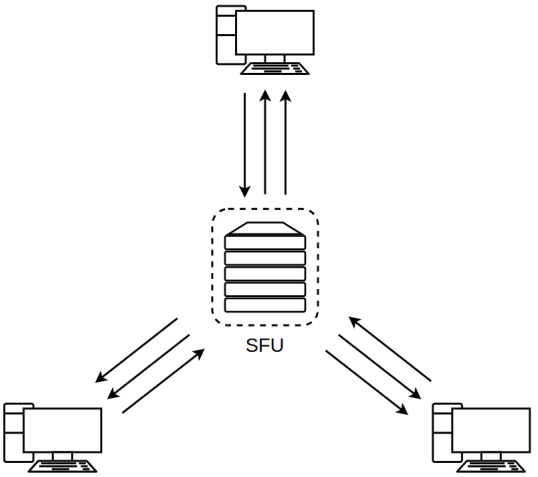


Figure 13: SFU Topology

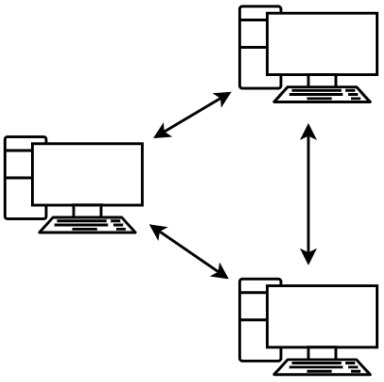


Figure 14: Mesh Topology

4.2.2 WebRTC

WebRTC was chosen as the technology through which to establish peer-to-peer connections streaming video and audio. Prerequisites of this technology requires a STUN (Session Traversal Utilities for NAT) server and optionally a TURN (Traversal Using Relay around NAT) server in addition for a way for the

clients to initialise connection between themselves. A STUN server is used to first gather ICE (Interactive Connectivity Establishment) candidates which are used to determine the IP address and port allocated to it by a NAT - through which it can be contacted directly by the other client.

A websocket server was chosen as the service to aid initialising a connection between two clients as opposed to a solution such as polling since it allows for real-time communication of SDP offer/answers. The SDP offer and answer made between two clients allow for the negotiation of media formats, transport protocols, and other important details required for a successful peer-to-peer connection.

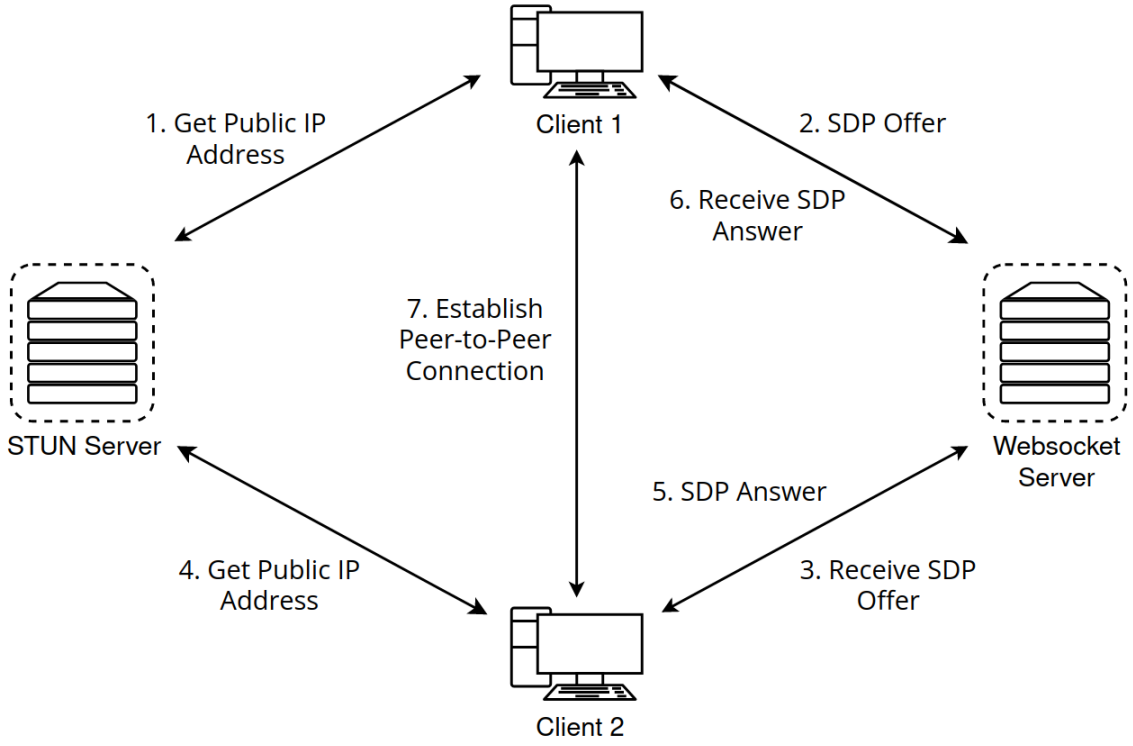


Figure 15: WebRTC connection establishment

A TURN server is used if there is a restrictive firewall preventing direct communication between the two clients, forcing a ‘middleman’ server to relay communication.

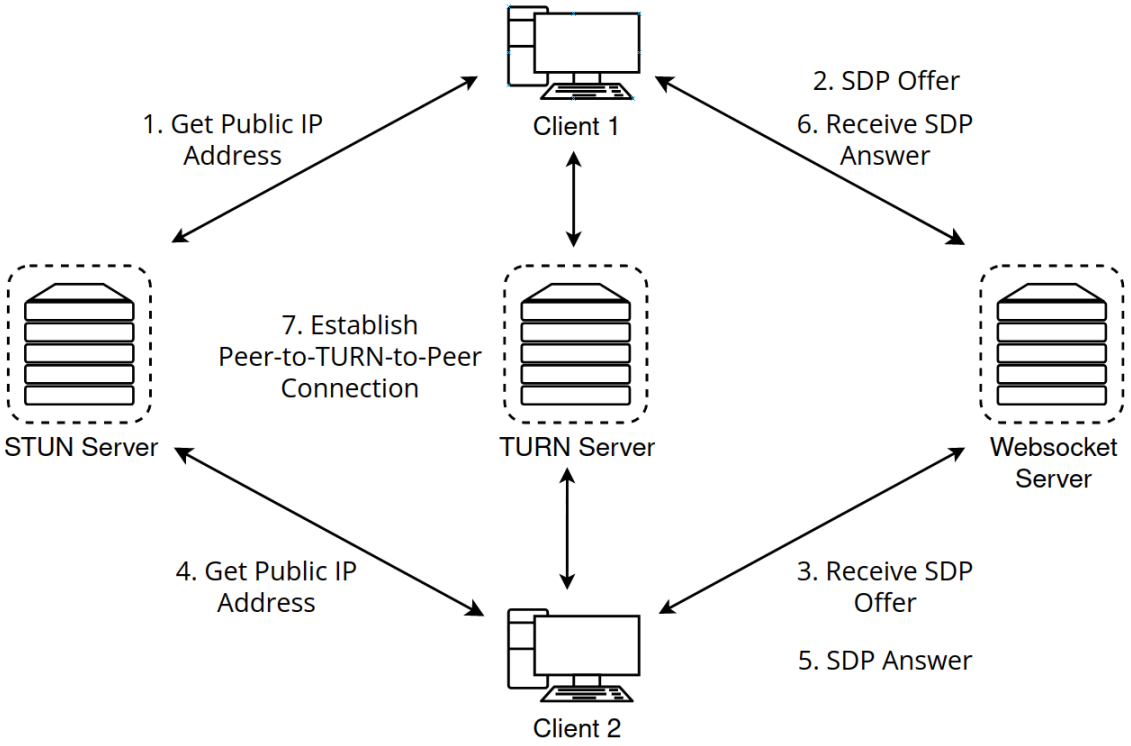


Figure 16: WebRTC connection establishment via TURN server

4.2.3 Web Server

A web server was chosen and designed with the purpose of supporting the system on the browser; as such it required to be able to send JSON responses for API requests and HTML responses for users visiting web pages appropriately. This logic was abstracted by using a URL prefix (as seen in Figure 17) to divide the two processes.

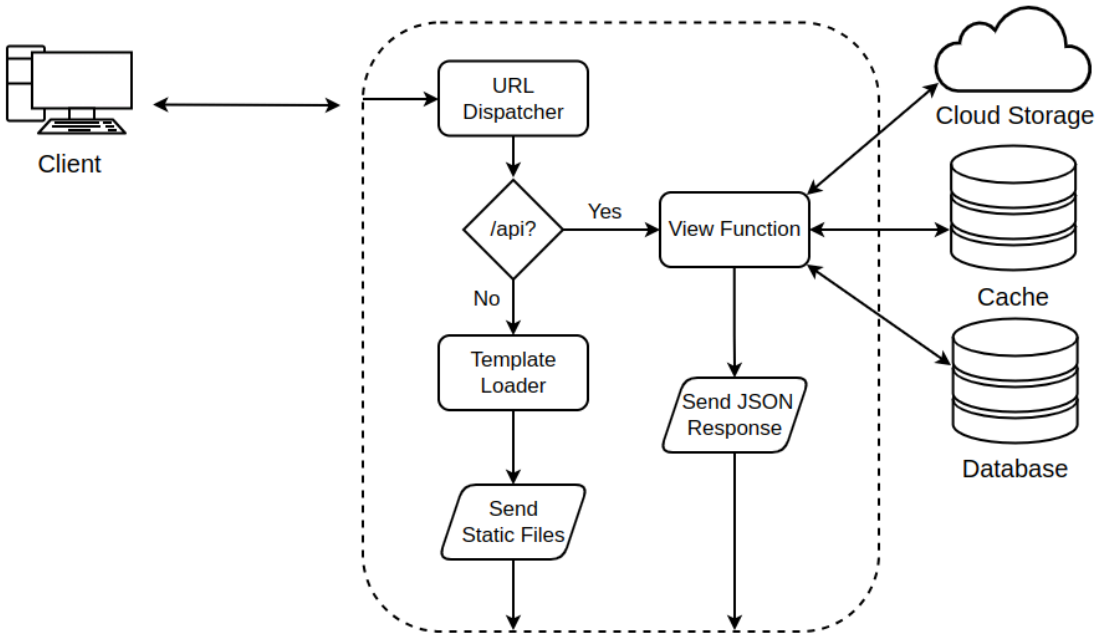


Figure 17: Web Server Architecture

A REST API was chosen and designed to represent data to the client. This was selected due to the conforming nature of REST API resources with the system's dataset. In addition, the common REST HTTP methods would all be required in the system. A summary of the API can be seen below, a full API diagram can be found in Appendix F.

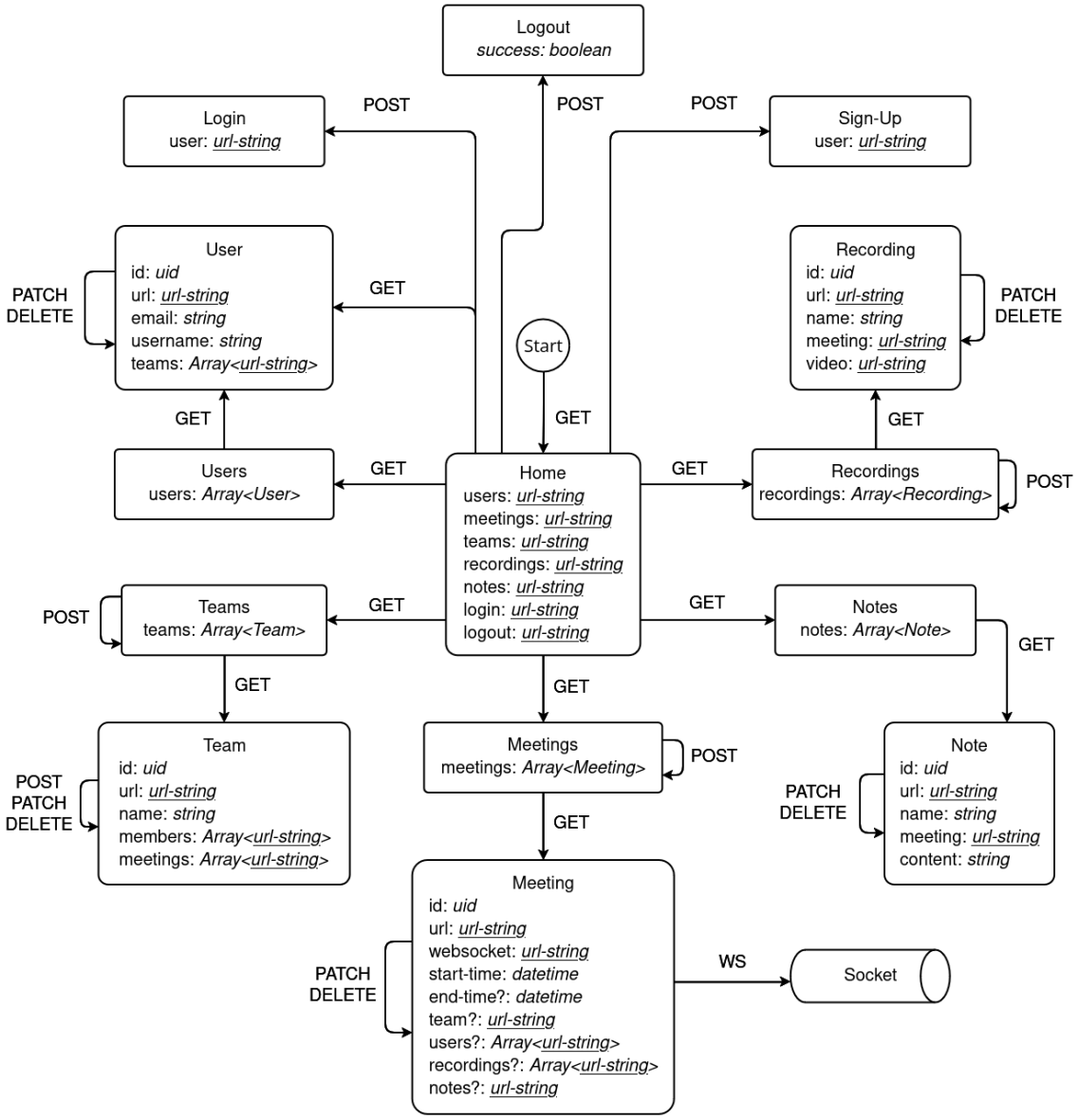


Figure 18: API Diagram

4.2.4 Websocket Server

As mentioned previously, a websocket server was chosen to enable peer-to-peer clients to establish a connection between themselves. Another reason for the choice of a websocket server was to allow for real-time communication in a meeting, namely group notes.

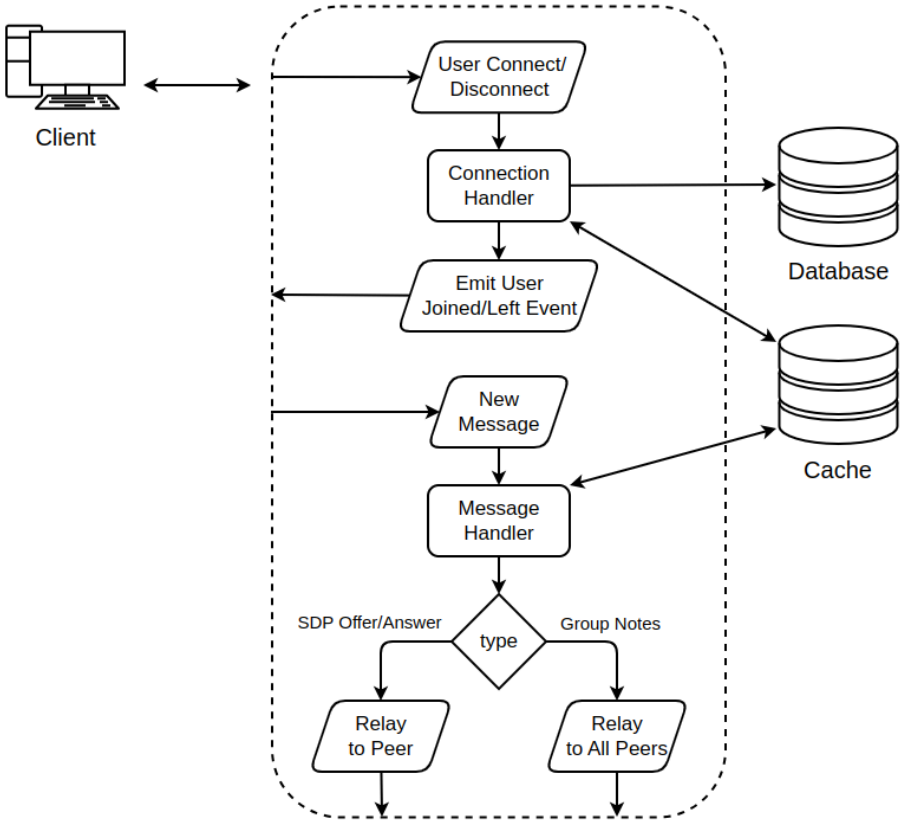


Figure 19: Websocket Server Architecture

In order to facilitate two clients initialising a connection, there needed to be two types of messages handled by the server: the first message would need to let a peer know about the existence of other peers in a room and the other message would need to let peers send SDP offer/answers between themselves. A full list of websocket events can be found in Appendix G.

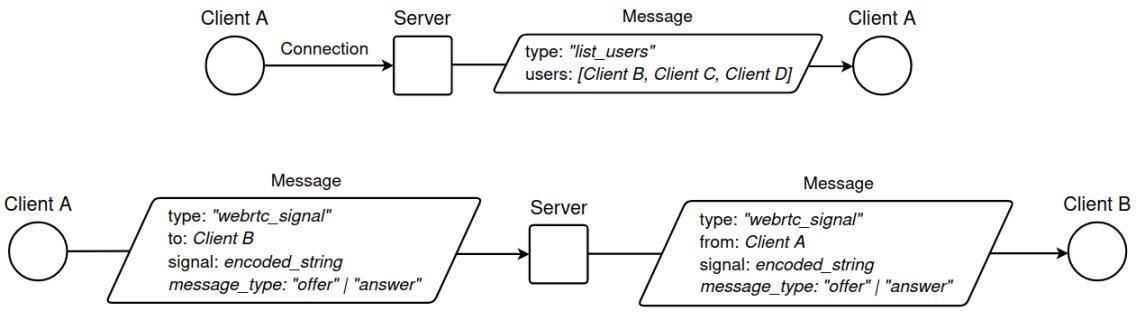


Figure 20: Websocket Server Events

4.2.5 Video-Processing Worker

In order to facilitate video recordings of meetings, a video-processing worker was required to be designed which could transform a video and save it to cloud storage whilst the web server still remained free to handle requests. The design of said worker can be found below.

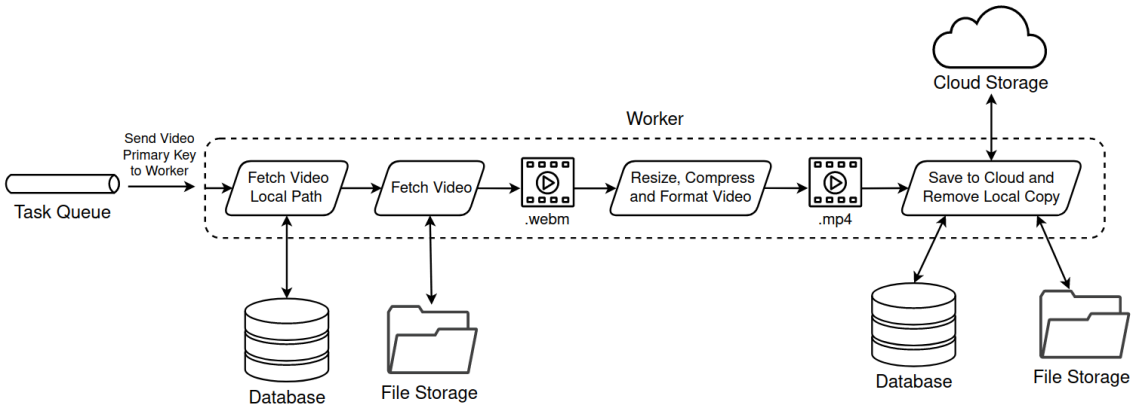


Figure 21: Video-Processing Worker Architecture

4.2.6 Relational Database

A relational database was chosen to store the data of the system as opposed to a non-relational database due to all the entities of data being interconnected. Using a non-relational database would mean data would need to be repeated which works well for a read-heavy system but the functionality of constant creation of meetings, teams, recordings and notes all lead to a write-heavy system which would have suffered in a non-relational database environment. Figures 22 and 23 detail the design process of creating a relational database schema.

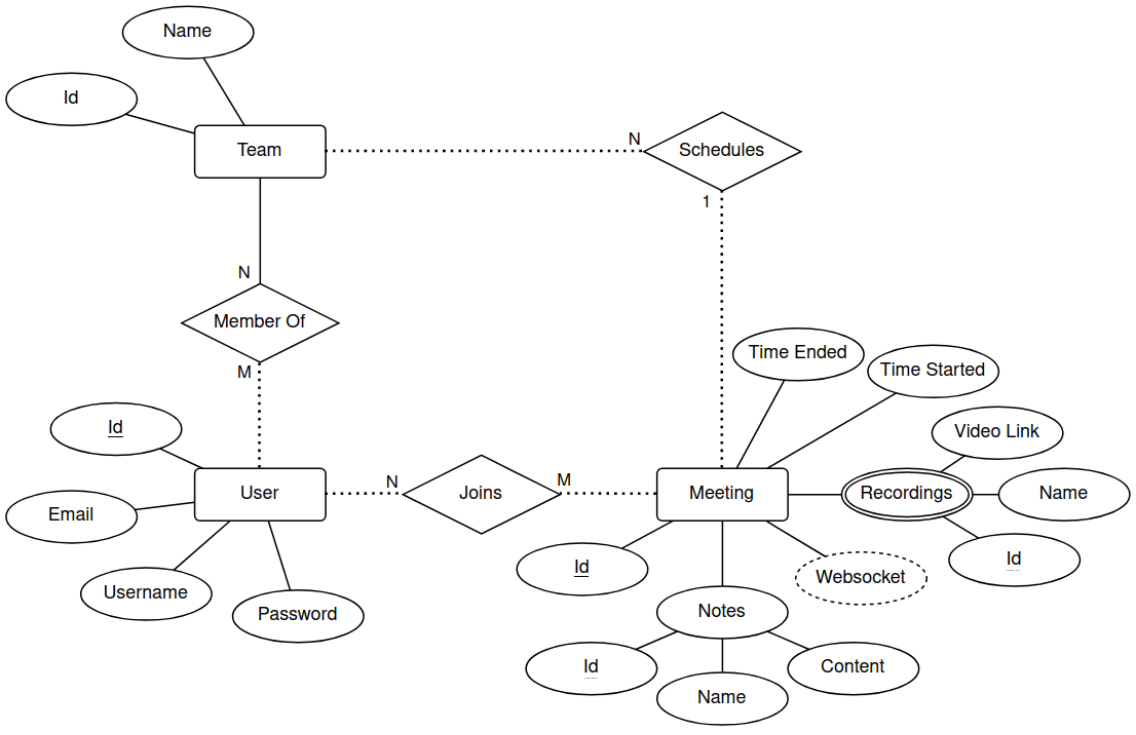


Figure 22: Database Diagram (Chen Notation)

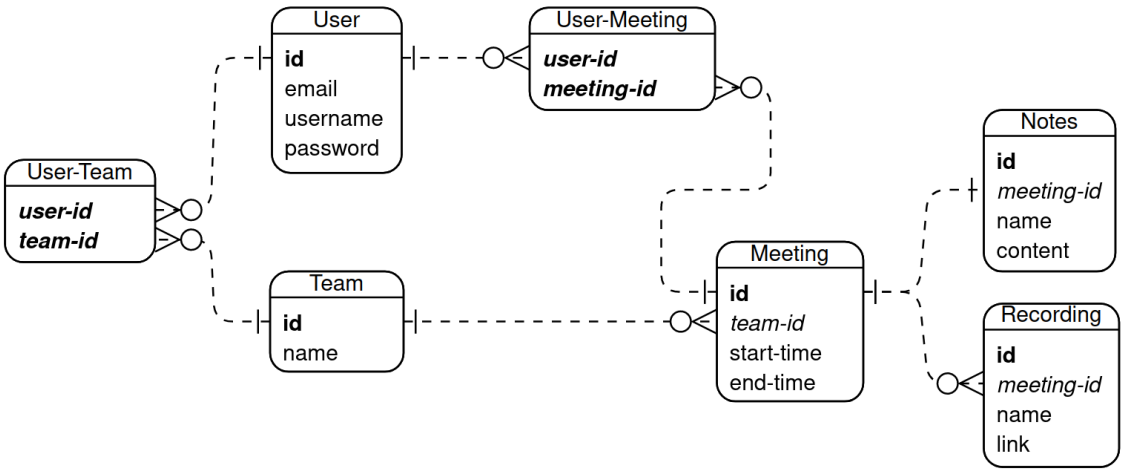


Figure 23: ER (Entity Relationship) Diagram

4.2.7 Cache Diagram

In order to optimise the system's speed, a cache was designed to be utilised by most microservices.

The cache was designed to store session data for:

- Logged-in users (used by the web server)
- Meeting metadata, participant data and meeting notes for active meetings (used by the websocket server)
- Data to be used by the task queue (used by the worker process).

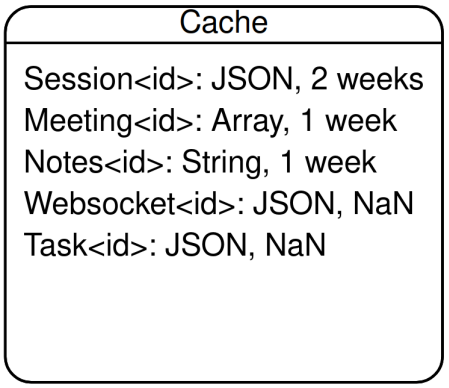


Figure 24: Cache Structure

Chapter 4: **Implementation**

This chapter details the process of implementing the design decisions from the previous chapter into a successfully functioning system. Included are the technologies chosen as well as code blocks detailing key concepts.

4.1 Languages and Frameworks

4.1.1 Python and Django

The Django framework (written in Python) was chosen to implement the backend of the system. Being a 'batteries-included' framework, focus was allowed to be given to implementing design choices detailed in the previous chapter as opposed to writing code from the ground up. The modularity of the framework also allowed for all the code to be written in one project and then deployed as separate microservices, providing a balance of abstraction of concerns and the utilisation of common data structures.

4.1.2 Typescript and React

The React library (written in Typescript) was chosen to implement the frontend of the system. Developed by Meta (formerly known as Facebook), it is one of the most popular frontend frameworks and consequently was a reason for the choice - as there are many third-party open-source libraries which can be utilised without having to be built from scratch. Typescript was chosen over Javascript ensuring type safety which facilitated a less error-prone end product.

4.2 Core Libraries and APIs

4.2.1 Django REST Framework

Django REST Framework was used to implement the API for the web server. Accompanied with Django's ORM (Object Relational Mapper), the abstraction of database queries and request authentication allowed for the direct implementation of the API design without having to implement low-level code. An example of how the REST User Resource was implemented can be seen below.

```
class UserSerializer(HashedIdModelSerializer):
  class Meta:
    model = User
    fields = [
      "id",
      "url",
      "email",
      "username",
      "password",
    ]
    extra_kwargs = {
```

```
      "password": {"write_only": True},
      "url": {"lookup_field": "hashed_id"},
    }

class UserViewSet(HashedIdModelViewSet):
  queryset = User.objects.all()
  serializer_class = UserSerializer
  permission_classes = [IsAuthenticated, IsUserOrReadOnly]
  http_method_names = ["get", "patch", "delete", "head", "options"]
```

Figure 25: Django REST Framework code example

4.2.2 Gunicorn

Gunicorn was used to handle HTTP requests and responses for the server and relied on using a WSGI (a Python specific protocol used to specify how a web server interacts with a web application) version of the project. This was chosen as WSGI applications are synchronous and are, therefore, most efficient for requests which don't require a lot of processing (such as returning static files or database queries). Gunicorn was run as a web server for our Django project simply by pointing it to the location of our WSGI application, as seen below.

```
$ gunicorn core.wsgi:application
```

Figure 26: Gunicorn code example

4.2.3 Channels

Django Channels was used to implement the websocket server functionality. Abstracting lifecycle events into core methods (connect, disconnect and receive), sending and receiving messages to clients was implemented by using a "type" identifier followed by the message contents - an example of which is seen below.

```
def connect(self):
  """Handle new connection to meeting including validation and auth."""
  # …
  # send new connection their meeting details
  self.send_json(content={"type": "your_connection", "user": self.user})

  # send new connection current members
  self.send_json(
    content={
      "type": "all_users",
      "users": current_participants,
    }
  )
  # …

def disconnect(self, close_code):
  """Handle dropped connection."""
  # …
  # broadcast user left to group
```

```
    async_to_sync(self.channel_layer.group_send)(
      self.meeting_id,
      {
        "type": "user.left",
        "user": self.user,
      },
    )
    # …

  def receive_json(self, content):
    """Handle connection messages."""
    # …
    if content["type"] == "webrtc_signal":
      # …
      async_to_sync(self.channel_layer.send)(
        content["to"],
        {
          "type": "user.signal",
          "from": self.user,
          "signal": content["signal"],
          "message_type": content["message_type"],
        },
      )
    elif content["type"] == "group_notes":
      # …
```

Figure 27: Django Channels code example

4.2.4 Daphne

Daphne was used as the websocket server and relied on using an ASGI version of the project. This was necessary as ASGI applications are asynchronous (as opposed to WSGI applications) which is important for long-life connections such as those of websockets. Being able to handle multiple connections simultaneously allows the server to make use of ‘dead time’ where there are no events from one connection and switch to another ‘live’ connection. Daphne was created as a web server by pointing it towards the ASGI version of our application as seen in Figure 28.

```
$ daphne core.asgi:application
```

Figure 28: Daphne code example

4.2.5 FFmpeg

FFmpeg-Python (a Python wrapper of FFmpeg) was used in the video-processing worker in order to transform the incoming recordings into an appropriate video file. This was done by creating a separate video and audio stream, resizing and changing the framerate of the video stream, and then merging the streams together with specified codecs and quality (as seen below).

```
def transform_temp_upload(self):
  # …
  video = (
    ffmpeg.input(self.temp_upload.path)
    .video.filter("scale", w=720, h=-2)
    .filter("fps", fps=30, round="up")
  )

  audio = ffmpeg.input(self.temp_upload.path).audio

  ffmpeg.output(
    video,
    audio,
    temp_output_path,
    vcodec="libx264",
    crf=24,
    acodec="aac"
  ).run()
  # …
```

Figure 29: FFmpeg-Python code example

4.2.6 Celery

Celery was chosen as the task queue to communicate between the web server and video-processing worker via the cache. This was done by triggering the worker any time a new recording was made available to be processed using Django's signal system as seen in Figure 30.

```
@receiver(post_save, sender=Recording)
def post_save_recording_handler(sender, instance, *args, **kwargs):
  if instance.temp_upload and not instance.upload:
    transform_video.delay(instance.pk)
```

Figure 30: Django Signals code example

The worker would then attempt to transform the uploaded video with a given number of retries inside a Celery Task as seen below.

```
@shared_task(bind=True)
def transform_video(self, instance_pk: int):
  """Compress and resize video file then move to permanent storage"""

  # get instance from id sent by the queue (celery)
  recording = Recording.objects.get(pk=instance_pk)

  try:
    recording.transform_temp_upload()
  except ffmpeg.Error as e:
    raise self.retry(exc=e, countdown=5, max_retries=3)
```

Figure 31: Celery Task code example

4.2.7 Simple Peer

Simple-Peer was used to implement the WebRTC protocol client-side. A peer would be initialised for each participant in the meeting and then an SDP offer would be sent to them via the websocket server. Upon receiving an SDP answer and establishing a connection, streams would then be sent bi-laterally.

```
const createPeer = (
  channel: string,
  initiator: boolean = false,
) => {
  const peer = new Peer({
    initiator: initiator || undefined,
    trickle: false,
    stream: localParticipantStream.current || undefined,
    config: { iceServers: getRelayServers() },
  });
  peer.on("signal", (data) => {
    websocket.current!.send(
      JSON.stringify({
        type: "webrtc_signal",
        to: channel,
        signal: JSON.stringify(data),
        message_type: initiator ? "offer" : "answer",
      }),
    );
  });

  peer.on("stream", (stream) => {
    for (const s of participantStreams.current) {
      if (s.channel === channel) {
        s.stream = stream;
        // …
      }
    }
  });

  return peer;
};
```

Figure 32: Simple-Peer code example

4.2.8 Editor.js

Editor.js was used to implement the group notes feature. Being a feature-rich third-party library, the only addition that was required was the live-streaming aspect (different tools such as paragraphs, tables, lists were already provided). To implement this, whenever data was added to Editor.js, a websocket message was sent containing the new notes' contents; this message was then relayed to the remaining meeting participants who would get the new notes instantaneously.

```
const editor = new EditorJS({
  holder: "editorjs",
  data: JSON.parse(groupNotes.current),
  onChange: () => {
    editor.saver.save().then((data) => {
      // …
      websocket.current.send(
        JSON.stringify({
          type: "group_notes",
          content: JSON.stringify(data),
        }),
      );
    });
  },
  tools: {
    paragraph: {
      class: Paragraph,
      inlineToolbar: true,
      config: {
        placeholder: "Add text here",
      },
    },
    // …
  }
});
```

Figure 33: Editor.js code example

4.2.9 React Query

React Query was also used in the frontend in order to provide caching functionality. This was important to limit the number of requests a client makes and to reduce loading times in between pages. This was implemented by using a ‘query key’ for each unique API request as seen below.

```
const query = useQuery({
  queryKey: ["profile"],
  queryFn: () => axios.get<{user: User}>("/api/who_am_i/")
                      .then((res) => res.data),
  // 1 min
  staleTime: 1 * 60 * 1000,
  // …
});
```

Figure 34: React Query code example

4.2.10 Tailwind

The CSS library Tailwind was also used throughout the project as a way to implement a consistent UI/UX theme throughout the website.

4.3 External Tools and Technologies

4.3.1 Version Control

Git and GitHub were used throughout the project as version control. This allowed breaking changes throughout the development phase to be undone seamlessly and also allowed the creation of feature branches for testing purposes while maintaining a working version of the project as the master branch. GitHub also allowed the cloning of the project in different machines which was important when deploying the project (as mentioned below).

4.3.2 Docker

Docker was used to facilitate the microservice functionality of the system. Paired with using the django framework, most of the microservices could be built using the same image and simply started with a different command. Using Docker Compose facilitated the use of separate frontend and backend networks to enhance the security of the system as well as manage all the services in an abstracted way. A simplified version of the docker-compose file is detailed below.

```
services:
  proxy:
    image: nginx:1.25.3-alpine3.18
    networks:
      - frontend
  web:
    image: dictate
    command: "gunicorn core.wsgi:application"
    volumes:
      - recording-temp-storage:/opt/dictate/media
    networks:
      - frontend
      - backend
  websocket:
    image: dictate
    command: "daphne -b 0.0.0.0 -p 80 core.asgi:application"
    networks:
      - frontend
      - backend
  worker:
    image: dictate
    command: "celery -A core worker -l INFO --concurrency 2"
    volumes:
      - recording-temp-storage:/opt/dictate/media
    networks:
      - backend
  relay:
    image: coturn/coturn:alpine3.18
    networks:
      - frontend
volumes:
```

```
  recording-temp-storage:
    driver: local
networks:
  frontend:
  backend:
```

Figure 35: docker-compose.yaml simplified

4.3.3 NGINX

NGINX was chosen as a reverse-proxy and load balancer for the microservice architecture, routing requests to the corresponding forward-facing servers (web server, websocket server and relay server). This was implemented using the official NGINX Docker Image with a custom nginx configuration file (as seen below).

```
http {
  server {
    listen 443 ssl;
    # …

    location / {
      proxy_pass http://app;
      # …
    }
    location /ws/ {
      proxy_pass http://websocket;
      # …
    }
  }
}

stream {
  server {
    # STUN Ports
    listen 3478 udp;
    listen 5349;
    # TURN Ports
    listen 49160-49163 udp;
    # …

    proxy_pass relay:$server_port;
  }
}
```

Figure 36: nginx.conf simplified

4.3.4 Coturn

Coturn was selected to be used as the STUN/TURN server due to being a free, open-source project which was fully functional. The configuration of the server was made using a default configuration file with some adjustments made such as limiting the number of TURN ports available and by passing command line

arguments to the docker image specifying the details of the database to be used.

4.4 Implementation of Core Functionality

4.4.1 Video-conferencing

In order to implement the video-conferencing behaviour of the system, we firstly needed to gain access to the clients local video and audio stream (which was then sent to each peer in the meeting).

```
navigator.mediaDevices.getUserMedia({ video: true, audio: true })
  .then((stream) => {
    // …
    localParticipantStream.current = stream;
    for (const p of participantStreams.current) {
      p.peer?.addStream(stream);
    }
    // …
  });
```

Figure 37: Obtaining local video and audio streams code example

After the local stream was obtained via the browser and the peer streams obtained through WebRTC connections, the final step was to display them on the screen which was done by laying out an appropriate number of HTML video tags (each stream belonging to a unique video tag).

```
const PeerVideo: FC<Props> = ({ stream, participant, ...props }) => {
  const videoRef = useRef<HTMLVideoElement | null>(null);
  // …
  useEffect(() => {
    // …
    if (videoRef.current && stream) {
      videoRef.current.srcObject = stream;
      // …
    }
    // …
  }, [stream]);

  return (
    <Video
      videoMuted={participant ? participant.videoMuted : true}
      username={participant?.username || "Guest"}
      videoRef={videoRef}
      {...props}
    />
  );
};
```

Figure 38: Displaying WebRTC peer in browser code example

4.4.2 Group Notes

The group notes feature was implemented using the websocket server and Editor.js libraries as mentioned above. The react-resizable-panels library was used in order to display the notes in the save window as the video-conference as seen below.

```
<PanelGroup direction="horizontal">
  <Panel defaultSize={50}>
    <Suspense fallback={<LoadingIcon className={`/* … */`} />}>
      <Notes
        groupNotes={groupNotes}
        groupNotesState={groupNotesState}
        websocket={websocket}
      />
    </Suspense>
  </Panel>
  <PanelResizeHandle className={`/* … */`} />
  <Panel>
    <VideoGrid
      participants={participants}
      participantStreams={participantStreams}
      gotLocalStream={gotLocalStream}
      localParticipant={localParticipant}
      localParticipantStream={localParticipantStream}
    />
  </Panel>
</PanelGroup>
```

Figure 39: Displaying group notes code example

4.4.3 Recordings

In order to capture recordings, the browser utility function getDisplayMedia was used. This allowed the client to capture the browser's video and audio which was then saved into an array of chunks.

```
const recorder = useRef<{
  stream: MediaStream | null;
  recorder: MediaRecorder | null;
  chunks: Blob[];
}>({ stream: null, recorder: null, chunks: [] });

navigator.mediaDevices
  .getDisplayMedia({
    audio: true,
    video: { displaySurface: "browser" },
    preferCurrentTab: true,
  })
  .then((stream) => {
    recorder.current.recorder = new MediaRecorder(stream, {
      audioBitsPerSecond: 128000,
      videoBitsPerSecond: 2500000,
```

```
      mimeType: "video/webm;codecs=vp8",
    });

    // store stream data
    recorder.current.recorder.ondataavailable = (e) => {
      if (e.data.size > 0) {
        recorder.current.chunks.push(e.data);
      }
    };
    // …
  }
  .catch(() => {
    // …
  });
```

Figure 40: Capturing browser's video and audio code example

After the capture had ended (either by the client leaving the meeting or pressing stop) the array of chunks was saved in as a file and then sent as POST data to the web server to then be processed by the video-processing worker (as seen previously).

```
const file = new File(
  recorder.current.chunks,
  `recording_${meetingId}_${Math.floor(Math.random() * 100)}.webm`,
  { type: "video/webm" },
);
```

Figure 41: Saving recording as webm file

4.4.4 Calendars

To finally implement the calendars functionality, first an array of sorted meetings was created after which could be displayed on a calendar a month at a time. This was achieved by getting all available meetings as an API request and then sorting and filtering the meetings into an array as seen below.

```
type SortedMeetings = {
  month: number;
  year: number;
  meetings: Meeting[];
};

const cleanedMeetings: SortedMeetings[] = useMemo(() => {
  let sortedMeetings: SortedMeetings[] = [];
  let filteredMeetings: Meeting[] = [];

  if (meetings) {
    if (teamFilter.value === "all") {
      filteredMeetings = meetings;
    } else if (teams) {
     // …
```

```
      filteredMeetings =
        meetings.filter((meeting) => teamMeetings.includes(meeting.url)) ||
        [];
    }
  }

  for (const meeting of filteredMeetings) {
    // ...
    let sortedMeeting = sortedMeetings.find(
      (v) => v.month === month && v.year === year,
    );
    if (
      teamFilter.value === "all" ||
      teamFilter.value === meeting.team?.split("/").slice(-2, -1)[0]
    ) {
      if (sortedMeeting) {
        sortedMeeting.meetings.push(meeting);
      } else {
        sortedMeetings.push({ month, year, meetings: [meeting] });
      }
    }
  }
  return sortedMeetings;
}
```

Figure 42: Generating sorted meeting array for calendars code example

4.5 Page Designs

The implementation of the frontend page designs can be seen below. A full list of which can be found in Appendix H.

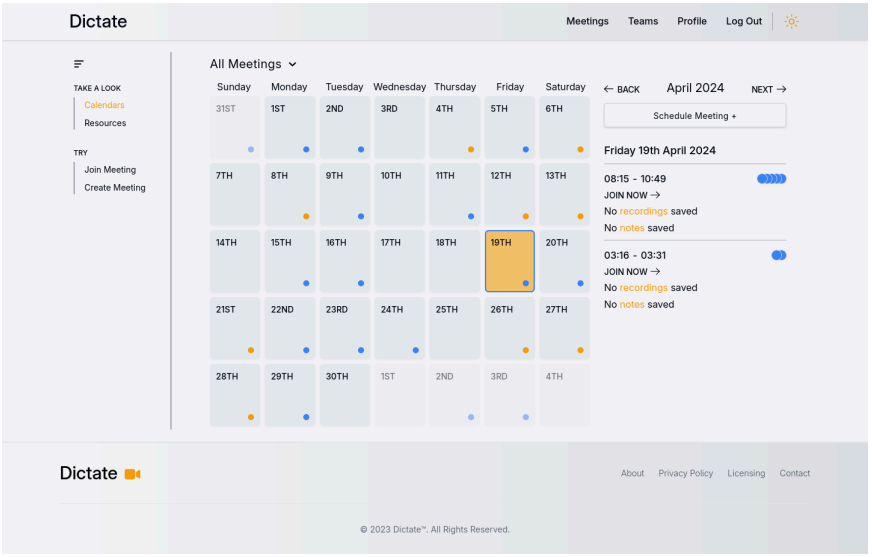


Figure 43: Calendars Page

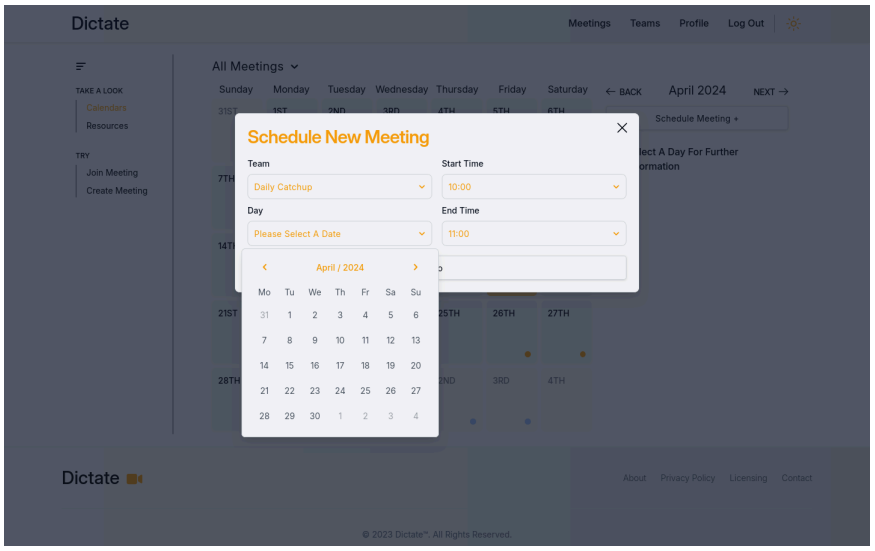


Figure 44: Schedule Meeting Modal, Calendars Page

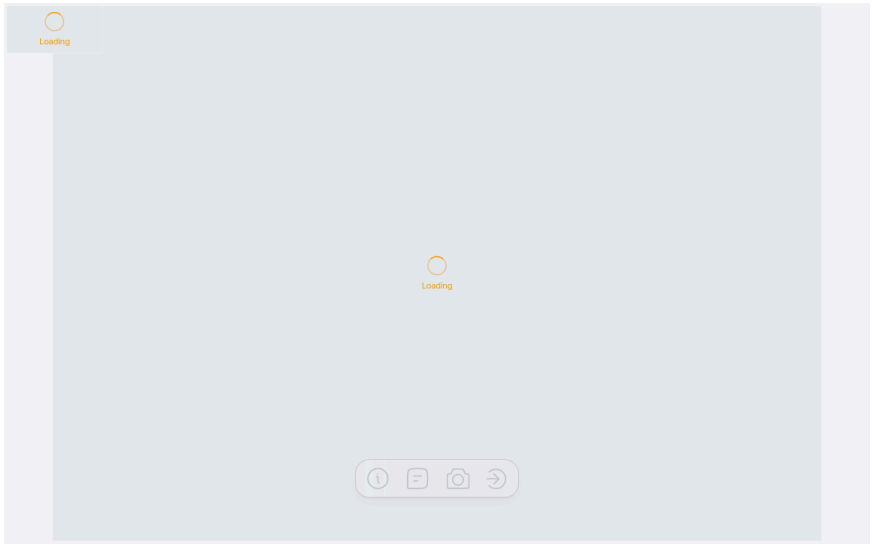


Figure 45: Video-Conference Meeting Page

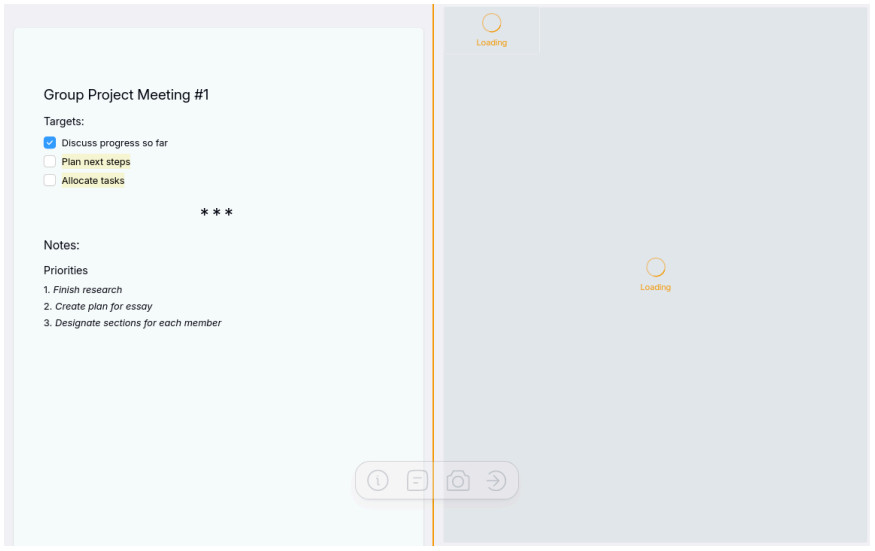


Figure 46: Group Notes, Video-Conference Meeting Page

4.6 Deployment

4.6.1 Amazon Web Services

AWS (Amazon Web Services) was chosen to deploy the system to the cloud. From the benefits of the service was the ability to also deploy the database, cache and cloud storage all together. This also allowed the creation of custom security groups limiting access to the database and cache to solely the EC2 instance (the virtual machine hosting our microservices). Elastic IP was also set up (pointing to our EC2 instance) in order to provide a stable IP address - in case the virtual machine stopped/failed and needed to be restarted with a different IP.

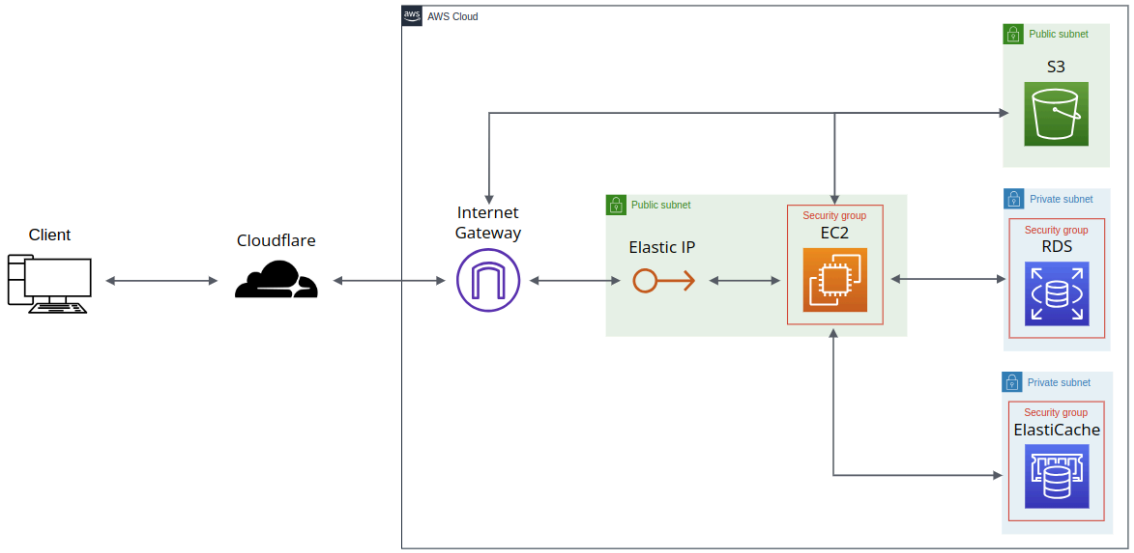


Figure 47: Deployment Architecture

Using Docker Compose to build the application meant that the deployment of the system in our virtual machine was seamless since the containerisation ensured a consistent environment for our system both in development and production. To start our microservices all that was required was to pull the repository from GitHub and then run the docker-compose.yaml file as a background process.

4.6.2 Cloudflare

Cloudflare was chosen to provide a domain name for the application to run in the browser as well as create DNS records for a group of sub-domains as detailed below.

DNS Record Type	Name	Value
A	dictate.com	(Elastic IP Address)
A	www.dictate.com	(Elastic IP Address)
A	ssh.dictate.com	(Elastic IP Address)
CNAME	stun.dictate.com	dictate.com

CNAME	turn.dictate.com	dictate.com

Table 5: Deployment DNS Records

TLS was also set up in addition to creating a domain name via the official docker image of certbot, this was achieved through running an nginx service and then the certbot service to create the website's certificates as seen below.

```
$ docker run -p 80:80 -p 443:443 \
    -v $(pwd)/nginx/nginx.certbot.conf:/etc/nginx/nginx.conf:ro \
    -v $(pwd)/certbot/www/:/var/www/certbot/:ro \
    -v $(pwd)/certbot/conf/:/etc/nginx/ssl/:ro \
    nginx:1.25.3-alpine3.18

$ docker run --rm -it \
    -v $(pwd)/certbot/www/:/var/www/certbot/:rw \
    -v $(pwd)/certbot/conf/:/etc/letsencrypt/:rw \
    certbot/certbot:latest certonly --webroot \
    --webroot-path /var/www/certbot/ \
    -d <domain_name>
```

Figure 48: TLS certificate creation code example

Chapter 5: **Validation**

This chapter details the steps taken to validate the video-conferencing system including software and end-user testing. This was accomplished with a mixture of unit and end-to-end software tests and a public survey which was completed after testing the system.

5.1 Software Testing

A series of unit tests and end-to-end tests were written and carried out upon both the backend and frontend of the system. A full list of test cases can be found in Appendix I.

5.1.1 Backend Testing

Backend testing of the web server, websocket server, task queue and video-processing worker was implemented through the Django wrapper of Python's unittest framework. To test the API endpoints of the web server, Django REST Framework's wrapper of Django's Unit Tests was used as seen in the example below to mimic client JSON requests. Unit Testing was chosen for the backend as opposed to integration testing to allow for errors to be granularly detected.

```
class RecordingTests(APITestCase):
  # …
  @patch("api.signals.transform_video.delay")
  def test_create_recording_success(self, transform_video_fn):
    recordings_count = self.meeting_1.recordings.count()
    self.client.login(email="tester_1@dictate.com", password="12345678")

    url = reverse("recording-list")
    data = {
      "meeting": self.meeting_1.hashed_id,
      "recording": SimpleUploadedFile(
        "file.webm", b"fake_recording", "video/webm"
       ),
    }
    with self.settings(MEDIA_ROOT=self.TEMP_MEDIA_ROOT):
      response = self.client.post(url, data)

    self.assertEqual(response.status_code, status.HTTP_201_CREATED)
    self.assertEqual(
      Meeting.objects.get(pk=self.meeting_1.pk).recordings.count(),
      recordings_count + 1,
    )
    self.assertEqual(transform_video_fn.call_count, 1)
```

Figure 49: Django Unit Test example code

5.1.2 Frontend Testing

Frontend tests were written using the Cypress framework. This allowed for end-to-end testing to be written as code and executed in the browser, ensuring an accurate representation of an end-user's experience. An example of the login re-usable command and a profile page test is mentioned below.

```
Cypress.Commands.add("login", () => {
  cy.visit("/login");
  cy.getCookie("sessionid").should("not.exist");
  cy.getCookie("csrftoken").should("exist");
  cy.get("input[name=email]").type("tester@dictate.com");
  cy.get("input[name=password]").type("12345678");
  cy.get("button").contains("Log In").click();
  cy.url().should("include", "/calendars");
  cy.getCookie("sessionid").should("exist");
});

describe("profile page", () => {
  beforeEach(() => {
       cy.login();
       cy.visit("/profile");
  });
  it("shows correct profile information", () => {
    cy.get("h2")
      .contains("password: ****")
      .prev()
      .should("contain", "tester@dictate.com")
      .prev()
      .should("contain", "tester");
  });
  // …
});
```

Figure 50: Cypress testing code example

5.2 User Testing

5.2.1 Feedback Survey

A feedback survey was carried out across 13 participants who were given hands on time to try out the application. A series of closed and open ended questions were asked in order to obtain both a statistical and qualitative group of results. A full list of the questions asked with their answers can be found in Appendix J.

5.2.2 Analysis of Results

Across all 13 participants, all features of the platform were tried out across a range of desktop, mobile and tablet devices. All participants found the app easy to use, fast and responsive as well as maintain a good video and audio quality in the meetings (judged by a score > 5 out of 10). Bugs found by participants

included rare cases of meeting codes not working, the browser attempting to use the wrong camera and recordings not picking up audio.

Chapter 6: **Evaluation**

This chapter expounds upon the steps taken to evaluate the system in relation to the project objectives and research questions stated in the introductory chapter. Sources included a feedback survey and technical evaluation of design choices made.

6.1 Feedback Survey

The feedback survey referenced in section 5.2 and detailed in Appendix J was used to evaluate the system and perform a critical analysis of its features.

6.1.1 Analysis of Results

Feedback of the system was overall positive with 77% of users ranking the platform 8 or more out of 10 and 84.6% of users stating that the platform bettered their perception of video-conference technologies and that it increased their likelihood of choosing a video-conference solution in the future. This paired with 77% of answers strongly agreeing that the design of the platform positively affected their experience (judged by a grading > 6 out of 10) supported the research questions: “Would advancements in video-conference technologies push more users to select a video-conference meeting?” and “How does UI/UX design impact the user experience of video-conference meetings?”.

Feedback suggesting improvements in UI/UX design such as the use of coloured icons and tooltips further support this claim as it is predicted that further improving the UI would further increase users satisfaction with the system and if this were not the case, it would not have been brought up as a concern.

Participants of the survey testified to the system’s stable and fast video-conferencing environment (one of the key concerns brought up in the requirements survey) where all answers to whether they agreed the platform was fast and responsive and to whether they found the video-conference to have good video and audio quality were 7 or higher out of 10. In particular two respondents chose to specify this as a highlight about the platform, one stating: “both audio and video were surprisingly really good quality” while the other mentioned: “Easy to join the call, no login or lengthy process to join, video and audio quality was great”.

69.3% of respondents strongly agreed (9 or more out of 10) that the platform supported group work. 7 out of the 13 responses to the open-ended question “What functionalities (if any) did you like about the platform?” specifically stated the group notes or calendar functionalities. Combined, a positive correlation was found to answer the research question: “How can group teams and calendars affect the user experience of video-conference systems?”.

6.2 Summary

6.2.1 Strengths of the system

As mentioned by participants of the previously mentioned survey, key strengths of the system included its fast and responsive speeds. This could have been improved on by modifying the REST API to allow for more efficient queries. For example, for a client to get all meetings for a specific team it needs to make two queries: the first to get all meetings (/api/meetings/) and the second to get a list of meeting ids for a specific team (/api/teams/<id>/) and then filtering the first response through the second. This could be made more efficient by utilising a single endpoint and query parameters, for example /api/meetings?team=<id>/ - this would transfer the computational load of filtering and sorting meetings from the client to the server and so seemingly make the end-user sense a faster platform.

6.2.2 Weaknesses of the system

Key concerns highlighted in the feedback were related to the UI/UX design. This could have been improved from different angles such as: using more distinct colours, using tooltips for icons and by using a more engaging layout in the video-conference page. Concerns related to the limited functionality of the video-conference meeting also arose which could have been addressed by implementing a mute button, hide video button, dedicated chat feature and screen sharing functionality - all of which could have been implemented with the current system architecture but were simply overlooked. Overall the fixes to these weaknesses can all be facilitated in the platform relatively quickly due to the design choices made in the previous section and so also highlight the strength of the flexibility of the system.

Another key weakness of the system was the implementation of recordings being client-side as opposed to server-side. This means that theoretically a client can upload any video they like without verification and it will be saved to cloud storage. Ideally to implement recordings there would be an SFU or MCU topology where the recordings would take place server-side in the media server which would then guarantee the contents of the video as well as be able to adjust the layout of the video from the perspective of a particular person to a more general perspective.

Chapter 7: **Legal, Social & Ethical Issues**

This chapter highlights the legal, social and ethical issues pertaining to the video-conferencing platform as well as its sustainability.

7.1 Legal Issues

Since the platform stores personal information of users (username, email and video/audio recordings) there are legal rulings pertaining to its proper management and eventual deletion. For example GDPR (General Data Protection Regulation) requires personal data to be secured appropriately and if requested to be deleted, should be deleted within a month.

7.2 Social Issues

Since the platform allows communication between 2 or more participants via video, audio and text, it is important to take into account the conditions of both parties. Should it be allowed for participants of ages under 16 years old to join meetings where there are older participants? Enforcing this would require users to create an account before joining a meeting and to specify their age. This functionality was not implemented in the platform but is subject to further review.

7.3 Ethical Issues

Following from the previous point, there are clear ethical issues pertaining to video-conference meetings being used for illegal and immoral activities such as intoxicants, drugs, gambling, violence, etc. in addition for these meetings to be recorded and then stored in the platform's cloud storage. Whilst the platform may not necessarily be legally liable for this, it is important to consider avenues to avoid such behaviour such as implementing a terms of service agreement before use of the software and the potential use of AI to detect illegal activities in any recordings stored by the platform.

7.4 Sustainability

The AWS services used for deployment can be cleared and reused for any future projects, alternatively more resources can be added to them to continue to support a larger quantity of users to the platform.

Chapter 8: **Conclusions**

This chapter highlights personal achievements, challenges and future work in regards to the project.

8.1 Achievements

Through the project I have achieved multiple aims I had from the start of the project. I learned new technologies such as Docker, NGINX, websockets and task queues as well as new languages such as Typescript. I also deployed the app to a production environment using AWS services and enabled TLS authentication.

Importantly I learnt how to do technical research pertaining to the design phase of a product life cycle, comparing alternative technologies and selecting a specific design pattern.

8.2 Challenges

Key challenges faced included having to learn, understand and subsequently compare technologies in a field I did not previously know in a short space of time. I was successful in doing this for the WebRTC technologies (STUN/TURN server and websocket server) but in the client topology design phase I did not have the resources nor time to learn how to implement a media server (SFU or MCU) and so was limited to using a mesh topology.

Another key challenge was the limited time: implementing a full scale web app from start to finish with documentation was challenging and so naturally left areas I would have liked to work more on but left off due to time constraints.

8.3 Future Work

Areas of future work pertaining to this project include the further development of functionality to the video-conference page (screen sharing, mute, whiteboards, chat, etc.) as well as the introduction of an SFU to the architecture to facilitate larger meetings and better recording quality.

References

Adriel Aseniero, B., Constantinides, M., Joglekar, S., Zhou, K. and Quercia, D. (2020). MeetCues: Supporting Online Meetings Experience. 2020 IEEE Visualization Conference (VIS). Salt Lake City, UT, USA, 2020, pp. 236-240.

Ahyia Adu-Oppong, A. & Agyin-Birikorang, E. (2014). Communication in the Workplace: Guidelines for Improving Effectiveness. Global Journal of Commerce & Management Perspective, vol. 3(5), pp. 208-213.

Allen, J.A., Tong, J. & Landowski, N. (2021). Meeting effectiveness and task performance: meeting size matters. Journal of Management Development, vol. 40 (5), pp. 339-351.

Choren, A. (2015). The Importance of Communication in the Workplace. IEEE Potentials, vol. 34 (3), pp.10-11.

Cohen, M.A., Rogelberg, S.G., Allen, J.A. & Luong, A. (2011). Meeting design characteristics and attendee perceptions of staff/team meeting quality. Group Dynamics: Theory, Research, and Practice, vol. 15, pp. 90-104.

Denstadli, J.M., Julsrud, T.E. & Hjorthol, R.J. (2011). Videoconferencing as a Mode of Communication. Journal of Business and Technical Communication, vol. 26 (1), pp. 65-91.

Dialpad (2022). The State of Video Conferencing 2022 [online]. Available at: https://www.dialpad.com/blog/video-conferencing-report [Accessed 25th November 2023].

Doodle (2019). The Doodle State of Meetings Report 2019 [online]. Available at: https://assets.ctfassets.net/p24lh3qexxeo/axrPjsBSD1bLp2HYEqoij/d2f08c2aaf5a6ed80ee53b5ad7631494/Meeting_Report_2019.pdf [Accessed 25th November 2023].

Fisher, P.H. (1953). An analysis of the primary group. Sociometry, vol. 16, pp. 272-276.

Fortune Business Insights (2022). Video Conferencing Market Size, Share, Trends and Growth, 2026 [online]. Available at: https://www.fortunebusinessinsights.com/industry-reports/video-conferencing-market-100293 [Accessed 25th November 2023].

Fulk, J. & Collins-Jarvis, L. (2001). Wired meetings: technological mediation of organizational gatherings. In: Jablin, F.M., Putnam, L.L. (Eds.), The New Handbook of Organizational Communication. Sage, Thousand Oaks, CA, pp. 624-703.

Google (2024). Google Meet [online]. Available at: https://meet.google.com [Accessed 26th April 2024].

Kauffeld, S. & Lehmann-Willenbrock, N. (2012). Meetings matter: Effects of team meetings on team and organizational success. Small Group Research, vol. 43, pp. 130-158.

Kloppenborg, T.J. & Petrick, J.A. (1999). Meeting management and group character development. Journal of Managerial Issues vol. 11 (2), pp. 166-179.

Lundgren, D.C. & Bogart, D.H. (1974). Group size, member dissatisfaction, and group radicalism. Human Relations, vol. 27, pp. 339-355.

McKinsey Global Institute (2012). The social economy: Unlocking value and productivity through social technologies [online]. Available at: https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/the-social-economy [Accessed 25th November 2023].

Microsoft (2024). What is Microsoft Teams? [online]. Available at: https://support.microsoft.com/en-gb/topic/what-is-microsoft-teams-3de4d369-0167-8def-b93b-0eb5286d7a29 [Accessed 26th April 2024].

O'Farrell, M. & Bates, J. (2009). Student information behaviours during group projects. Aslib Proceedings, vol. 61 (3), pp. 302-315.

Rogelberg, S.G., Scott, C. & Kello, J. (2007). The science and fiction of meetings. MIT Sloan Management Review, vol. 48 (2), pp. 18-21.

Schwartzman, H.B. (1989). The Meeting: Gatherings in Organizations and Communities. New York: Plenum Press, p. 7.

Seashore, S.E. (1954). Group Cohesiveness in the Industrial Work Group. Ann Arbor: University of Michigan Press.

Steiner, I. (1972). Group Processes and Productivity. New York: Academic Press.

Statista (2023). Global market share of videoconferencing software 2023, by program [online]. Available at: https://www.statista.com/statistics/1331323/videoconferencing-market-share [Accessed 26th October 2023].

Stefik, M., Foster, G., Bobrow, D.G., Kahn, K., Lanning, S. & Suchman, L. (1987). Beyond the chalkboard: computer support for collaboration and problem solving in meetings. Communications of the ACM, vol. 30 (1), pp. 32-47.

Zapier (2024). Google Meet vs. Zoom: Which video conferencing platform is better? [online]. Available at: https://zapier.com/blog/google-meet-vs-zoom/ [Accessed 26th April 2024].

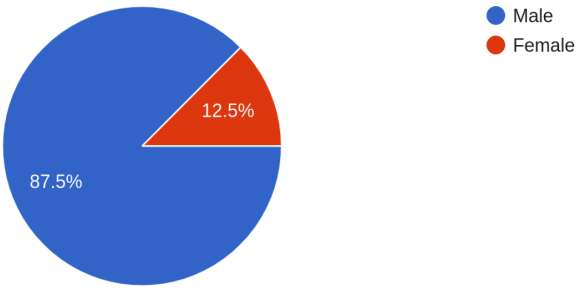
Zapier (2022). Zoom vs. Teams: Which is better for remote meetings and collaboration? [online]. Available at: https://zapier.com/blog/zoom-vs-teams/ [Accessed 26th April 2024].

Zoom (2024). About Zoom [online]. Available at: https://www.zoom.com/en/about/ [Accessed 26th April 2024].

Appendix A: **Research Survey**

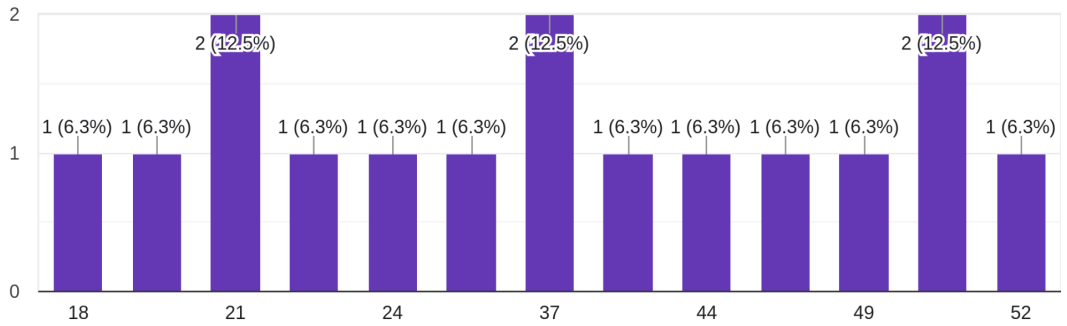
Gender

16 responses



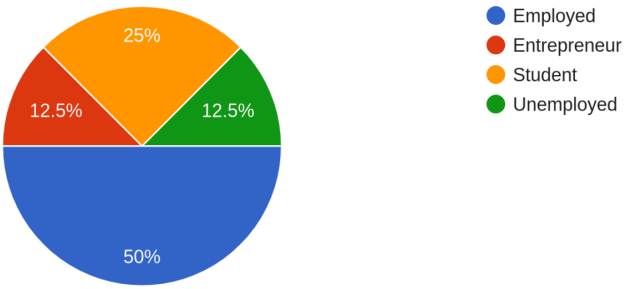
Age

16 responses



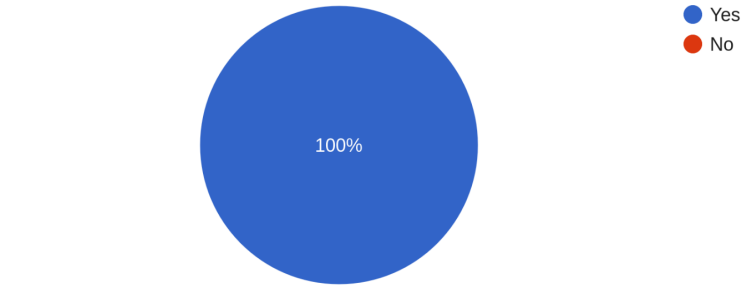
Occupation

16 responses



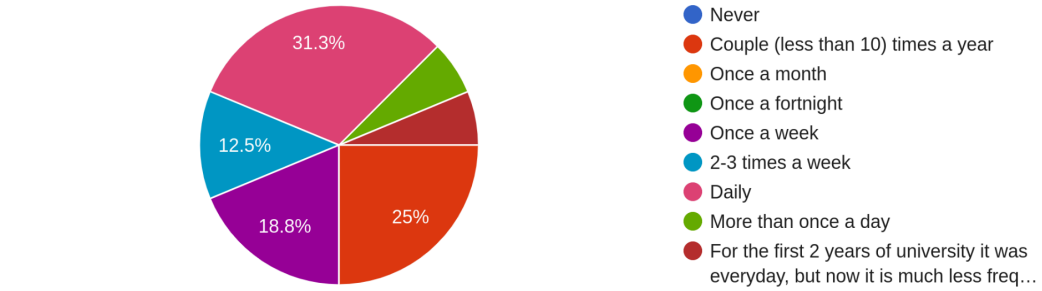
Do you use video-conferencing at work/school? (e.g. zoom)

16 responses



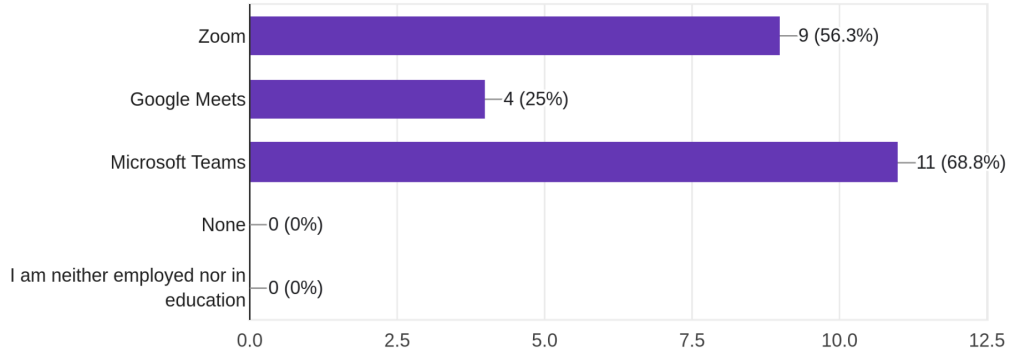
How often do you use video-conferencing technologies at work/school?

16 responses



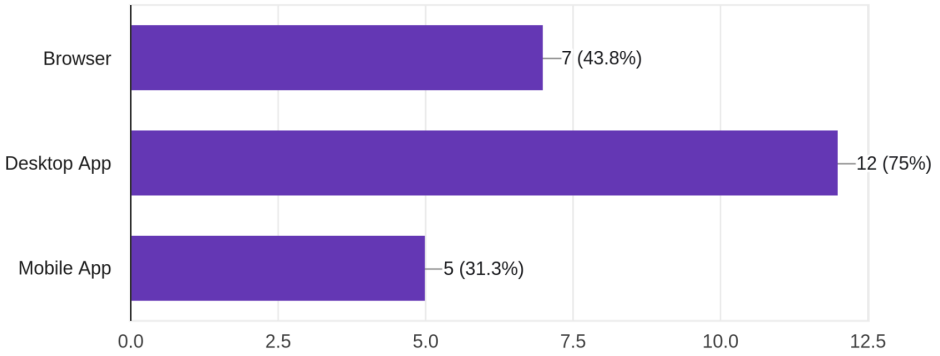
Which video-conference technologies do you use at work/school?

16 responses



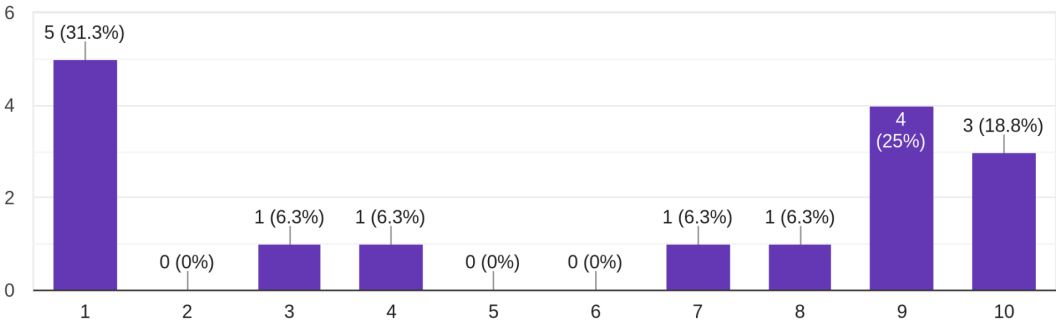
On which platform(s) do you use video-conference technologies?

16 responses



Do you use additional features when in said video-conferences? (e.g.virtual whiteboard, screen sharing, etc.)

16 responses



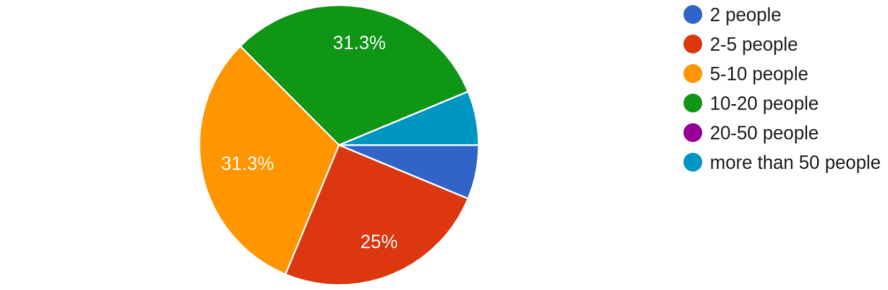
If yes, please mention which additional features you use.

8 responses

- Whiteboard, screen share, chat, recording
- screen sharing, whiteboards
- Screen sharing is used pretty much every time for presentations and lectures. Sometimes a virtual whiteboard was also used.
- Screen sharing, taking control of other screens, chat function, recording function
- Screen share, white board, chat box
- Screen share
- screen sharing/screen control
- Screen sharing, whiteboard, recording

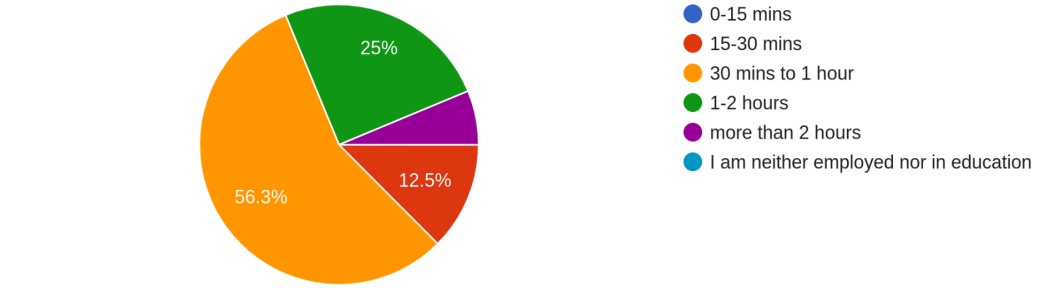
How large are the video-conferences you usually join at work/school?

16 responses



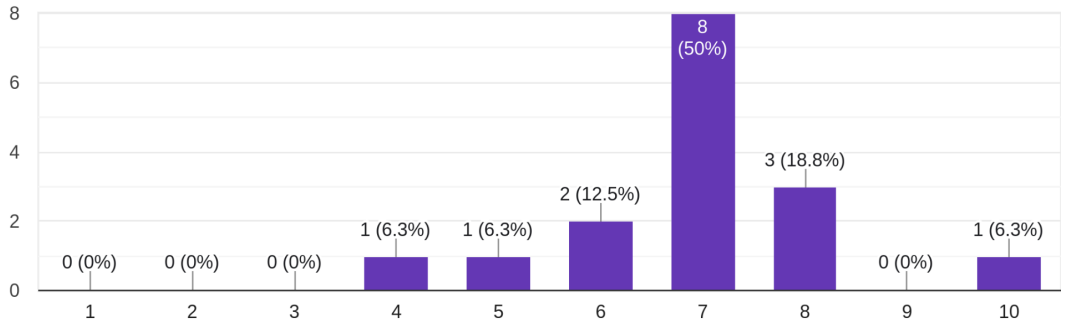
How long do your video-conferences usually last at work/school?

16 responses



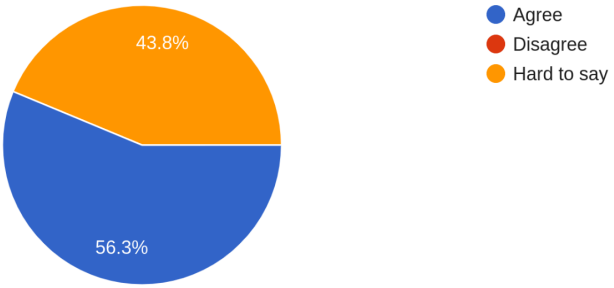
Do you like video-conferencing

16 responses



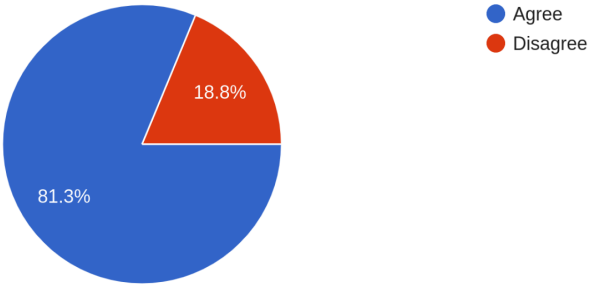
If video-conference technologies improved, I would use them more

16 responses



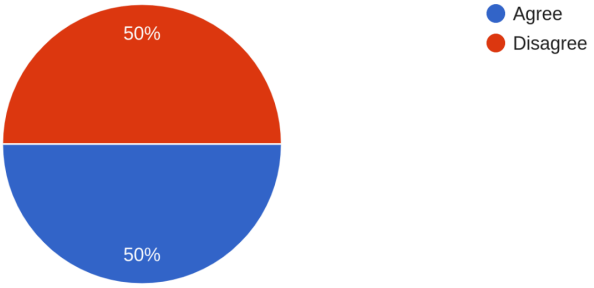
There is not a one-size-fits-all solution for video-conferencing (sometimes one app is better than another in different use cases)

16 responses



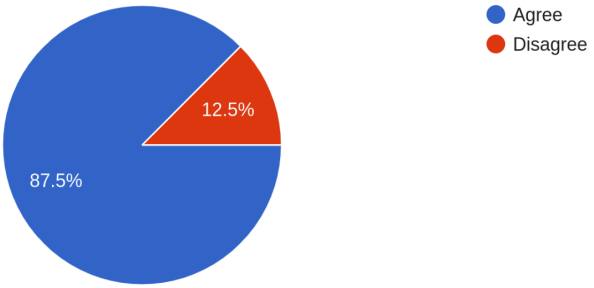
There is a good app which exists for large-scale video-conferencing (e.g. 50+ members)

16 responses



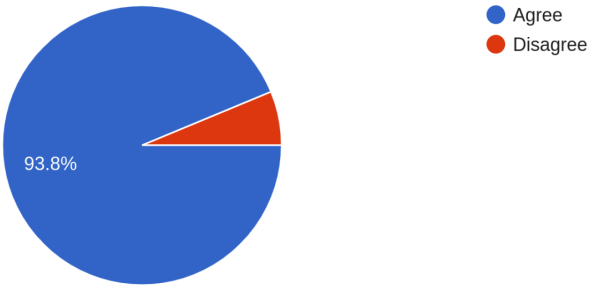
There is a good app which exists for small-scale video-conferencing (e.g. 1-5 members)

16 responses



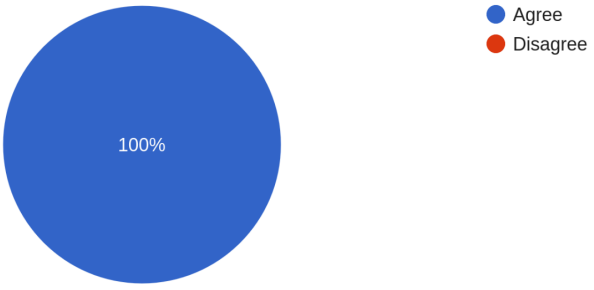
If I want to have a catch-up or meeting with my team there is a suitable software which exists for us to use

16 responses



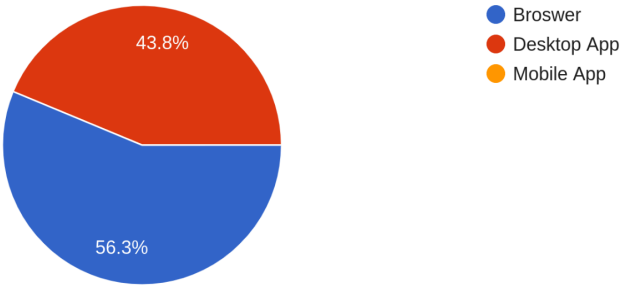
If I want to have a catch-up or meeting with a small group of people there is a suitable software which exists for us to use

16 responses



How would you prefer to access a video-conference on at work

16 responses



What problems do you have with current video-conference technologies? (e.g. zoom, skype, etc.) (state one problem)

16 responses

- None I can think of
- Laggy audio or video on poor networks
- Screen freezing
- Poor connect,no writing notes
- sometimes connection issues, lack of whiteboarding, screen sharing your entire desktop!
- External light sources affect image quality unpredictably
- The audio can cut out and glitch a lot sometimes.
- None at the moment, MS Teams works fine for what I need it for
- Connection and lag
- Sounds quality. Interacting is slow, you have to open up reactions and choose
- audio problems
- User interface
- The buttons are complicated, installing a whole new app. Difficult to join meetings sometimes
- Not being able to see the screen that's being shared.
- cannot write up notes during video call
- No problems

What would you like to see improved in future video-conference technologies? (state one improvement)

16 responses

- Not sure zoom does what I need it to do
- Better network error correction
- Greater reliability of the software
- AI writing notes
- excellent collaboration tools - like interactive whiteboards
- AI-assisted lighting enhancement
- The ability to use the app with worse internet speeds.
- N/A
- Stability

- Open documents in the app.
- stream quality
- Better UI
- Easier to access meetings
- Sound quality for large conferences.
- multi-functionality
- Pretty good for my needs

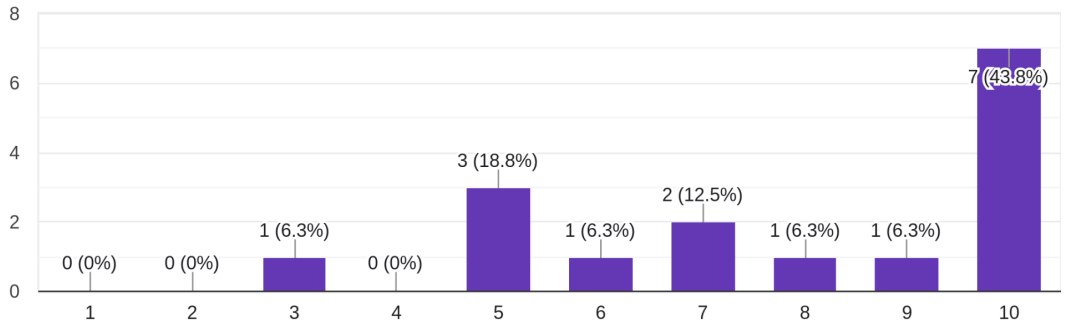
What built-in functionalities would you most like to see from a video-conference technology? (state one functionality)

16 responses

- Integration with go high level
- Desktop sharing, whiteboard, avatars, custom backgrounds,
- Greater array different backgrounds
- N/a
- meeting transcricpts
- Effective whiteboeading
- Being able to sync calendars with your peers.
- N/A
- Not sure
- White board, screen sharing, opening documents
- live collaboration features eg. a text document everyone can type on
- .
- More interactivity with document sharing etc
- Not sure.
- transcribing (note taking)
- Screen share

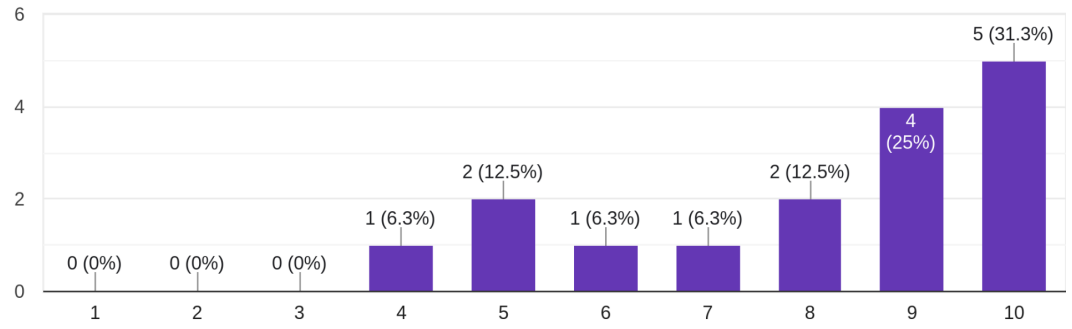
I would like team calendars in video-conference technologies to make it easier to schedule team meetings

16 responses



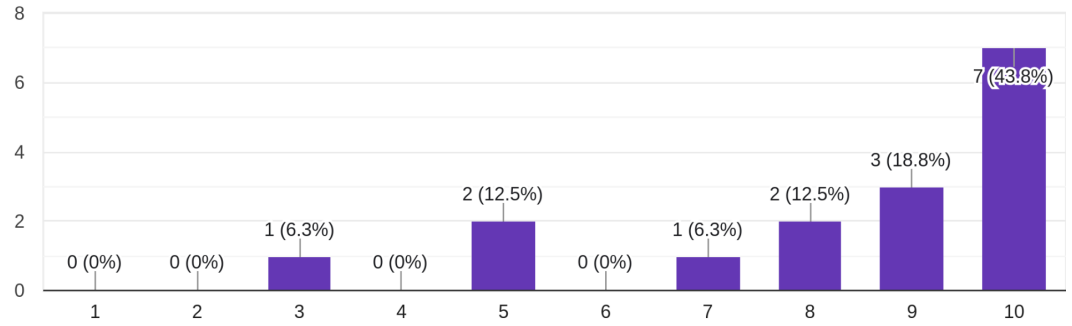
I would like recordings in video-conference technologies to make it easier to review

16 responses



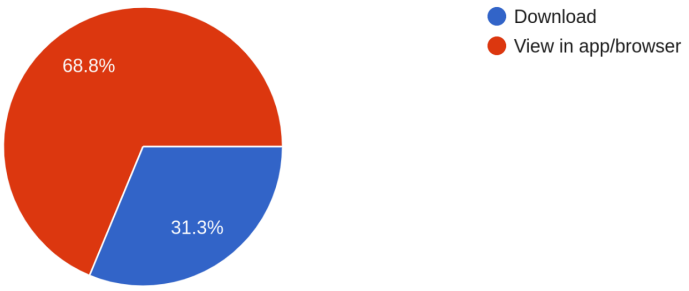
I would like live group note-taking in video-conference technologies so we can work together

16 responses



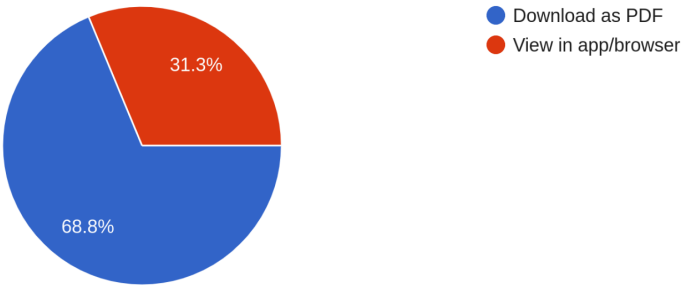
If you had meeting recordings, would you like the ability to download them on your local device or simply view them in the app?

16 responses



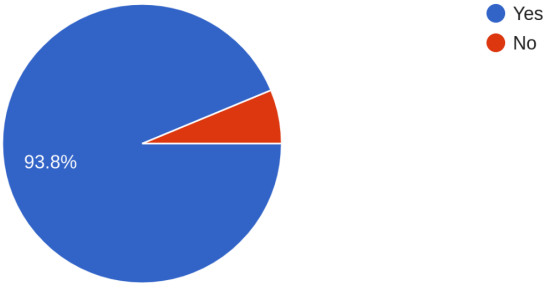
If you had team notes, would you like the ability to download the notes as a PDF afterwards or simply to view them in the app?

16 responses



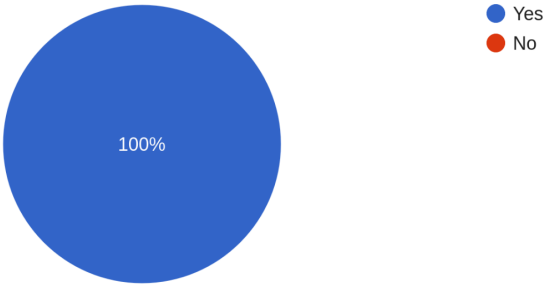
Would you like the ability to create an account in an app without entering an email and password (e.g. signing in via google, facebook, twitter, etc.)

16 responses



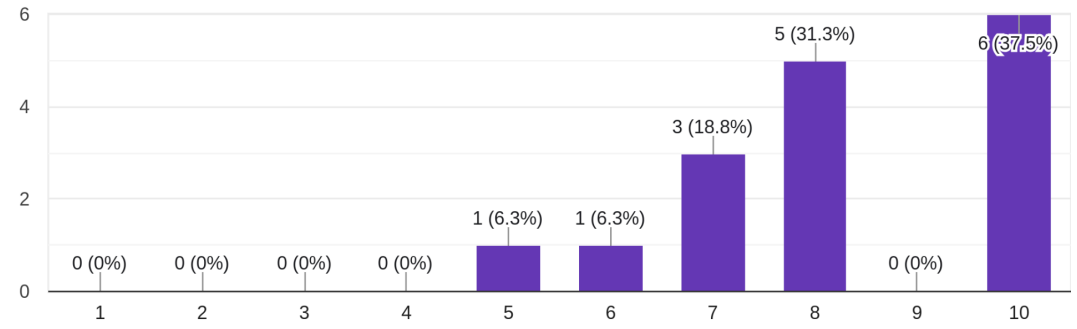
Would you like the ability to join a video-conference without a webcam?

16 responses



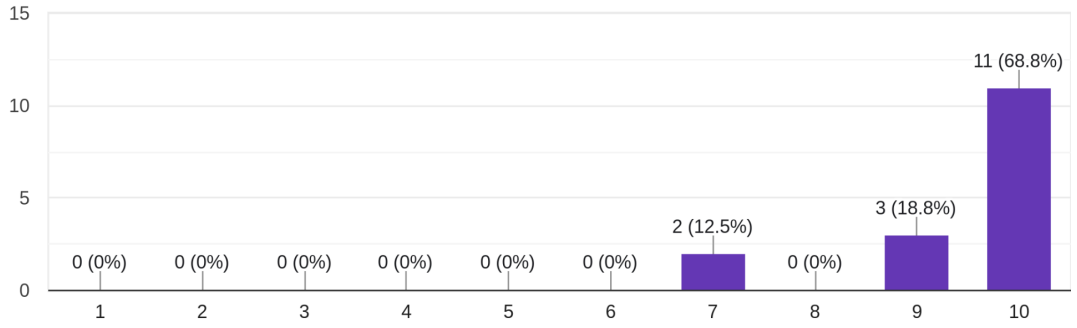
I would like to use an app which I could figure out how it works by myself (intuitively)

16 responses



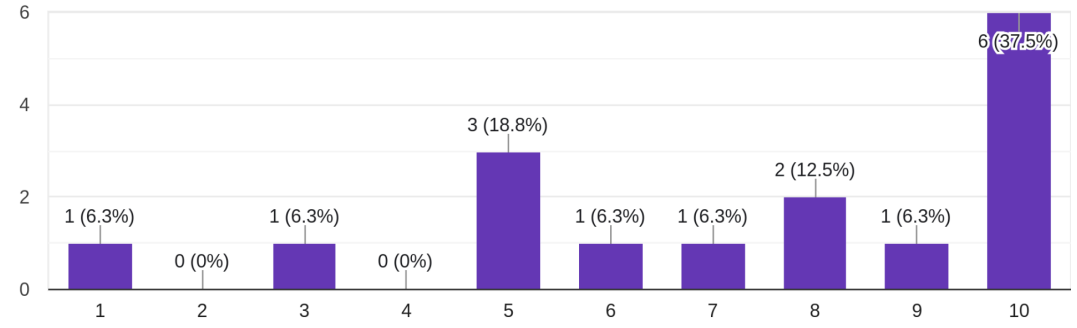
I value security and privacy in the apps I use

16 responses



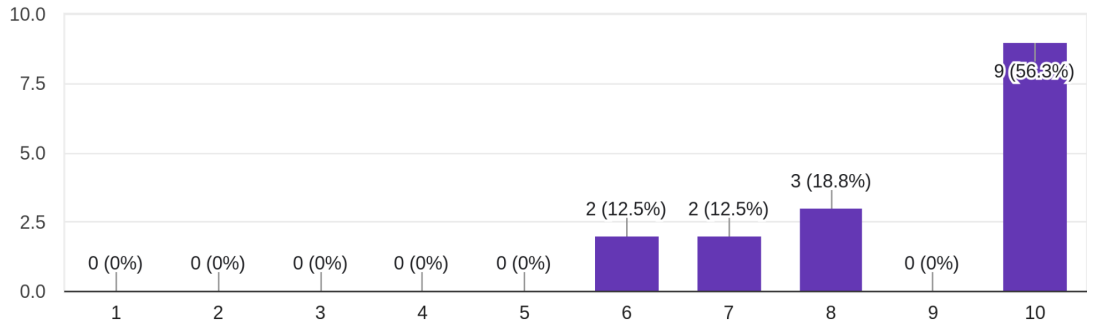
I value documentation in the apps I use (e.g. an instructions page)

16 responses



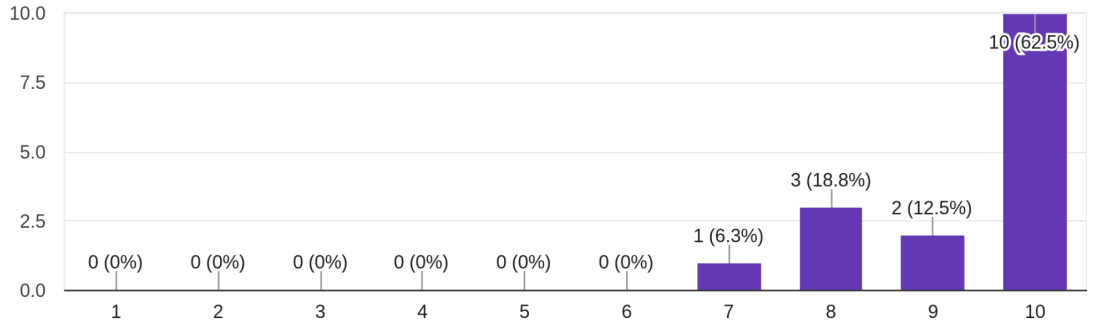
I value availability in the apps I use (e.g. a website not working because of maintenance, etc.)

16 responses



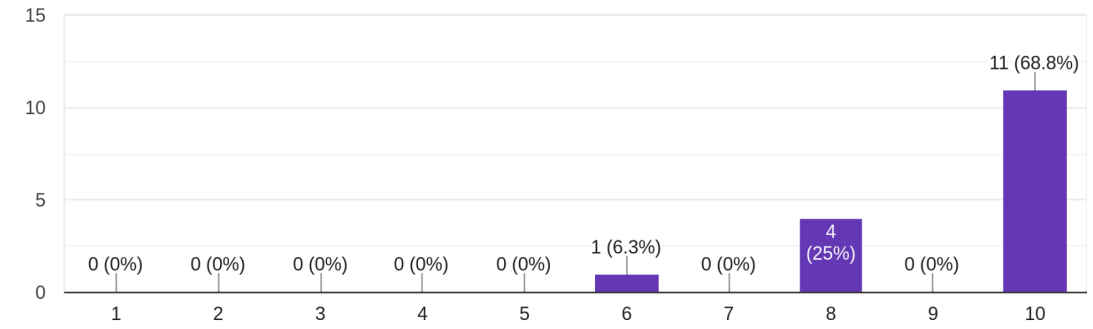
I value video and audio quality in video-conferences

16 responses



I value fast response times in the apps I use

16 responses



Appendix B: **Use Cases**

Use Case	Log In
Description	The Log In Use Case provides the functionality of logging a Guest into their user profile and allowing access to the internal features of the system such as Calendars, Profiles, etc.
Actors	Guest
Basic Flow	1. Guest clicks Login and is redirected to Log In Page
	2. Guest enters email and password
	3. Guest clicks Login button
	4. System authenticates credentials
	5. Guest becomes Logged In User and redirected to the homepage
Alternative Flows	None
Exception Flows	3.1. Guest entered invalid email format or password (e.g. email doesn't include '@' or password is less than 5 characters)
	3.2. System tells Guest to enter valid inputs
	3.3. Use Case returns back to step 2
	4.1. System cannot verify user with credentials entered
	4.2. System tells Guest that the login attempt was invalid and asks them to try again
	4.3 Use Case returns back to step 2
Special Requirements	1. The System should prevent a Guest from attempting to log in after 5 failed attempts until a timeout of 5 mins has passed
Pre-Conditions	1. The Guest must be verified to a be a Guest and not have an active Logged In session
Post-Conditions	1. The Guest must be logged into the same profile as their credentials entered
Extension Points	1. Guests are able to reset their password via email

Use Case 1

Use Case	Sign Up
Description	The Sign Up Use Case allows a Guest User to create an account and log in as a Logged In User which allows access to the internal features of the system such as Calendars, Profiles, etc.

Actors	Guest
Basic Flow	1. Guest clicks Sign Up and is redirected to Sign Up Page
	2. Guest enters email and password
	3. Guest enters profile details (username)
	4. Guest clicks Sign Up button
	5. System sends verification email and prompts Guest to check for aforementioned email
	6. Guest clicks on link in verification email to verify new account
Alternative Flows	None
Exception Flows	3.1. Guest entered invalid email format or password (e.g. email = dave.com or password < 5 characters)
	3.2. System tells Guest to enter valid inputs
	3.3. Use Case returns back to step 2
	4.1. System is unable to send verification email
	4.2. Guest is told to try again later or contact support
	4.3. Use Case ends
	5.1. Guest cannot access verification email
	5.2. Use Case ends
Special Requirements	1. A Guest should not be able to sign up with an email that is already in use in the system
	2. The verification email sent by the System should contain a link redirecting the user back to the login page and prompt the user to sign in with new account details
Pre-Conditions	1. The Guest must be verified to a be a Guest and not have an active Logged In session
	2. The Guest must have access to their email account to verify their new account
Post-Conditions	1. The Guest must be able to log into their profile with the same credentials they entered in the Sign Up form
	2. The Guest's profile must display the same information as they entered in the login form
Extension Points	None

Use Case 2

Use Case	Join Meeting
Description	The Join Meeting Use Case provides the functionality for Guest Users and Logged in Users to join an active video-conference session
Actors	Guest, Logged In User
Basic Flow	1. User clicks Join Meeting 2. User enters Meeting Id 3. System adds User to video-conference
Alternative Flows	None
Exception Flows	2.1. User enters invalid Meeting Id 2.2. System informs User of invalid Meeting Id and asks to try again 2.3. Use Case returns to step 2
Special Requirements	1. Guest Users should be prompted to enter their name to be displayed in the meeting
Pre-Conditions	1. The User must allow browser access to camera and microphone 2. The meeting must not have reached its capacity (30 members)
Post-Conditions	1. Users should be displayed in the meeting with either the name attached to their account (Logged In Users) or with the name inputted on entry (Guest Users)
Extension Points	1. Logged In Users are able to record the meeting by clicking the record button 2. Users are able to access group notes by clicking the notes button. 3. Users are able to leave the meeting

Use Case 3

Use Case	Create Meeting
Description	The Create Meeting Use Case provides the functionality for Users to start a new video-conference session
Actors	Guest, Logged In User
Basic Flow	1. User clicks Start Meeting 2. System adds User to new meeting
Alternative Flows	None
Exception Flows	None
Special Requirements	1. System should inform the User of the Meeting Id

	2. Guest Users should be prompted to enter their name to be displayed in the meeting
Pre-Conditions	1. The User must allow browser access to camera and microphone
Post-Conditions	1. The Logged In User should be displayed in the meeting with the name attached to their account
	2. Guests and Logged In Users should be able to join the meeting with the 'Join Meeting Use Case' with the Meeting Id provided
Extension Points	1. Logged In Users are able to record the meeting by clicking the record button
	2. Users are able to access group notes by clicking the notes button.

Use Case 4

Use Case	View Calendar
Description	The View Calendar Use Case allows Logged In Users to view their personal calendar, allowing them to view scheduled meetings, schedule new meetings and view meeting recordings and notes
Actors	Logged In User
Basic Flow	1. Logged In User clicks My Calendars
	2. System navigates Logged In User to Calendars Page
Alternative Flows	None
Exception Flows	None
Special Requirements	1. The calendars page should display the current month when first loading
Pre-Conditions	None
Post-Conditions	1. The Logged In User should be able to see all scheduled meetings in their calendar
	2. The Logged In User should be able to navigate to different months of the year (past and future)
Extension Points	1. The Logged In User is be able to navigate to a calendar specific to a team, this calendar will only show scheduled meetings and resources for that team
	2. The Logged In User is be able to schedule a meeting for a specific team (see Schedule Team Meeting Use Case)
	3. The Logged In User is be able to delete a scheduled meeting they created

	4. The Logged In User is be able to see a past meetings' recordings and notes (see View Meeting Files Use Case)

Use Case 5

Use Case	Schedule Team Meeting
Description	The Schedule Team Meeting Use Case enables Logged In Users to schedule a video conference meeting their team can then view in the team calendar
Actors	Logged In User
Basic Flow	1. Logged In User navigates to Calendars Page
	2. Logged In User clicks Schedule Team Meeting
	3. Logged In User selects which team the meeting is assigned to
	4. Logged In User selects date and time of meeting
	5. Logged In User clicks Add Meeting
Alternative Flows	1.1. Logged In User navigates to team-specific calendars page
	1.2. Logged In User clicks Schedule Team Meeting
	1.3. Use Case resumes at step 4
Exception Flows	4.1. Logged In User enters invalid date or time (e.g. date in past)
	4.2. System informs Logged In User of the error and asks to re-enter input
	4.3. Use Case repeats from step 4
Special Requirements	1. A meeting should only be able to be scheduled in 15 min intervals (e.g. 12:00, 12:15, 12:30, etc.)
	2. A meeting should only be able to be scheduled to last in 15 min intervals (i.e. lasting 15 mins, 30 mins, 45 mins, etc.)
Pre-Conditions	1. A Logged In User should only be able to schedule meetings for teams they are part of
Post-Conditions	1. All team members (including the Logged In User) should be able to see the newly scheduled meeting in the calendar
	2. All team members (including the Logged In User) should be able to join the scheduled meeting by clicking on the link
Extension Points	1. The Logged In User is able to delete their scheduled team meeting

	2. The Logged In User is able to edit their scheduled team meeting

Use Case 6

Use Case	View Meeting Files (Recording and Notes)
Description	The View Meeting Files Use Case enables Logged In Users to access a previous video-conference meetings' recordings and notes
Actors	Logged In User
Basic Flow	1. Logged In User navigates to Calendars Page
	2. Logged In User clicks on a scheduled meeting
	3. Logged In User clicks on view recording button/view notes button
Alternative Flows	1.1. Logged In User navigates to Resources Page
	1.2. Logged In User selects appropriate resource tab (either recording or notes)
	1.3. Logged In User selects desired resource to view
Exception Flows	2.1. Logged In User selects meeting which has not taken place yet or hasn't finished (i.e. scheduled in the future or currently taking place)
	2.2. System hides view recording and notes buttons
	2.3. Use Case ends
	3.1. Logged In User selects unavailable resource (i.e. meeting was not recorded or notes were not taken)
	3.2. System disables relevant button(s) (i.e. view recording button and/or view notes button)
	3.3. Use Case ends
Special Requirements	1. Clicking the view recording button should open the video as a pop-up window
	2. A meeting recording should have the option to be displayed full screen
	3. A meeting recording should start the player from the beginning (0:00)
	4. Viewing previous meeting notes should be read-only
	5. The resources page should be able to be filtered to show desired resources only
Pre-Conditions	1. Logged In Users should only be able to access notes and recordings from their own team meetings
Post-Conditions	1. The meeting recording and meeting notes displayed should be for the correct video-conference meeting

Extension Points	None

Use Case 7

Use Case	Join Team
Description	The Join Team Use Case provides the functionality for a Logged In User to join a team of other Logged In Users and subsequently gain access to their team calendar, scheduled team meetings and previous meeting resources (recordings and notes)
Actors	Logged In User
Basic Flow	1. Logged In User navigates to Teams Page 2. Logged In User clicks Join Team 3. Logged In User enters Team Id 4. System presents Logged In User with team preview (team name and members) 5. System asks Logged In User to confirm if they want to join the team 6. Logged In User confirms they want to join by clicking Join Now button
Alternative Flows	6.1. Logged In User clicks Cancel button 6.2. Use Case ends
Exception Flows	3.1. Logged In User enters invalid Team Id (e.g. Team Id doesn't exist, Logged In User is already part of the team, etc.) 3.2. System informs Logged In User of error and asks to re-enter a correct Team Id 3.3. Use Case returns back to step 2
Special Requirements	1. Only team members' first names and initial of their last names should be displayed in the team preview 2. The team preview should limit the members displayed to 5
Pre-Conditions	1. Only Logged In Users should be able to join teams 2. The team should not have reached its max limit of 30 members 3. The Logged In User should not have reached the max limit of number of joined teams (100 teams)
Post-Conditions	1. The Logged In User should be able to see the team added to the Teams Page 2. The Logged In User should be able to see the team calendar under the Calendars Page

	3. The Logged In User should have access to all team resources (recordings and notes)
	4. The Logged In User should be able to access all team meeting functionality (schedule a team meeting, join a team meeting, etc.)
Extension Points	1. The Logged In User is able to leave the joined team
	2. The Logged In User is able to view the members of the joined team

Use Case 8

Use Case	Create Team
Description	The Create Team Use Case provides the functionality for a Logged In User to create a new team
Actors	Logged In User
Basic Flow	1. Logged In User navigates to Teams Page
	2. Logged In User clicks Create Team
	3. Logged In User enters Team Name
	4. Logged In User clicks Create Team button
Alternative Flows	None
Exception Flows	3.1.1. Logged In User enters duplicate Team Name (i.e. they are already in a team which has that same name)
	3.1.2. System asks Logged In User if they want to continue with duplicate team name or change it
	3.1.3. Use Case ends
	3.2.1. Logged In User enters Team Name longer than limit (50 characters)
	3.2.2. System informs Logged In User of error and asks to enter another team name
	3.2.3. Use Case returns to step 3
Special Requirements	1. The Logged In User should be automatically added to the new team
Pre-Conditions	1. The Logged In User should not have reached the max limit of number of joined teams (100 teams)
Post-Conditions	1. The Logged In User should be able to see the correct joined team under: Profile, List of Joined Teams
	2. The Logged In User should be able to see the team calendar under the Calendars Page
	3. The Logged In User should be the only member of the new team

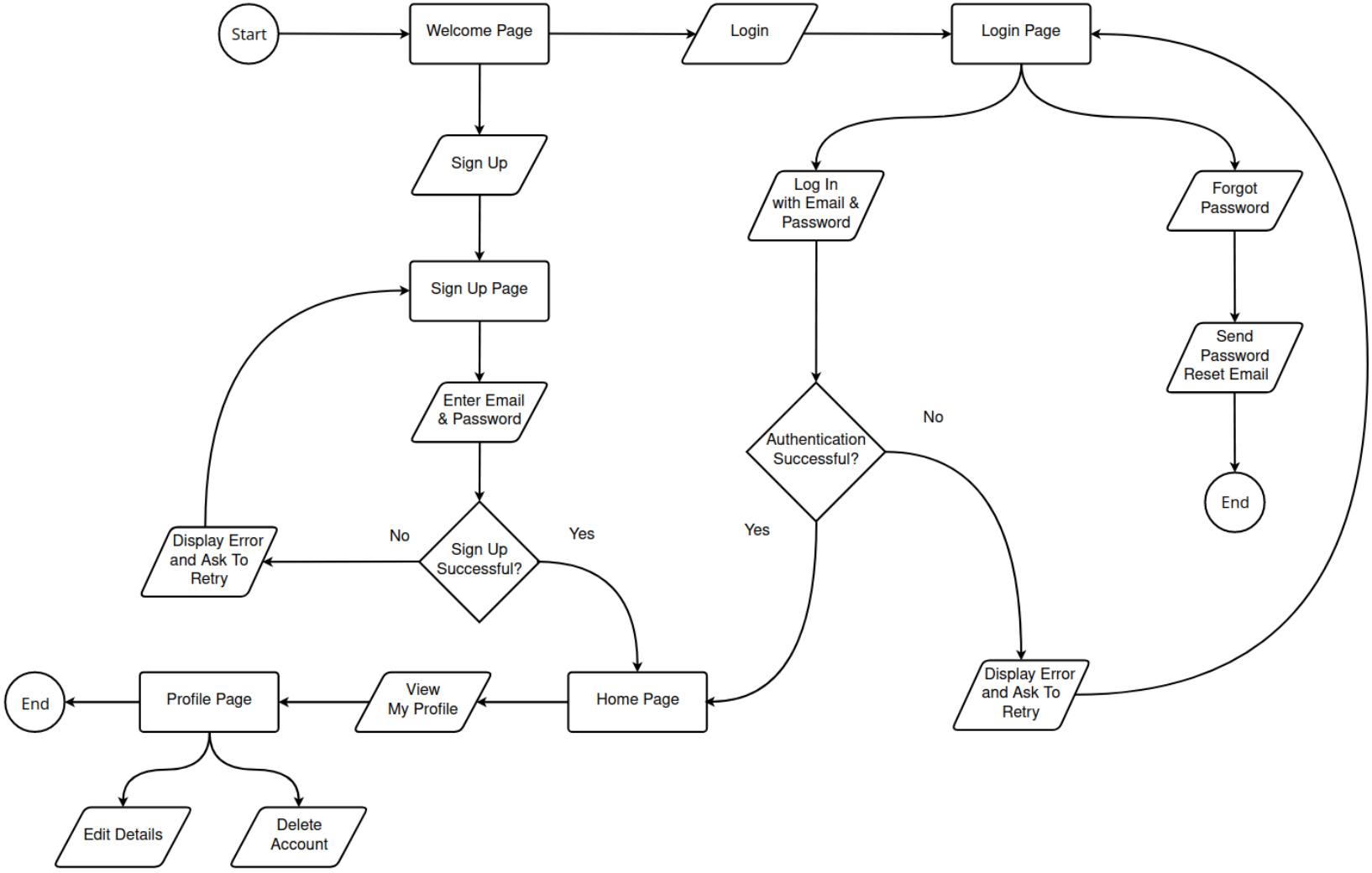
Extension Points	1. The Logged In User is able to leave a joined team

Use Case 9

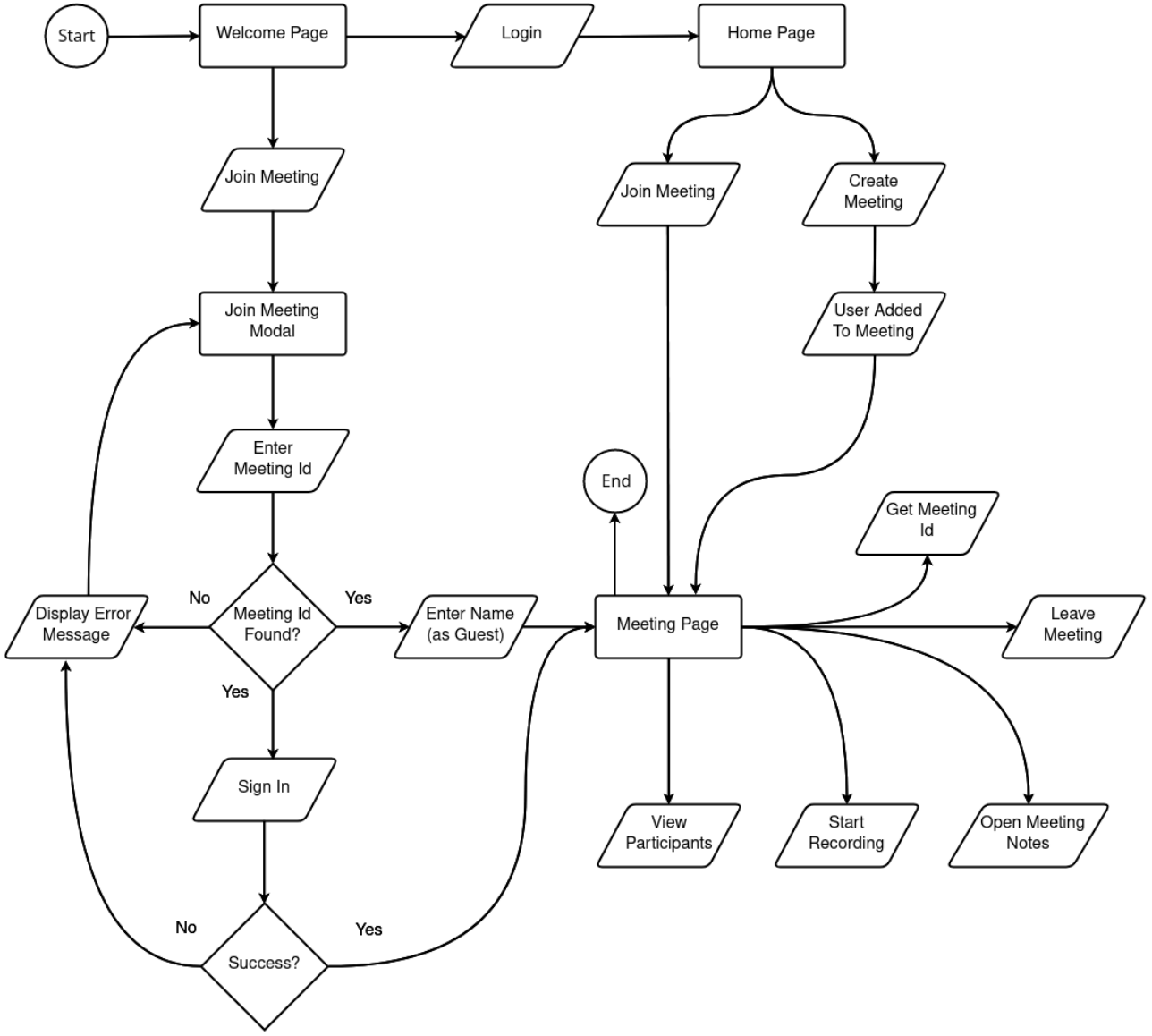
Use Case	View Profile
Description	The View Profile Use Case provides the functionality for a Logged In User to view their profile information
Actors	Logged In User
Basic Flow	1. Logged In User navigates to Profile Page
Alternative Flows	None
Exception Flows	None
Special Requirements	1. The System should display all profile information including: name, email and password
Pre-Conditions	None
Post-Conditions	1. The profile displayed should be the correct account (i.e. the same profile the Logged In User signed in with)
Extension Points	1. The Logged In User can edit their name
	2. The Logged In User can update their password
	3. The Logged In User can delete their account

Use Case 10

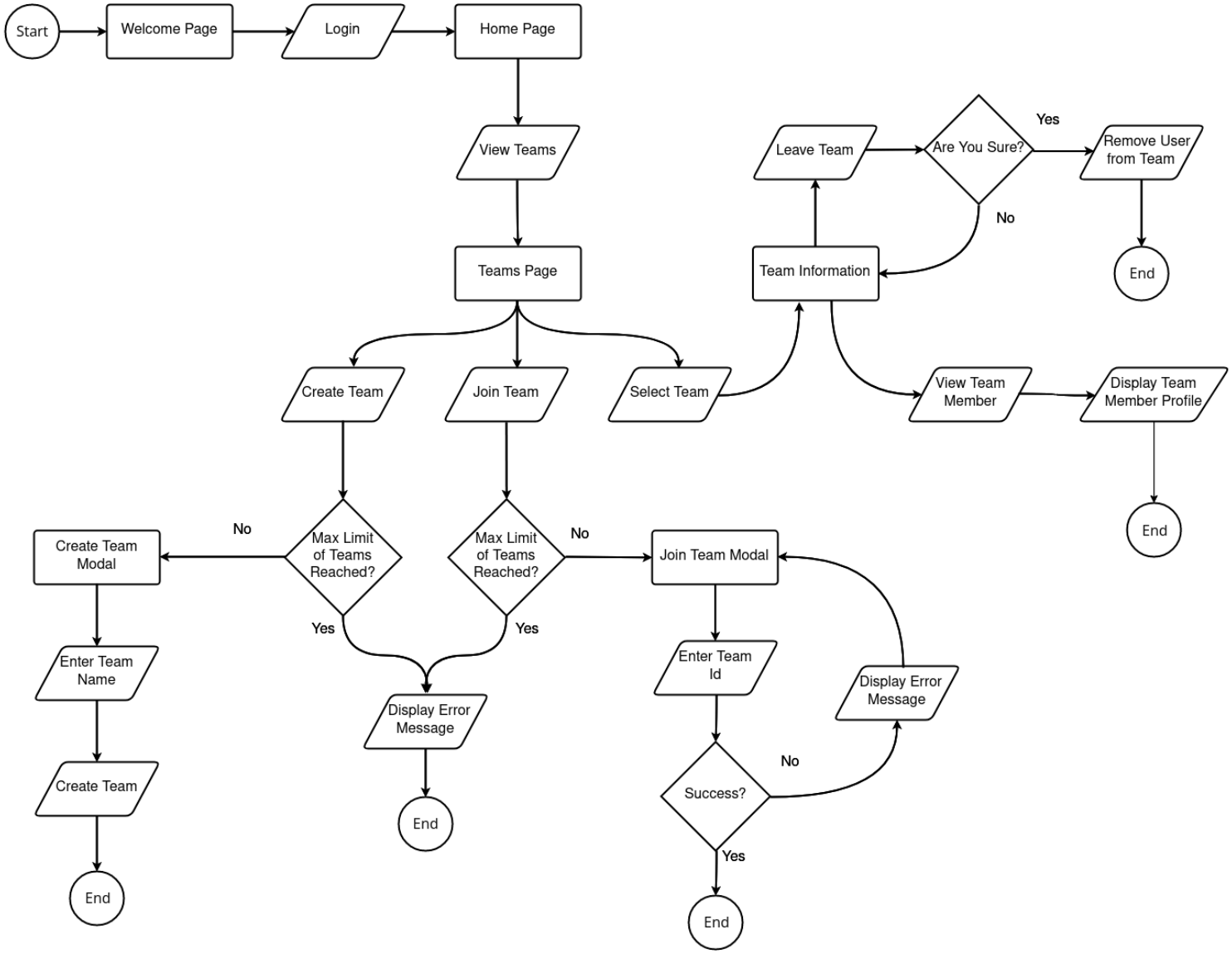
Appendix C: **User Flow Diagrams**



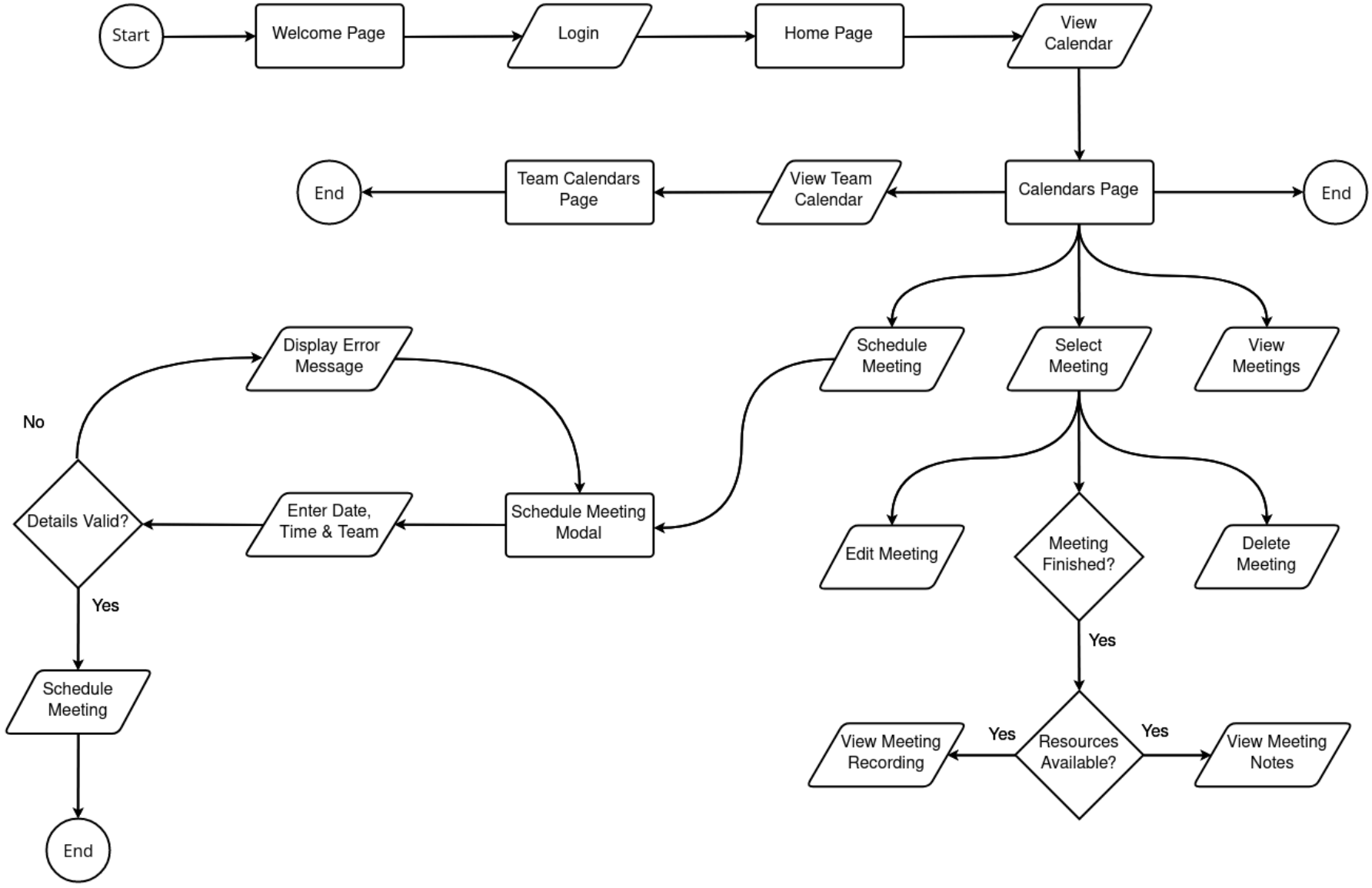
User Flow Diagram Depicting Login and Sign-Up Use Cases



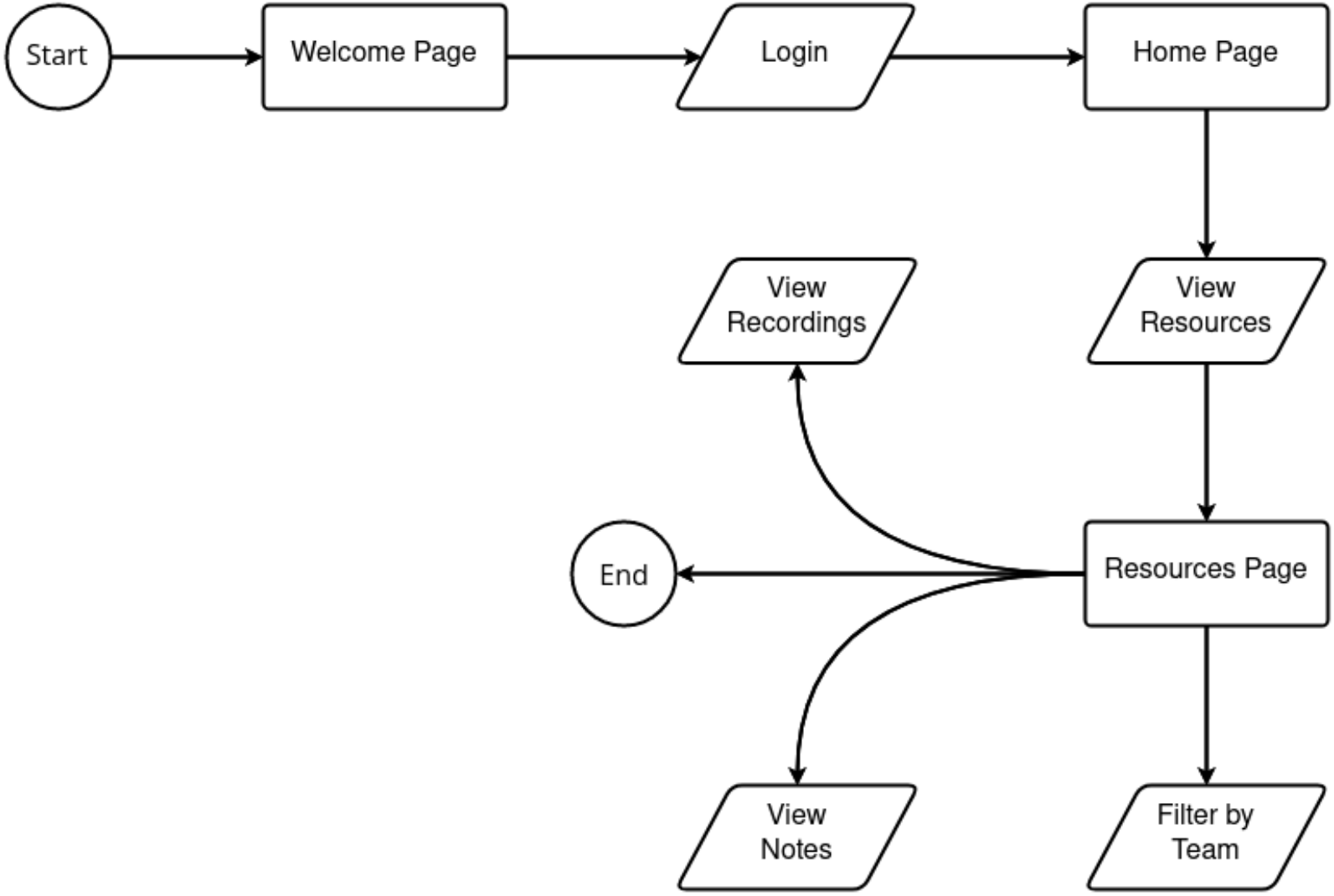
User Flow Diagram Depicting Join And Create Meeting Use Cases



User Flow Diagram Depicting Join and Create Team Use Cases

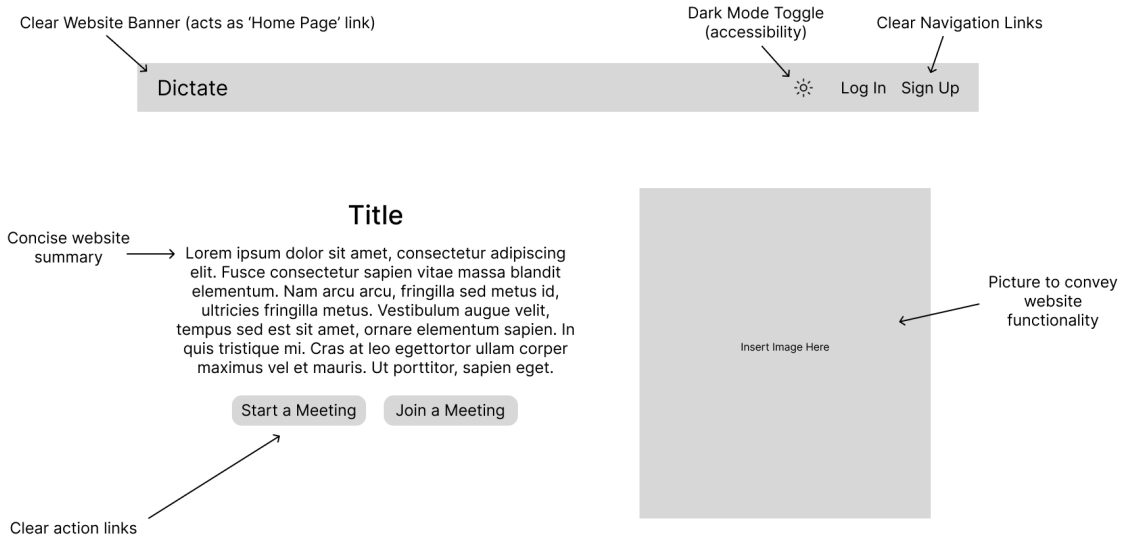


User Flow Diagram Depicting Meeting Use Cases

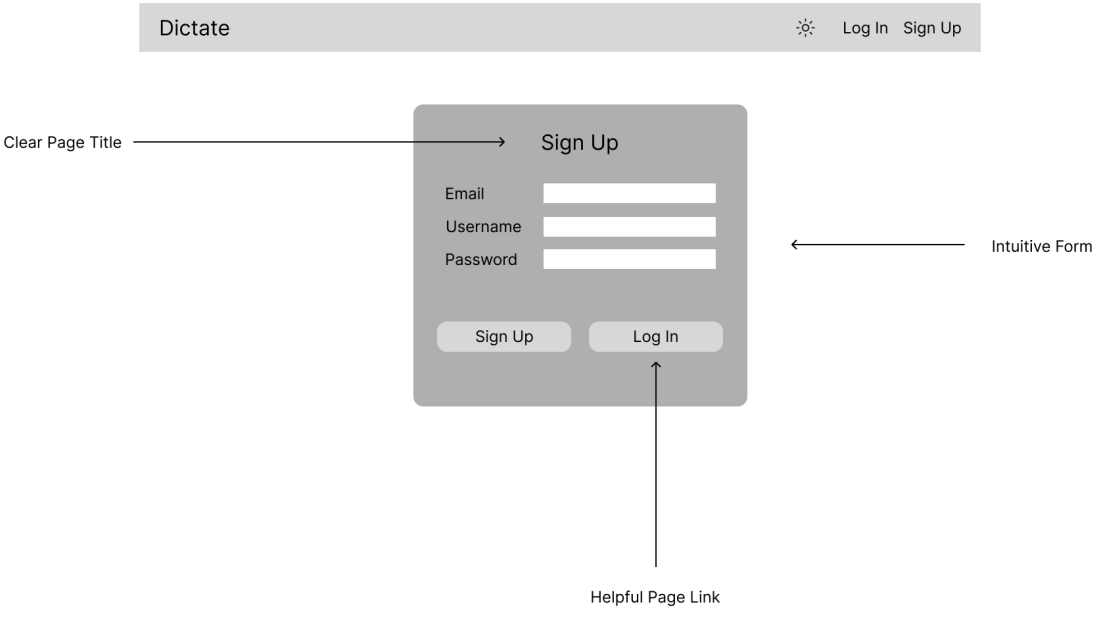


User Flow Diagram Depicting View Recordings And Notes Use Cases

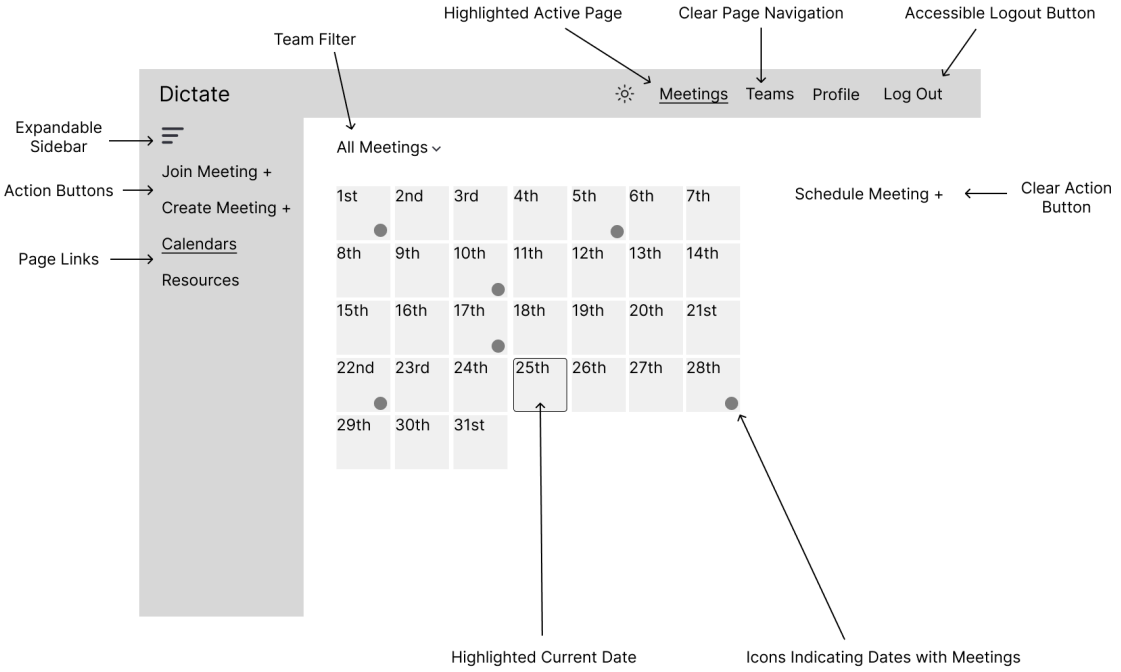
Appendix D: **Wireframes**



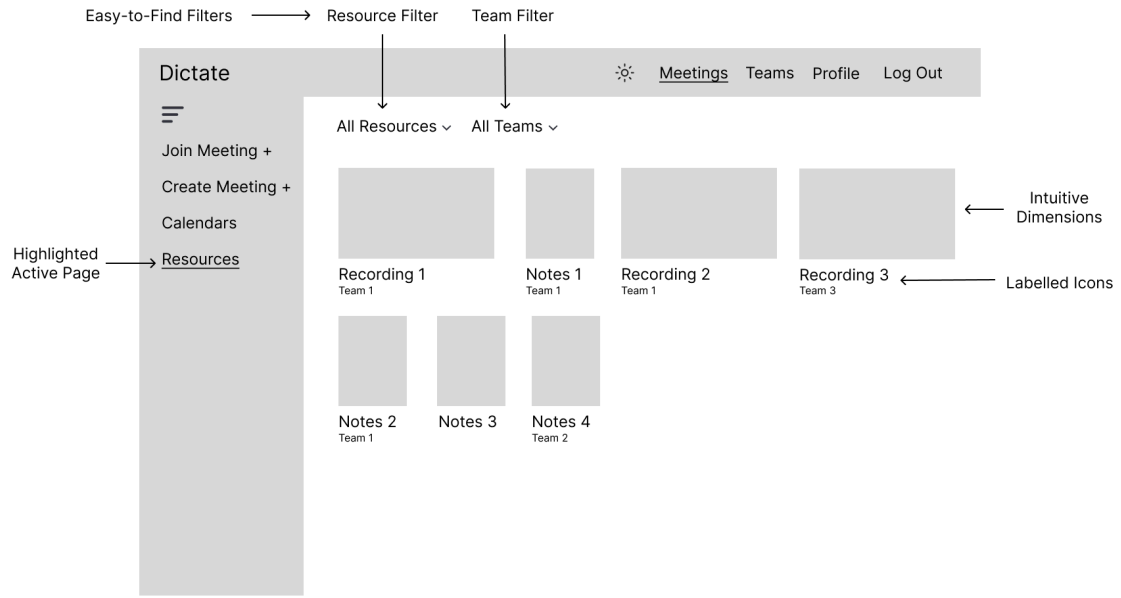
Wireframe Diagram: Home Page



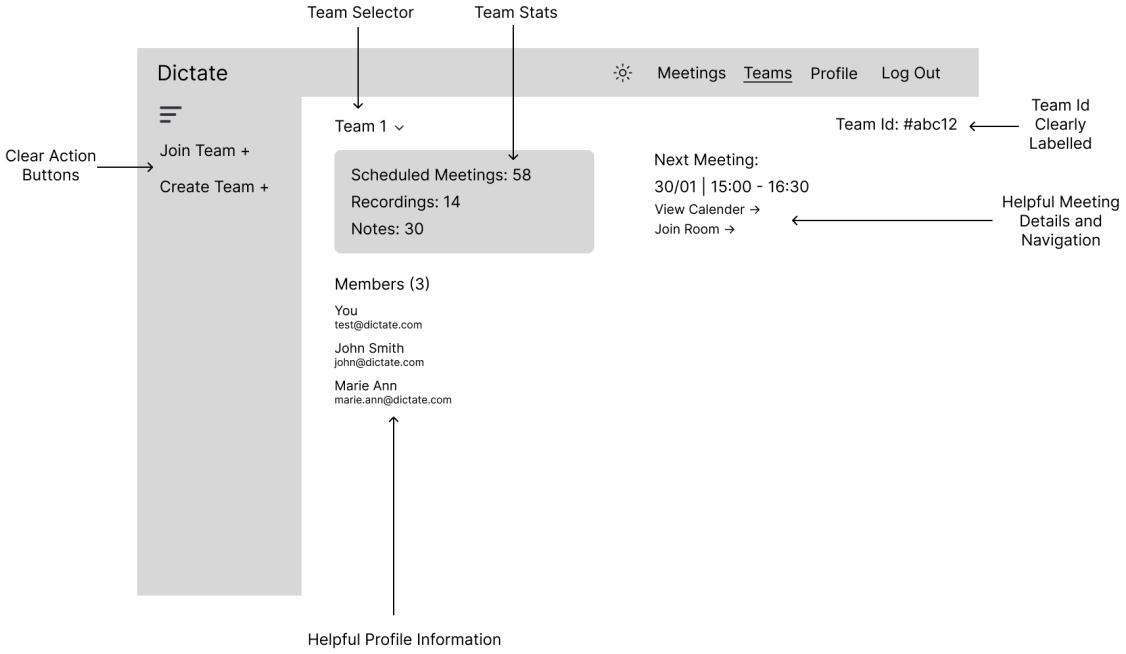
Wireframe Diagram: Signup and Login Page



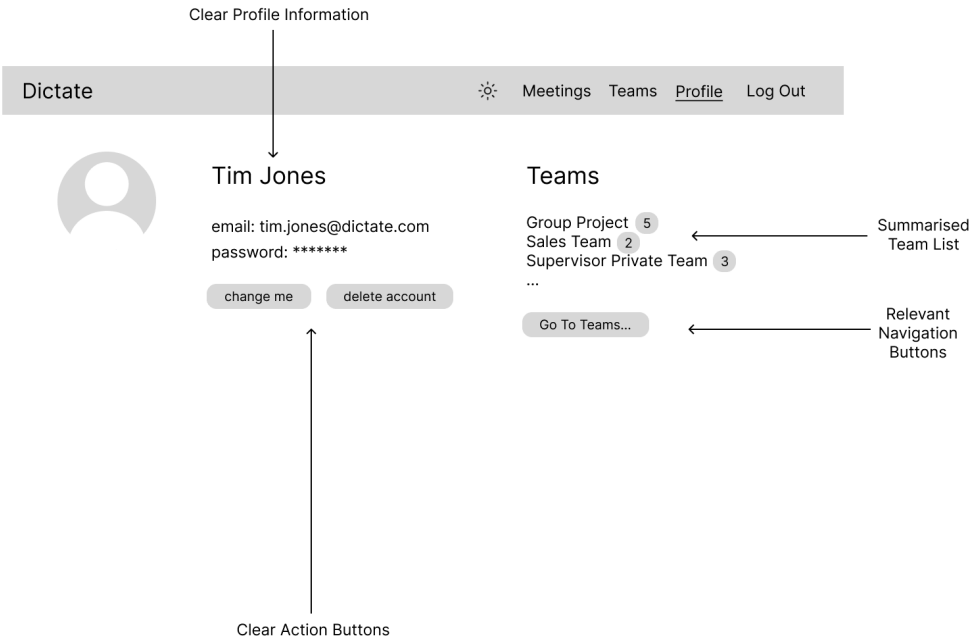
Wireframe Diagram: Calendars Page



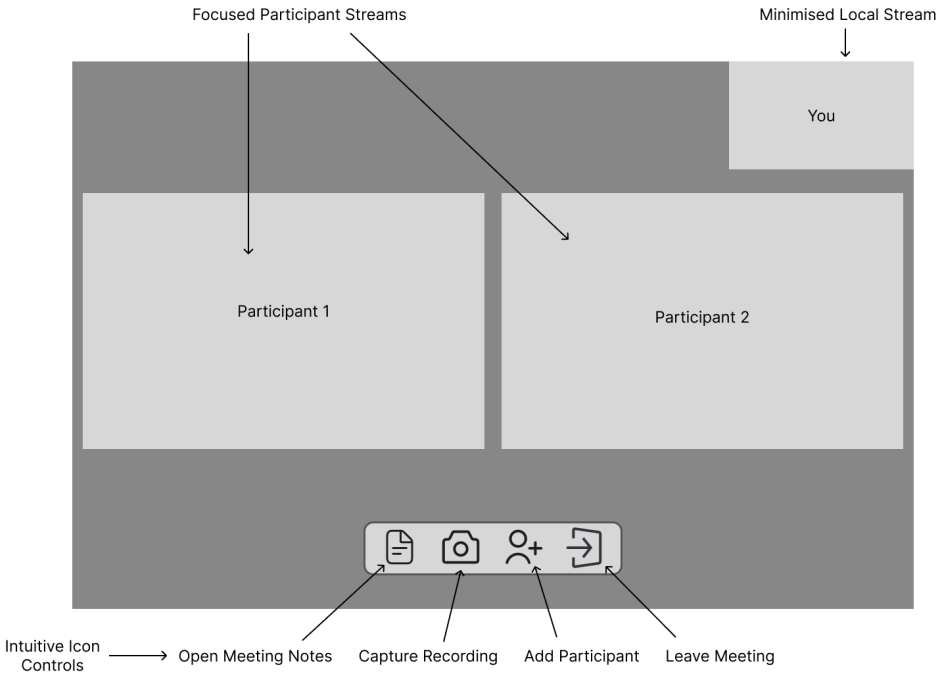
Wireframe Diagram: Resources Page



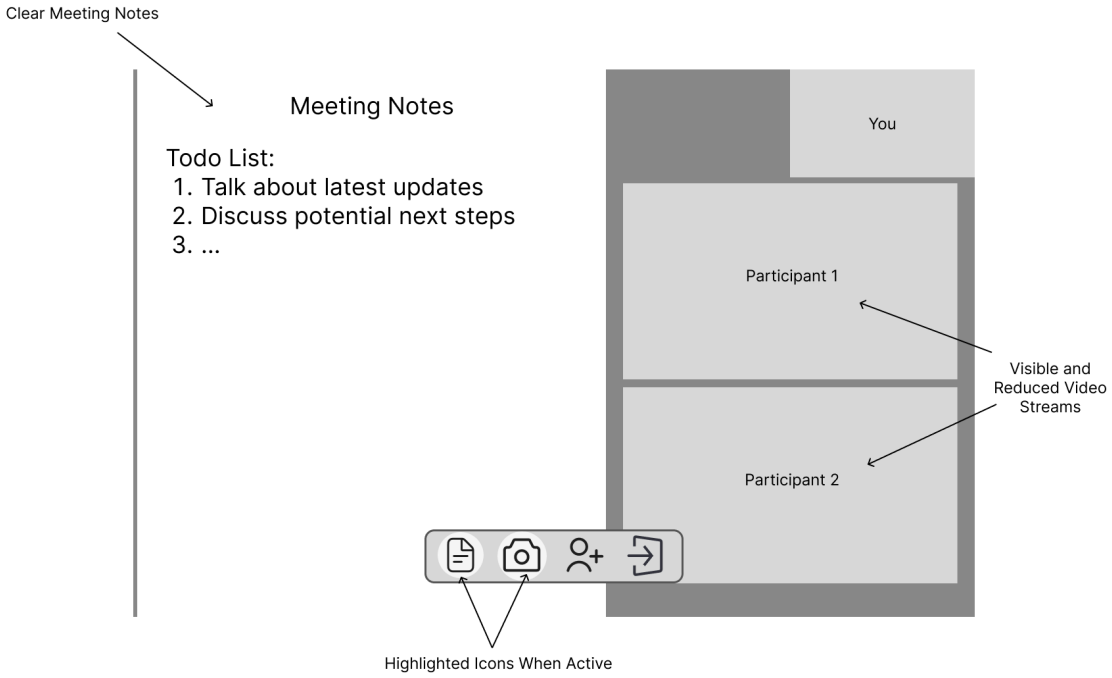
Wireframe Diagram: Teams Page



Wireframe Diagram: Profile Page

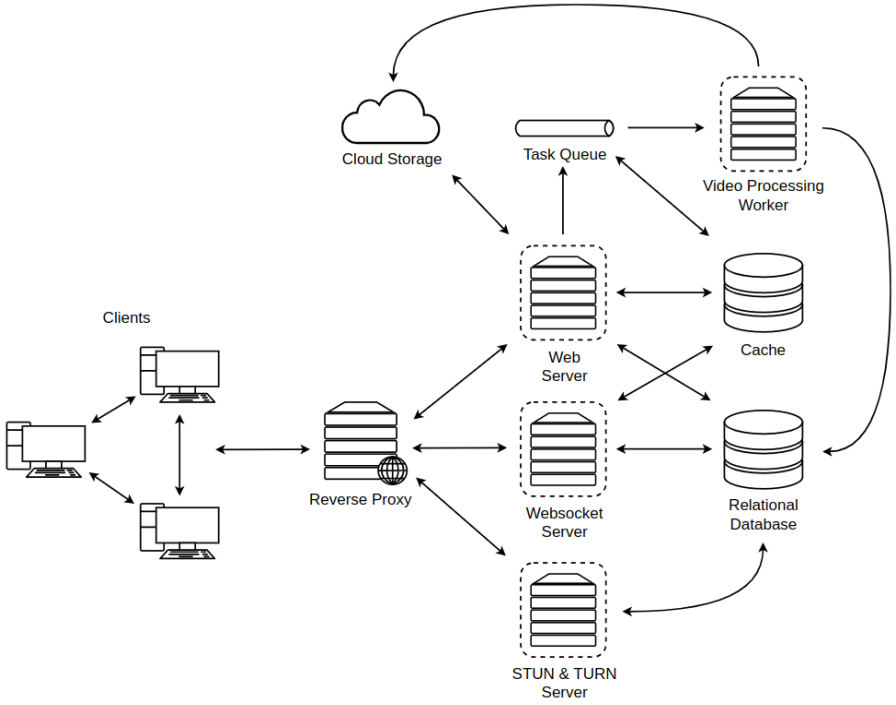


Wireframe Diagram: Video-conference Page (1)

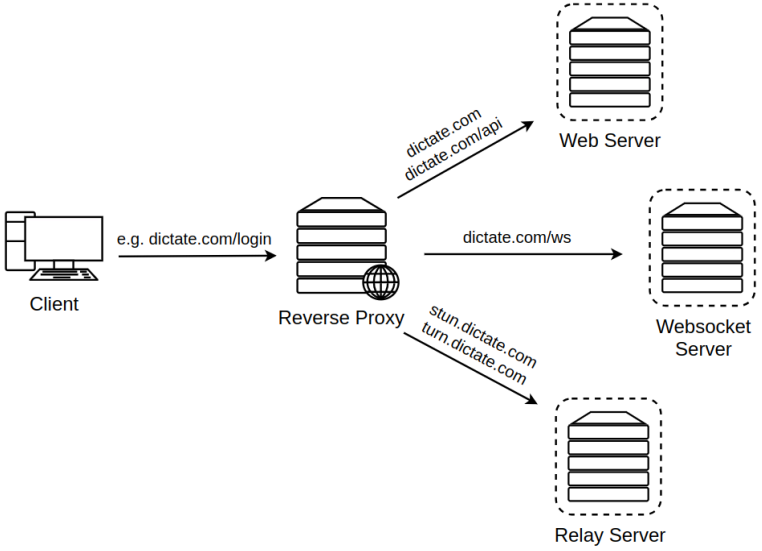


Wireframe Diagram: Video-conference Page (2)

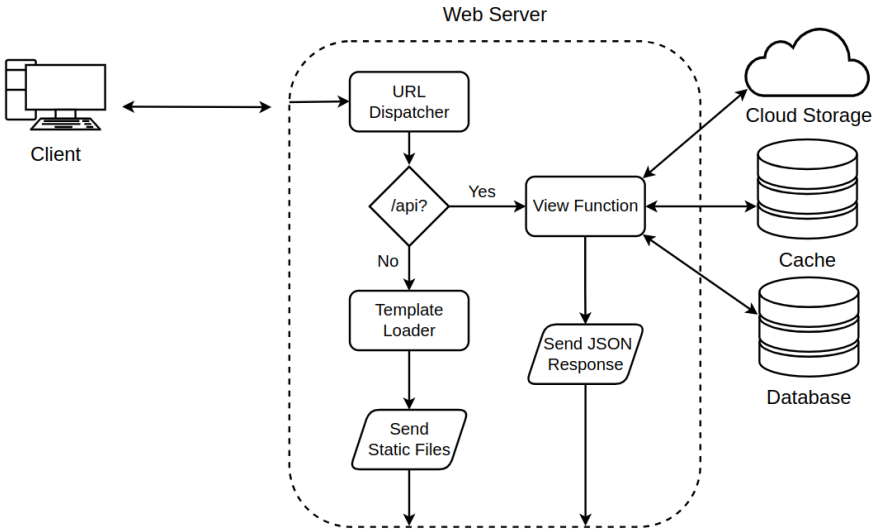
Appendix E: **System Architecture Diagrams**



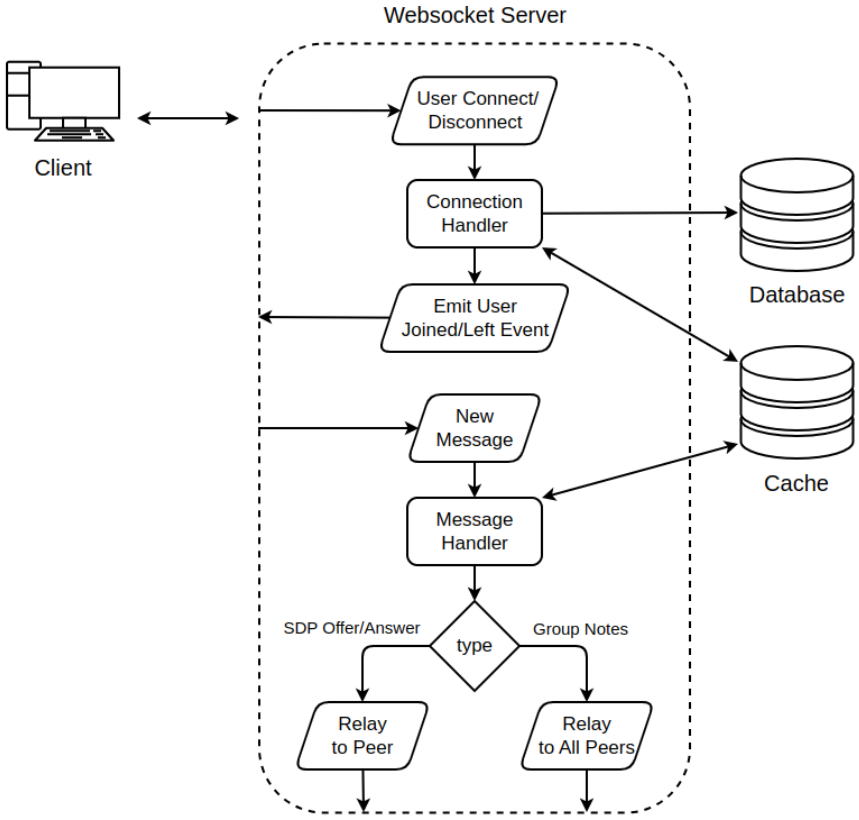
Overall System Architecture



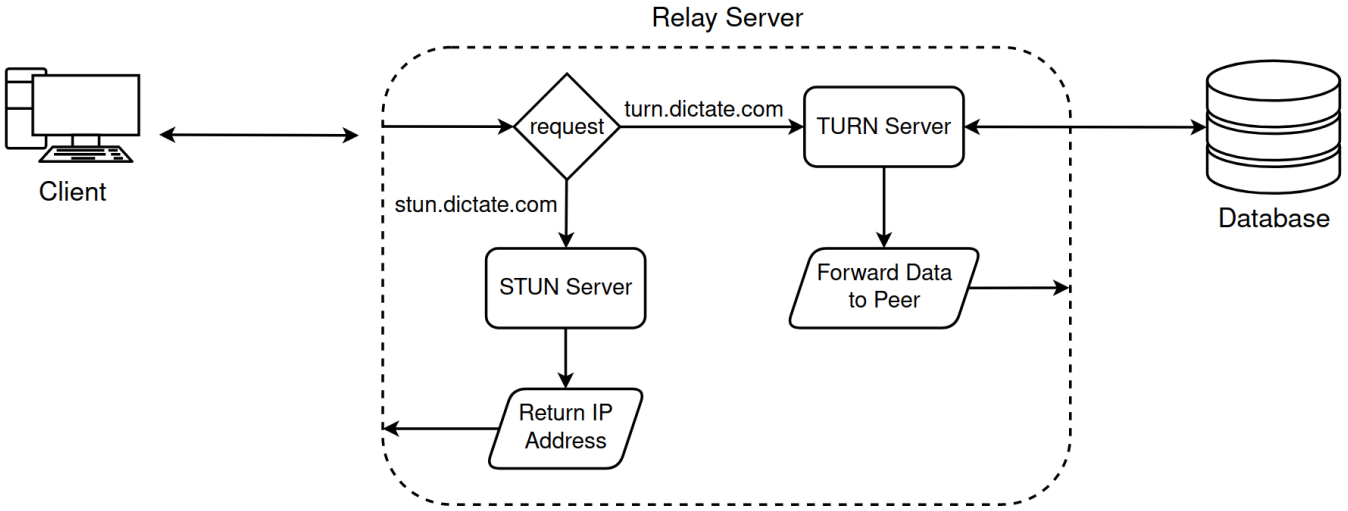
Reverse Proxy Architecture



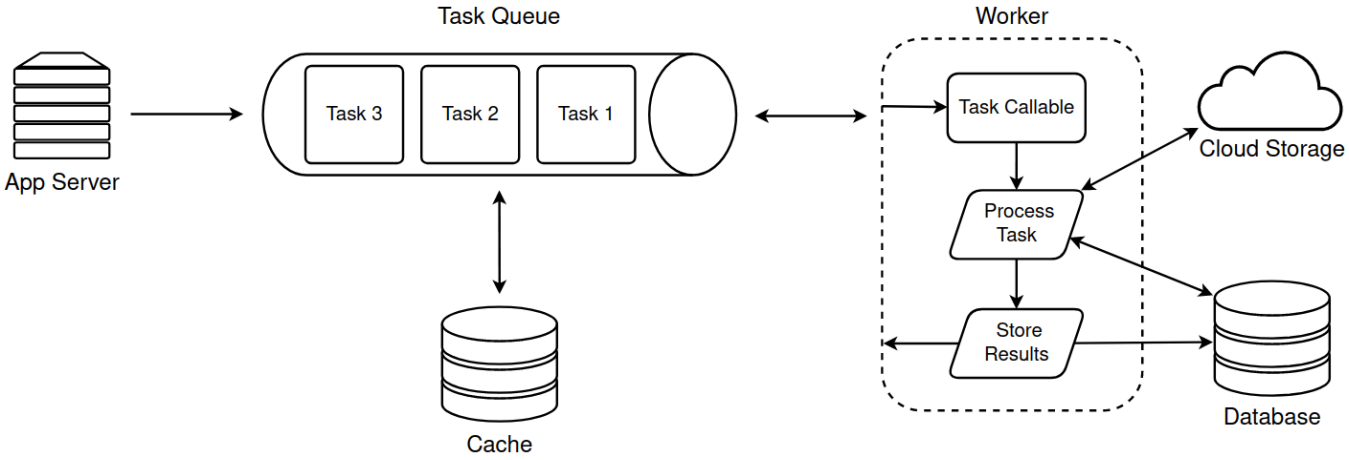
App Server Architecture



Websocket Server Architecture

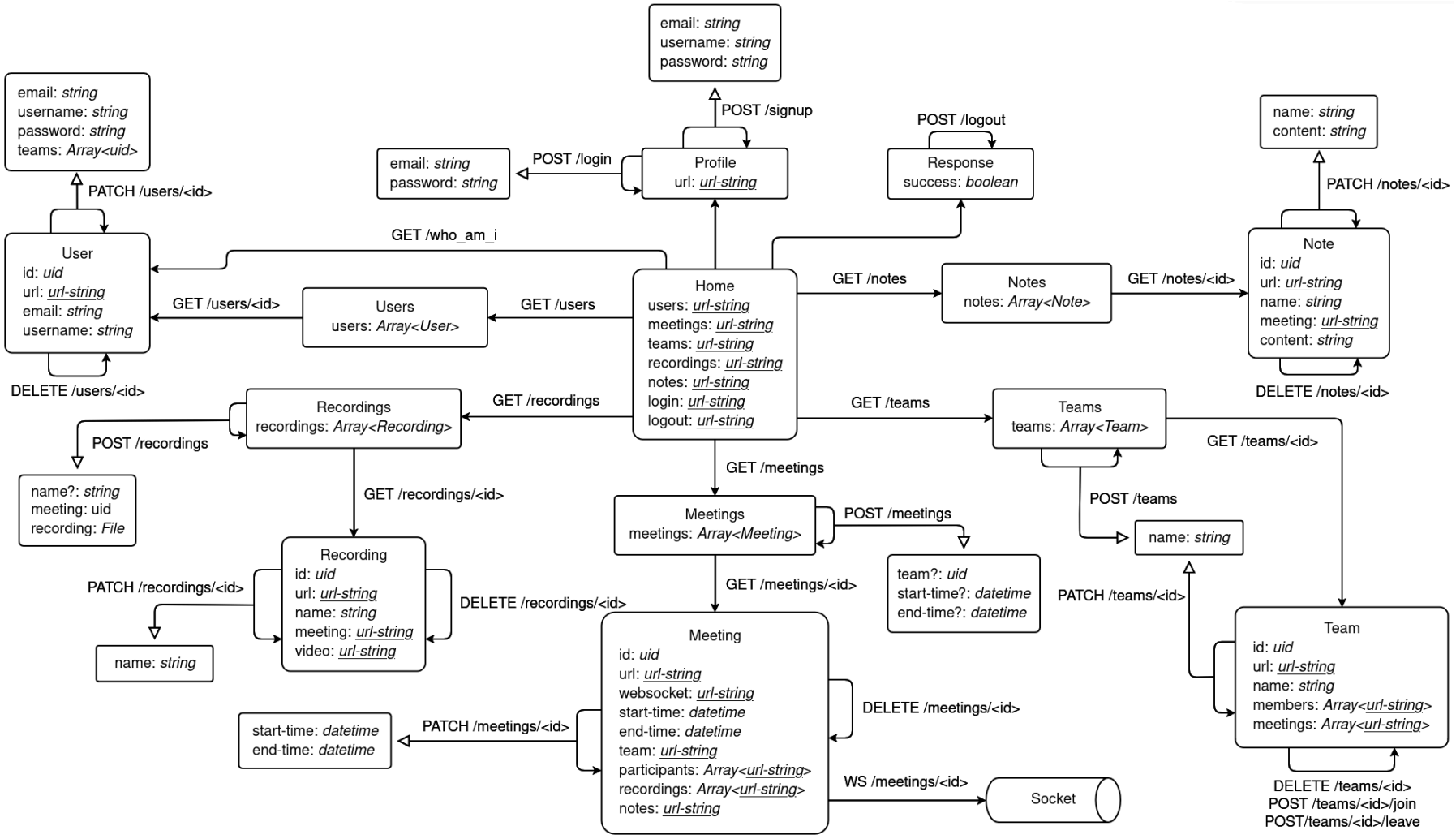


Relay Server Architecture



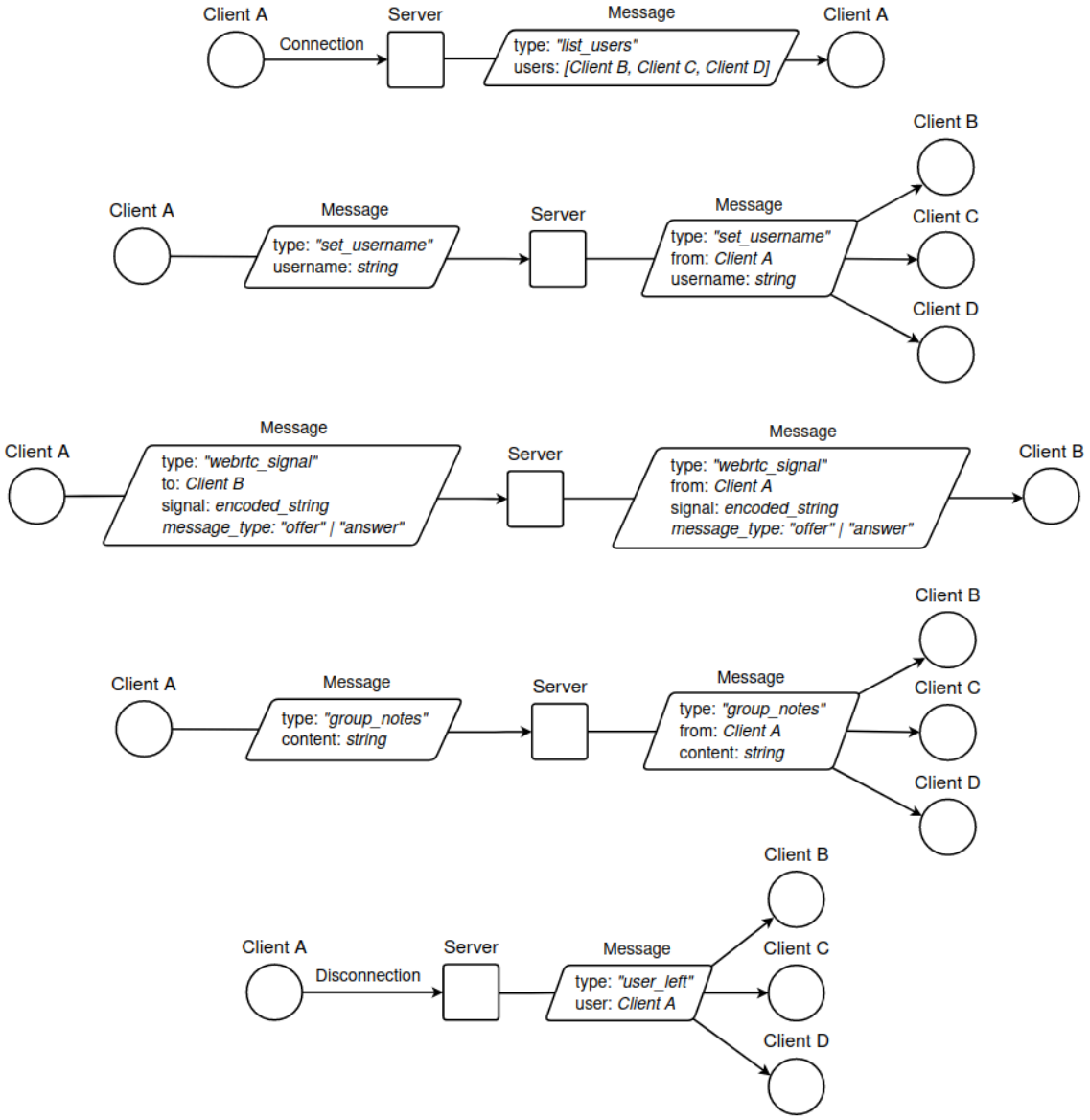
Task Queue Architecture

Appendix F: **API Diagram**



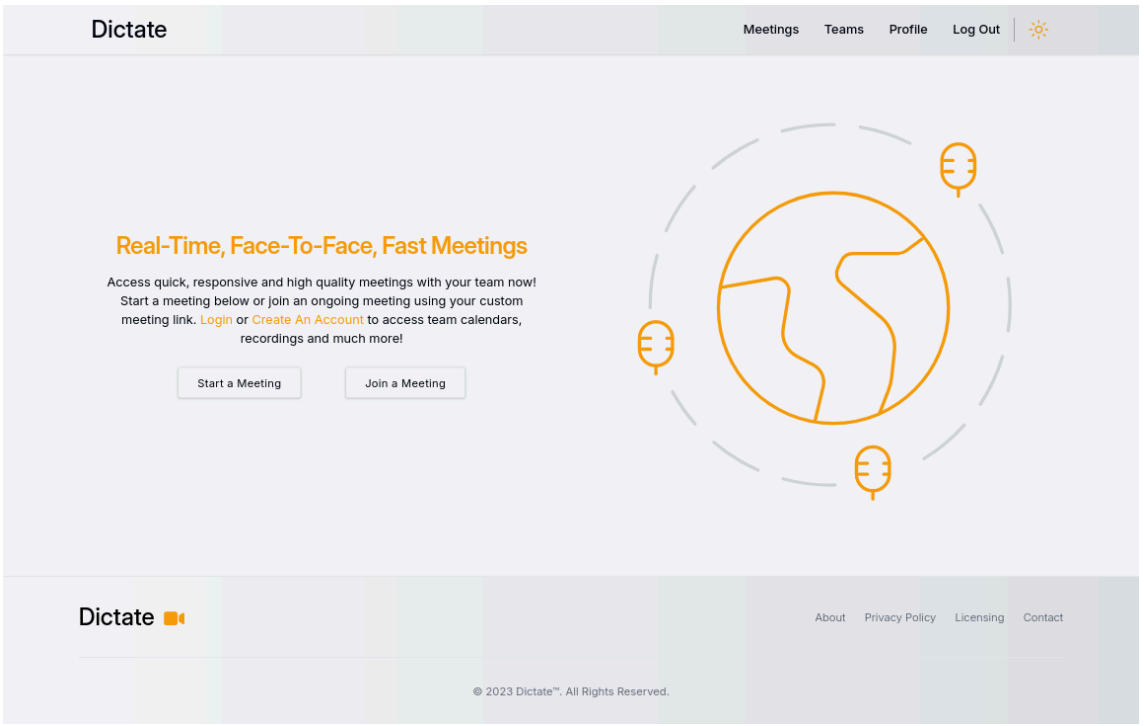
API Architecture Diagram

Appendix G: **Websocket Events Diagram**

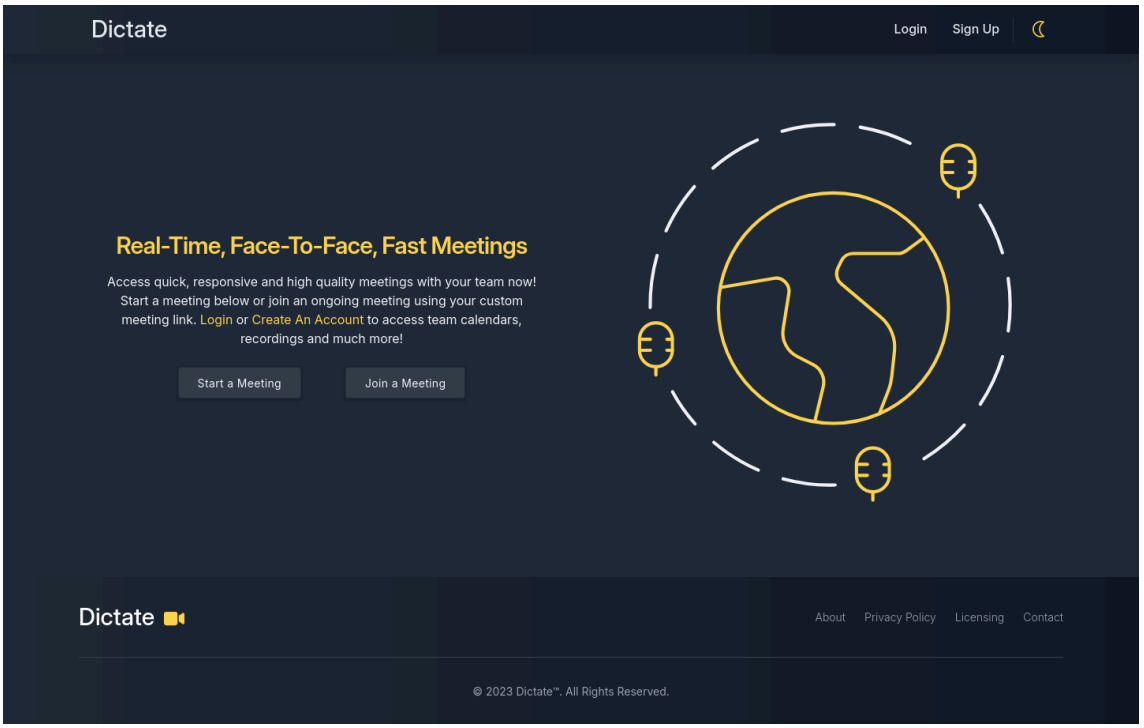


Websocket Events Architecture

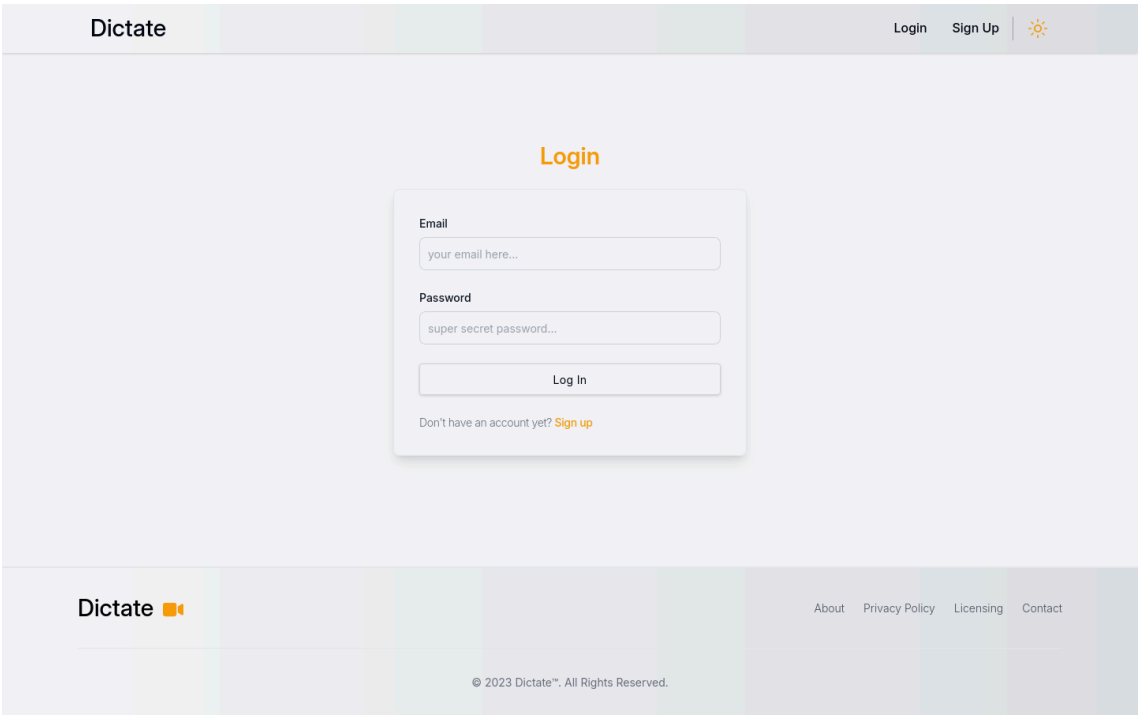
Appendix H: **Page Designs**



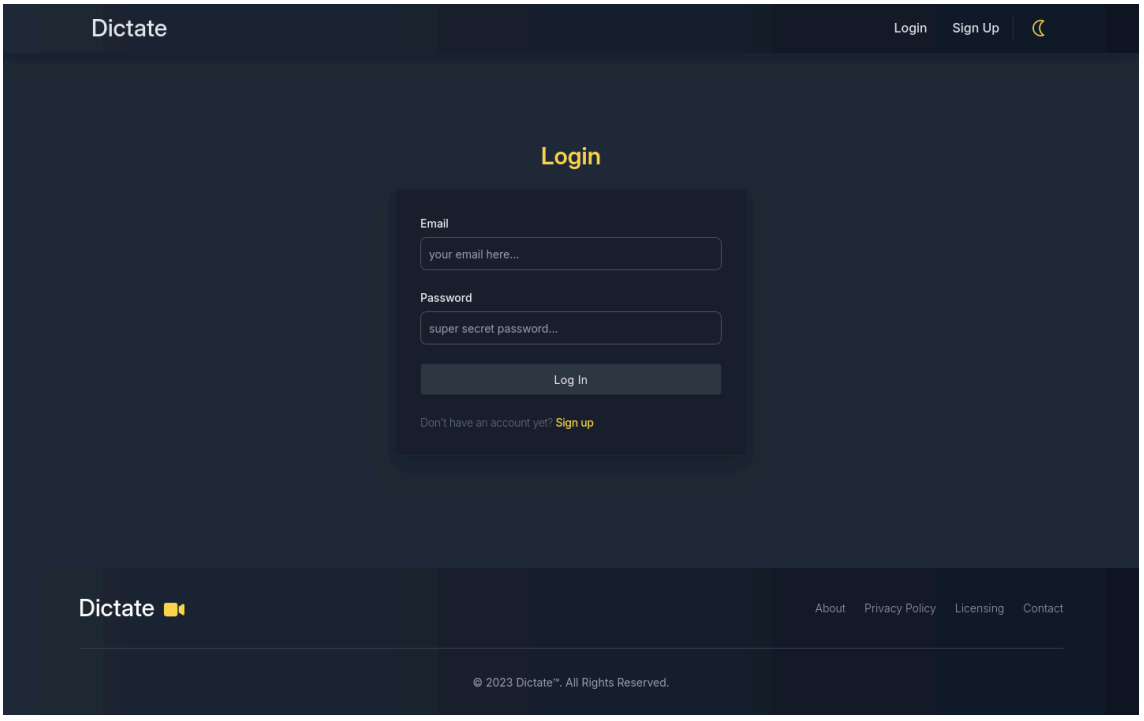
Home Page (Light Mode)



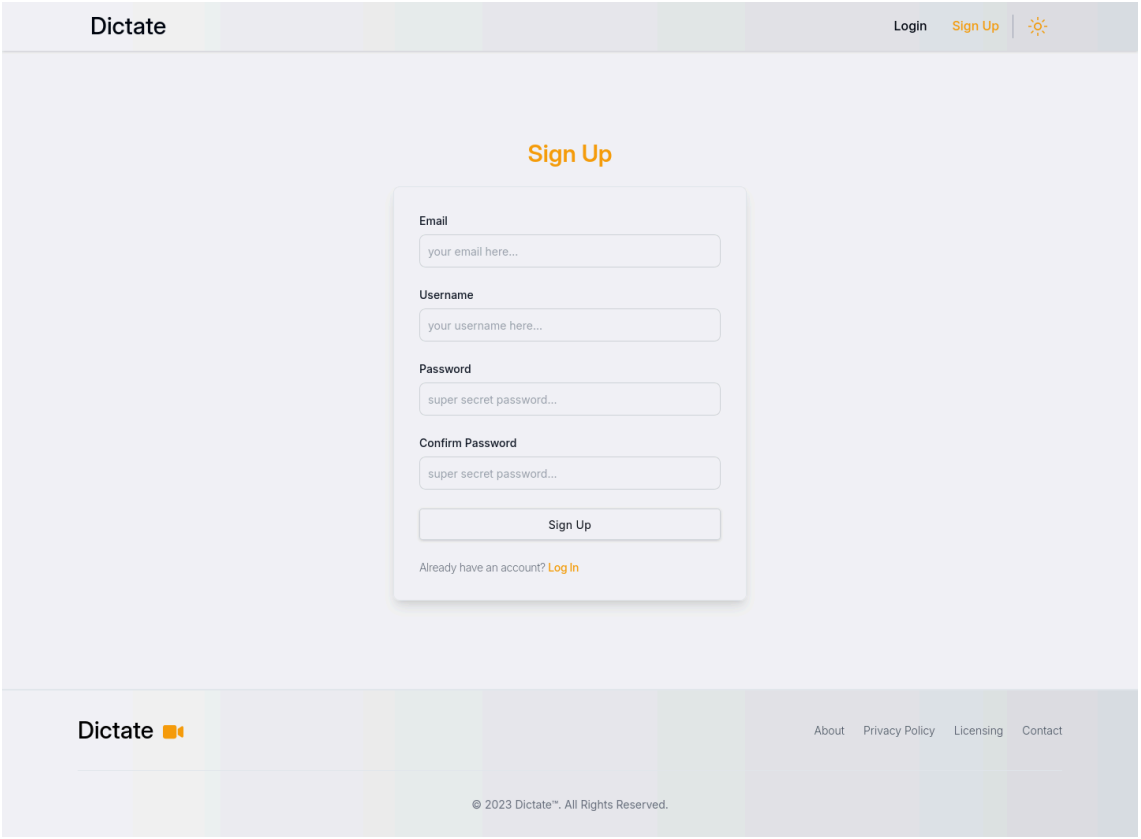
Home Page (Dark Mode)



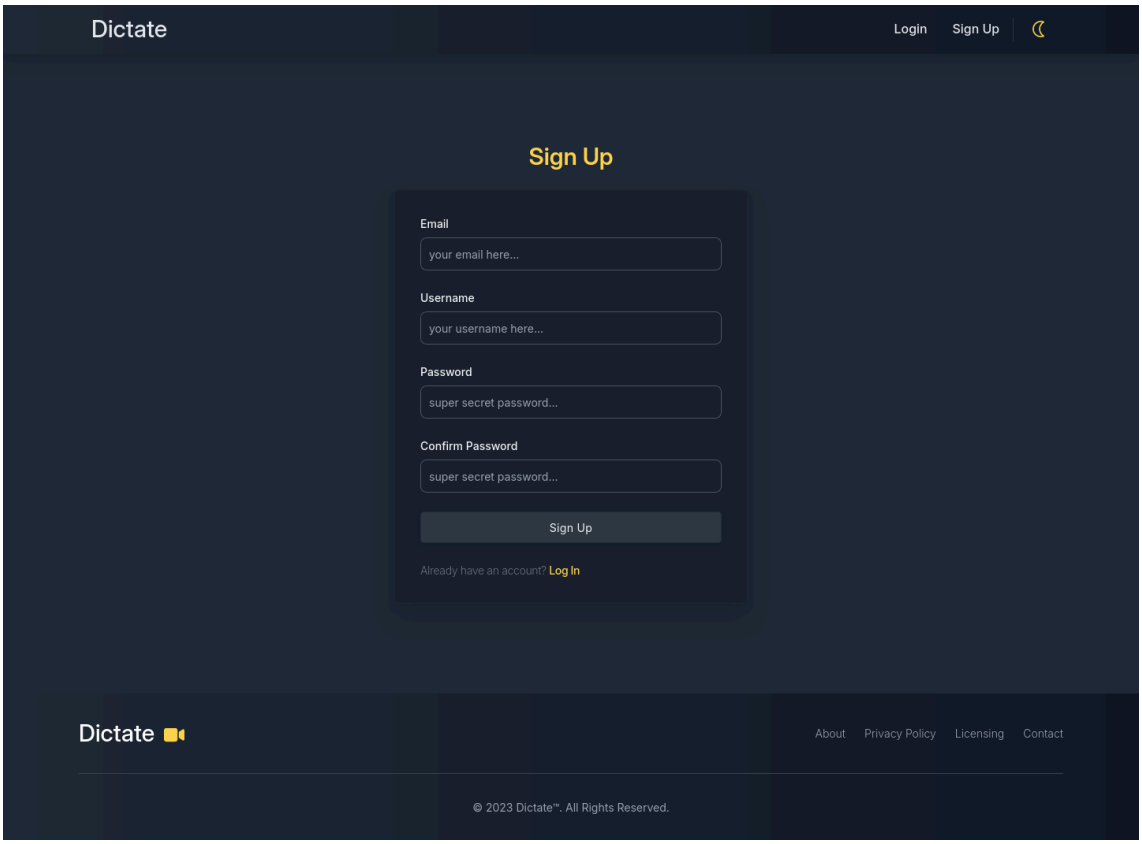
LoginPage (Light Mode)



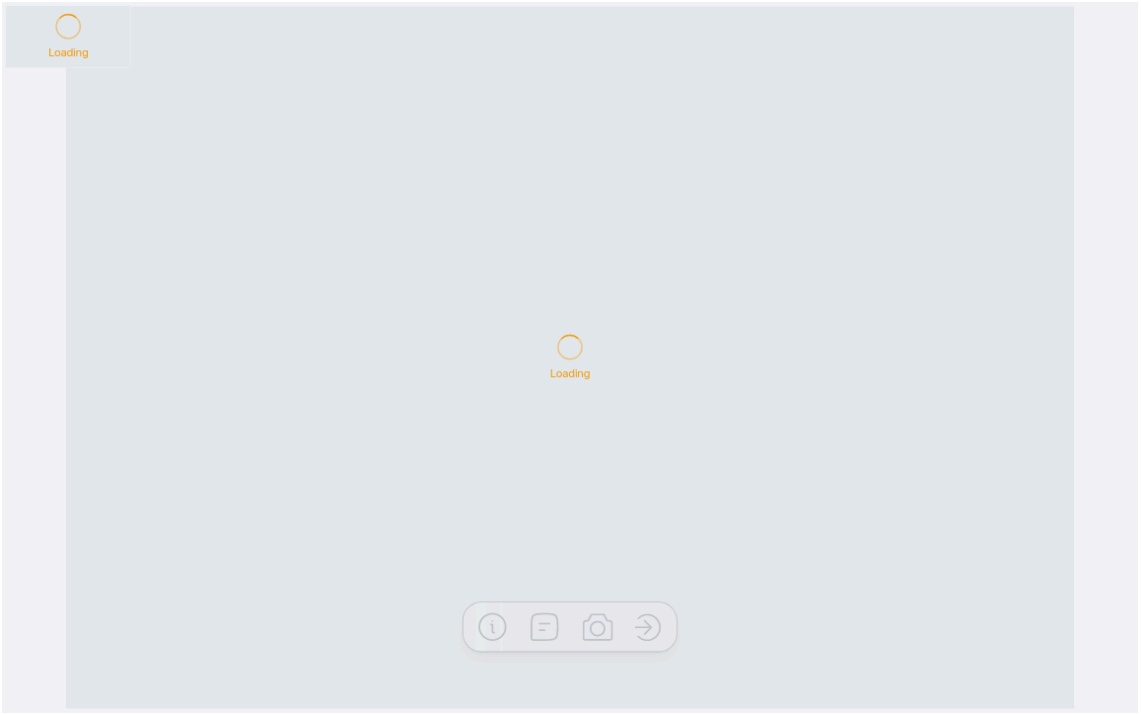
Login Page (Dark Mode)



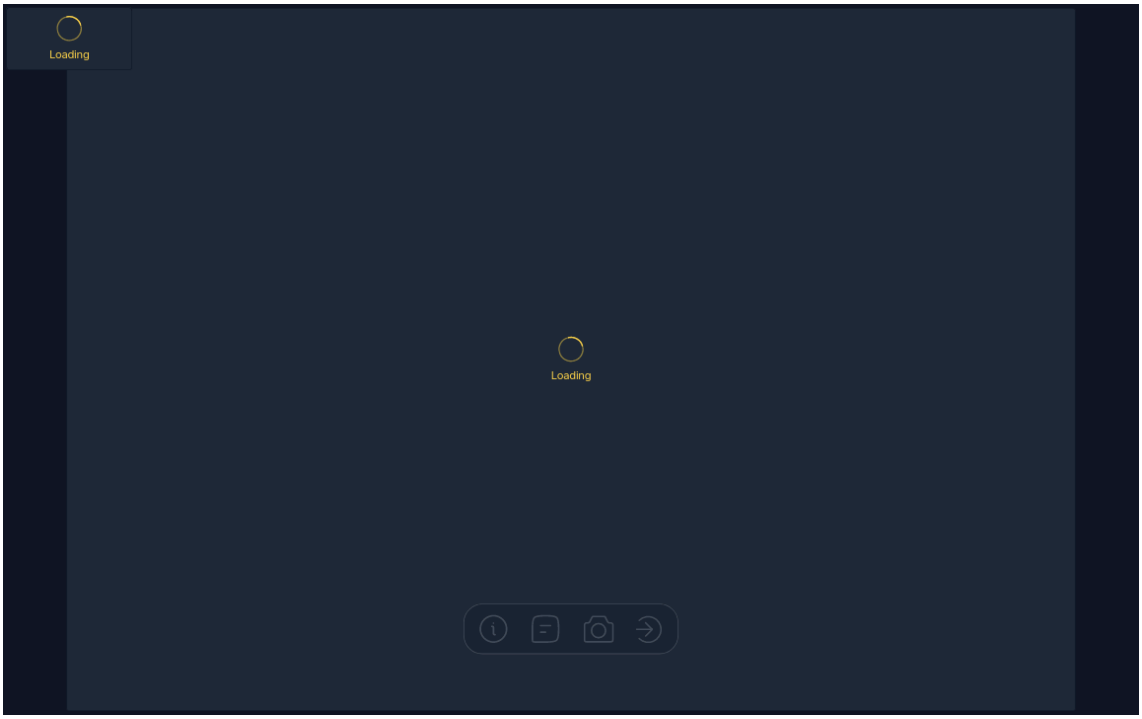
Sign-Up Page (Light Mode)



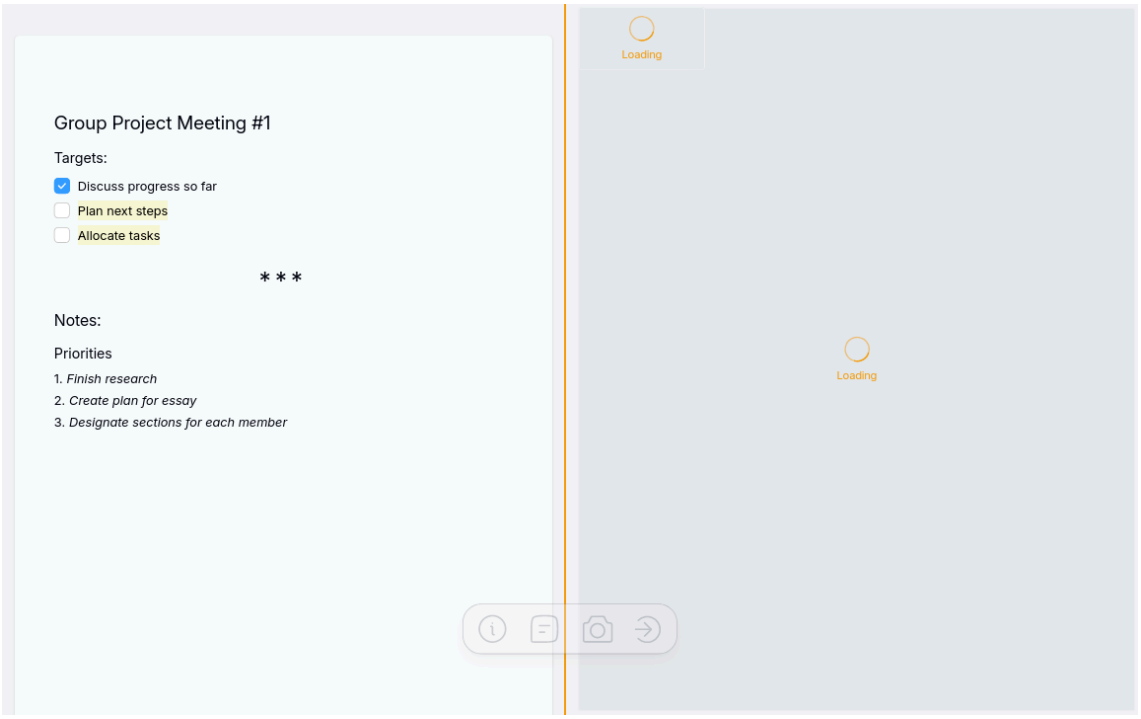
Sign-Up Page (Dark Mode)



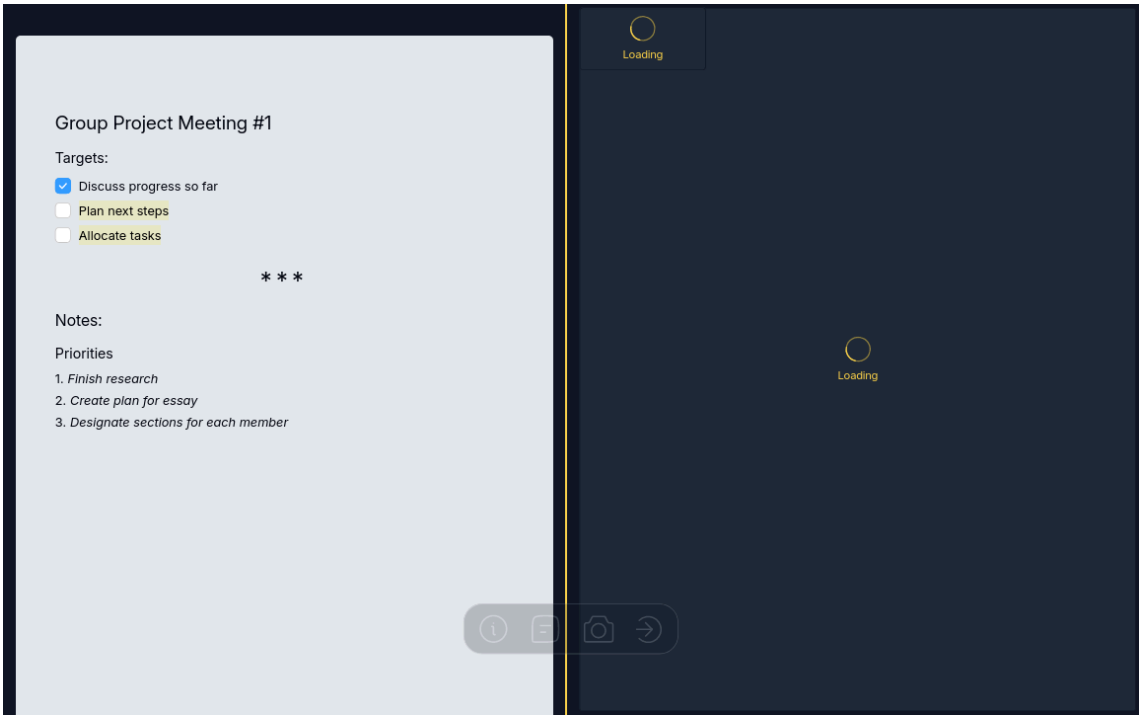
Video-Conference Page (Light Mode)



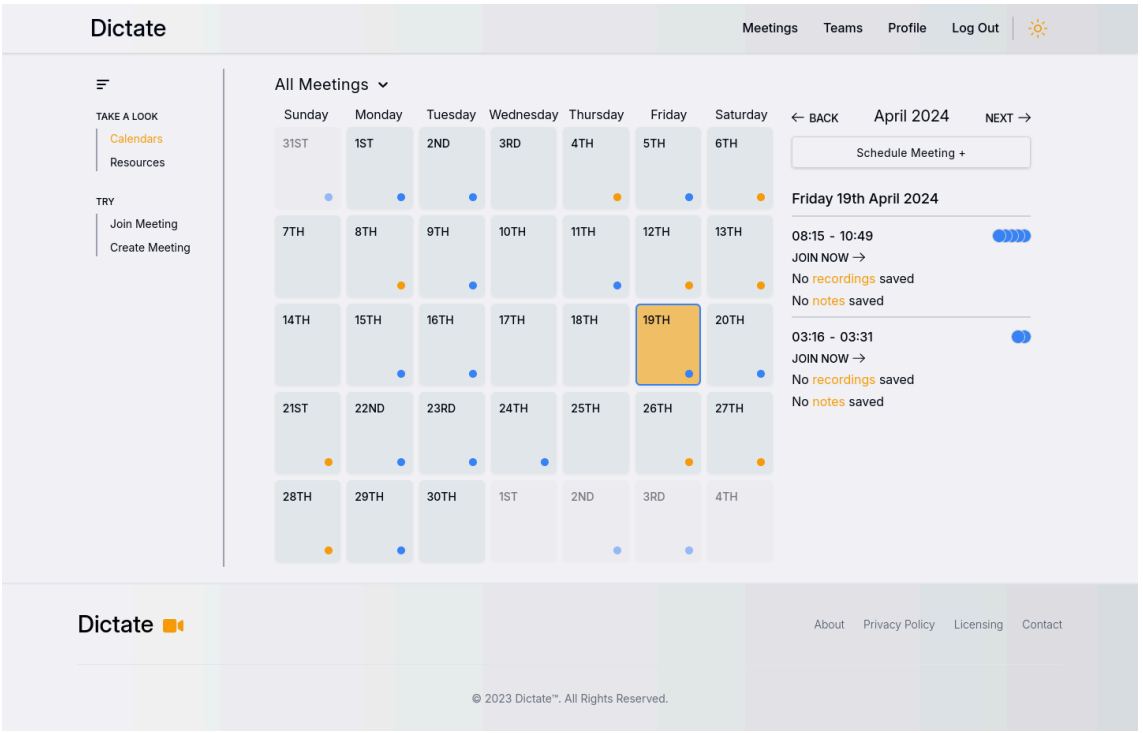
Video-Conference Page (Dark Mode)



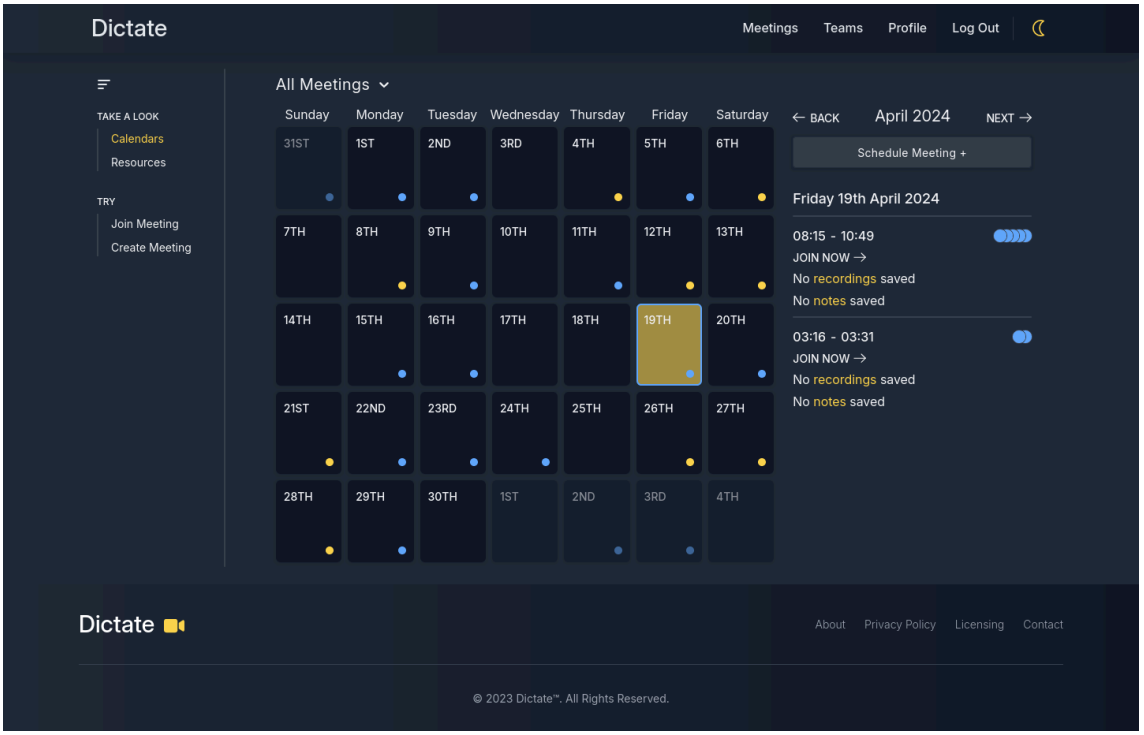
Video-Conference Page - Group Notes (Light Mode)



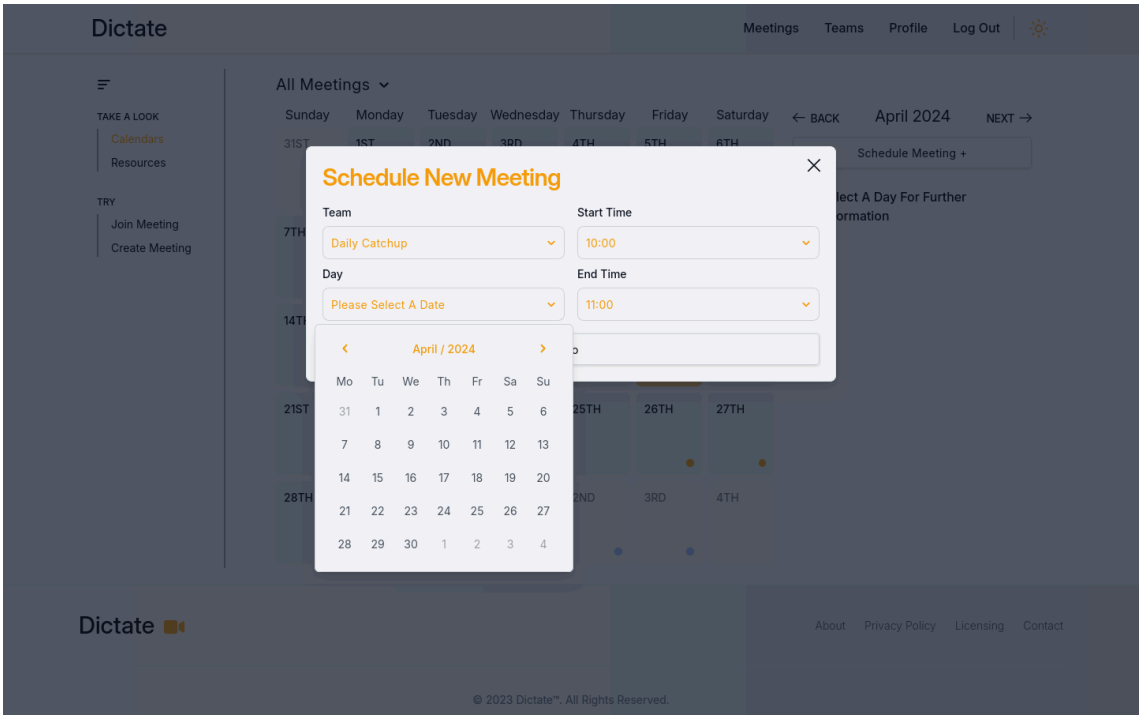
Video-Conference Page - Group Notes (Dark Mode)



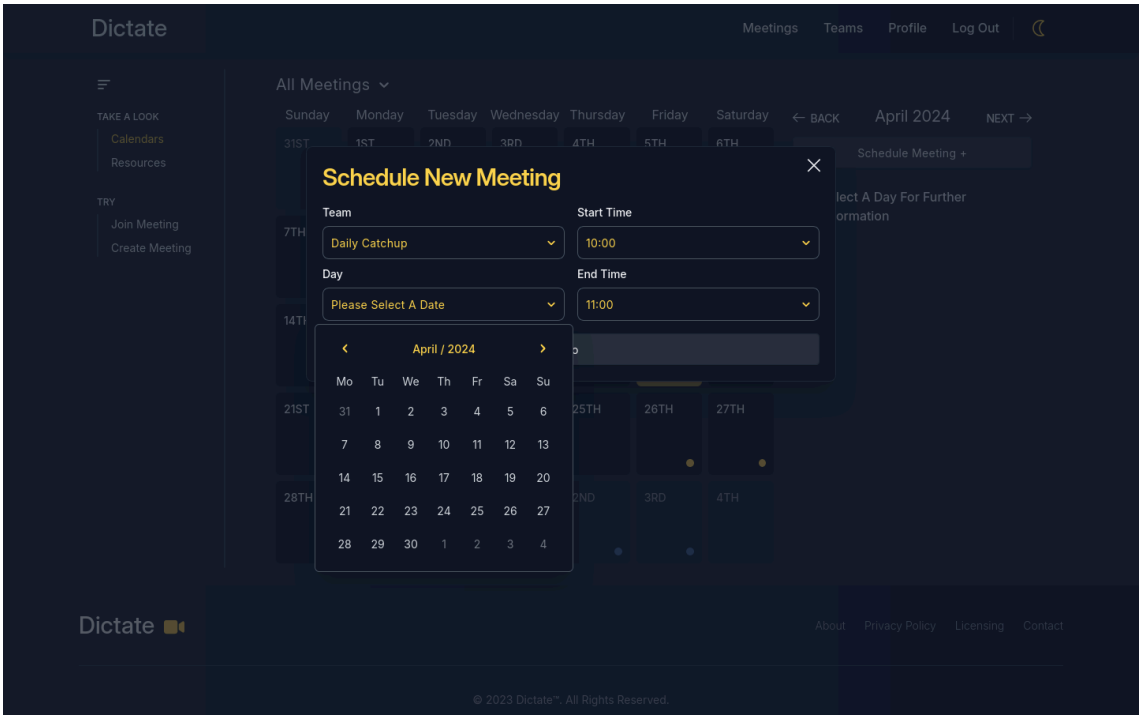
Calendars Page (Light Mode)



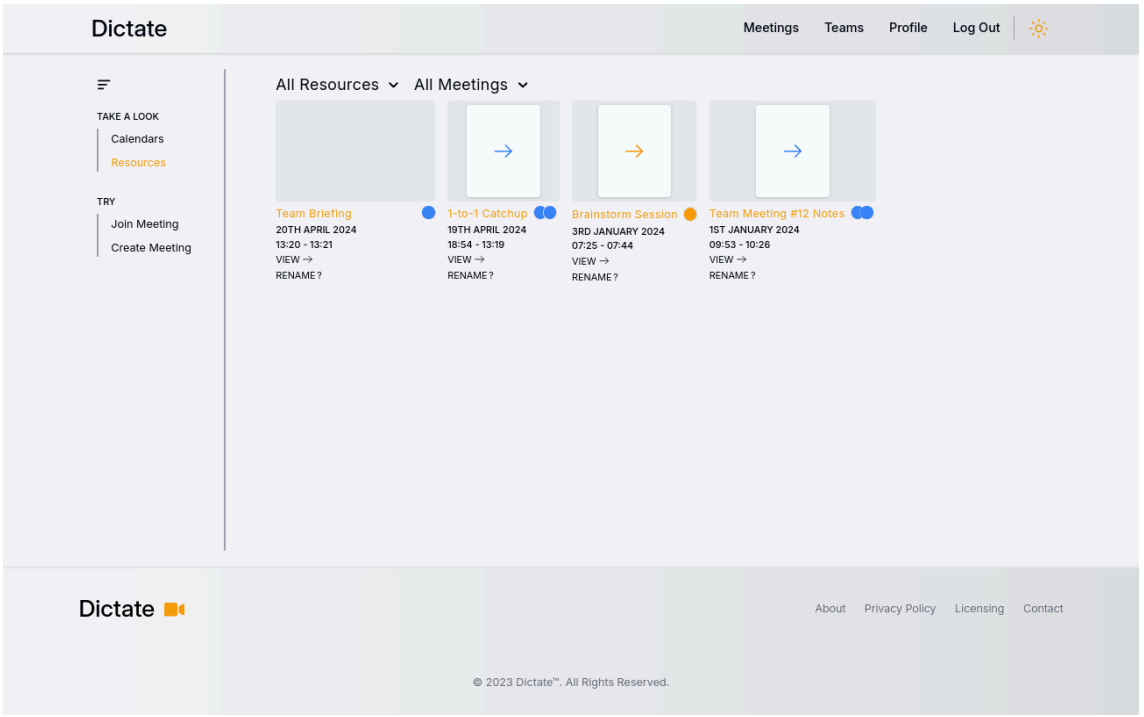
Calendars Page (Dark Mode)



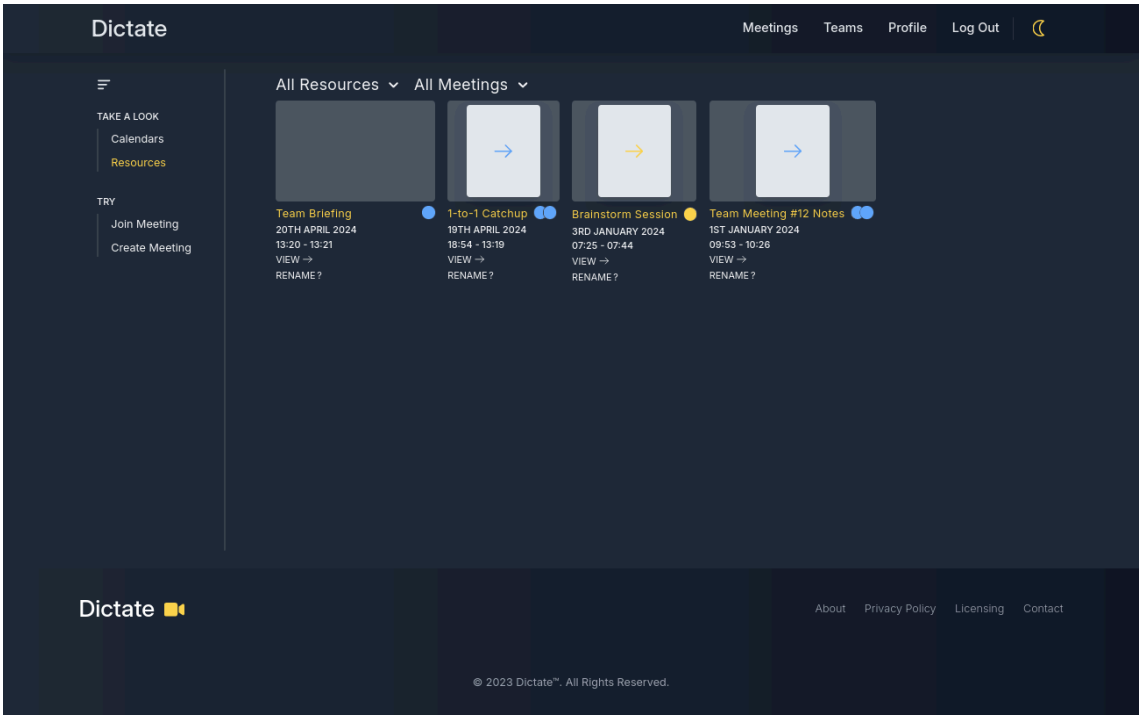
Calendars Page - Schedule Meeting Modal (Light Mode)



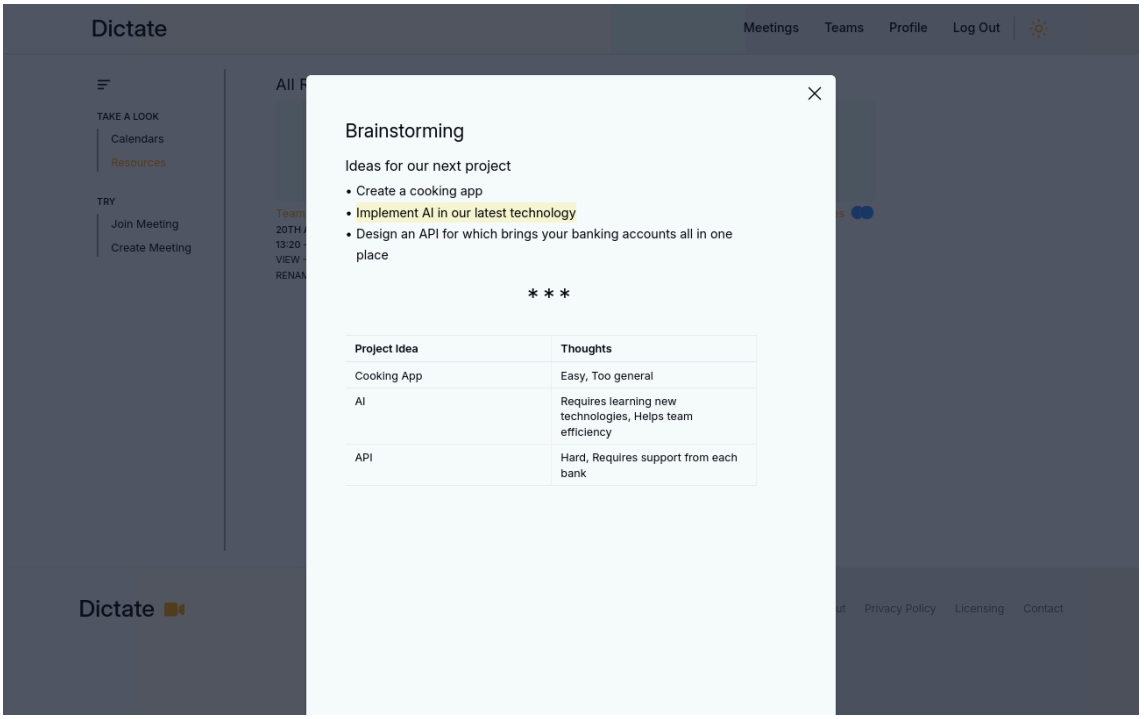
Calendars Page - Schedule Meeting Modal (Dark Mode)



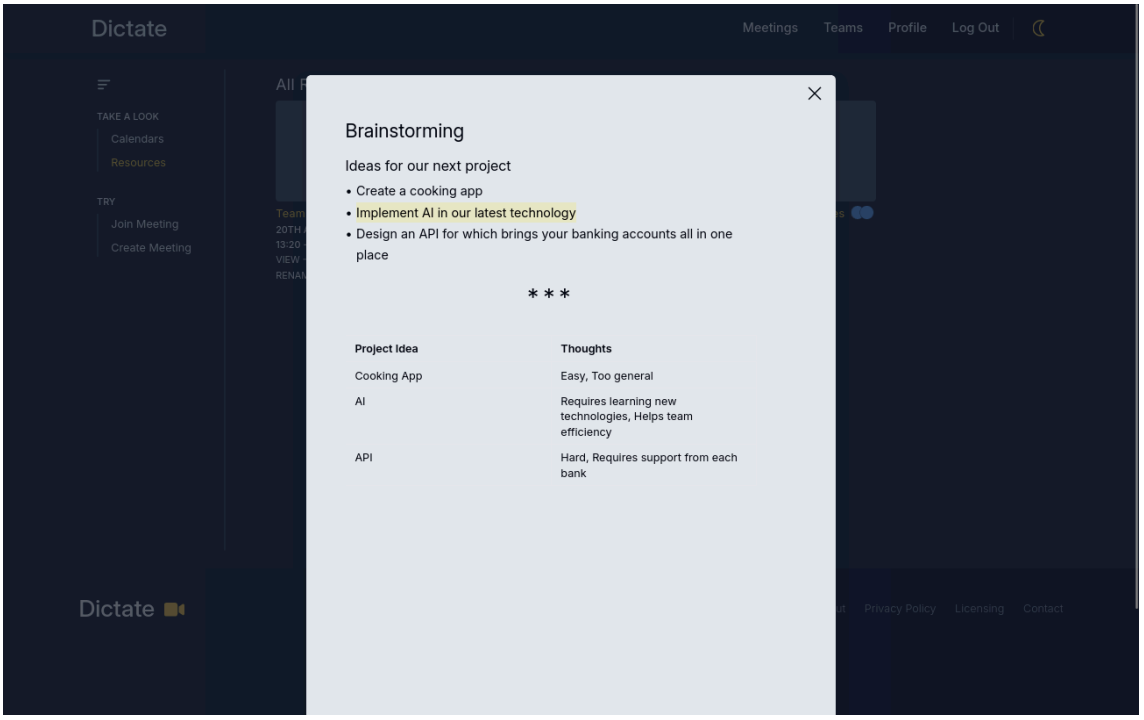
Resources Page (Light Mode)



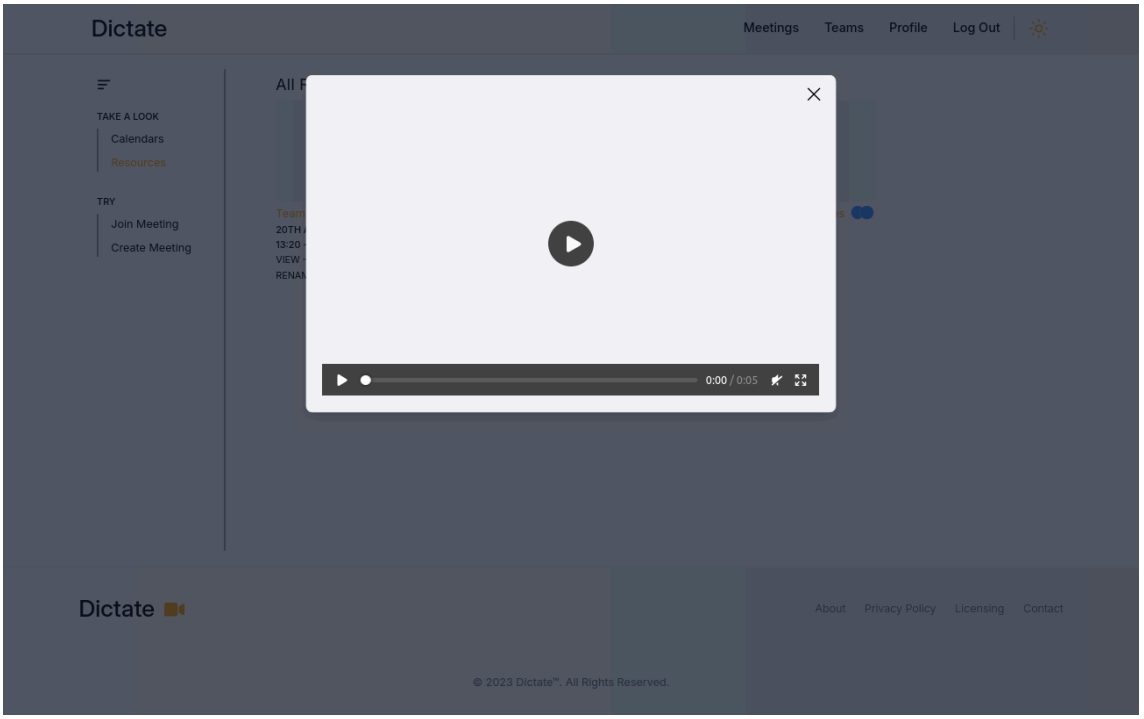
Resources Page (Dark Mode)



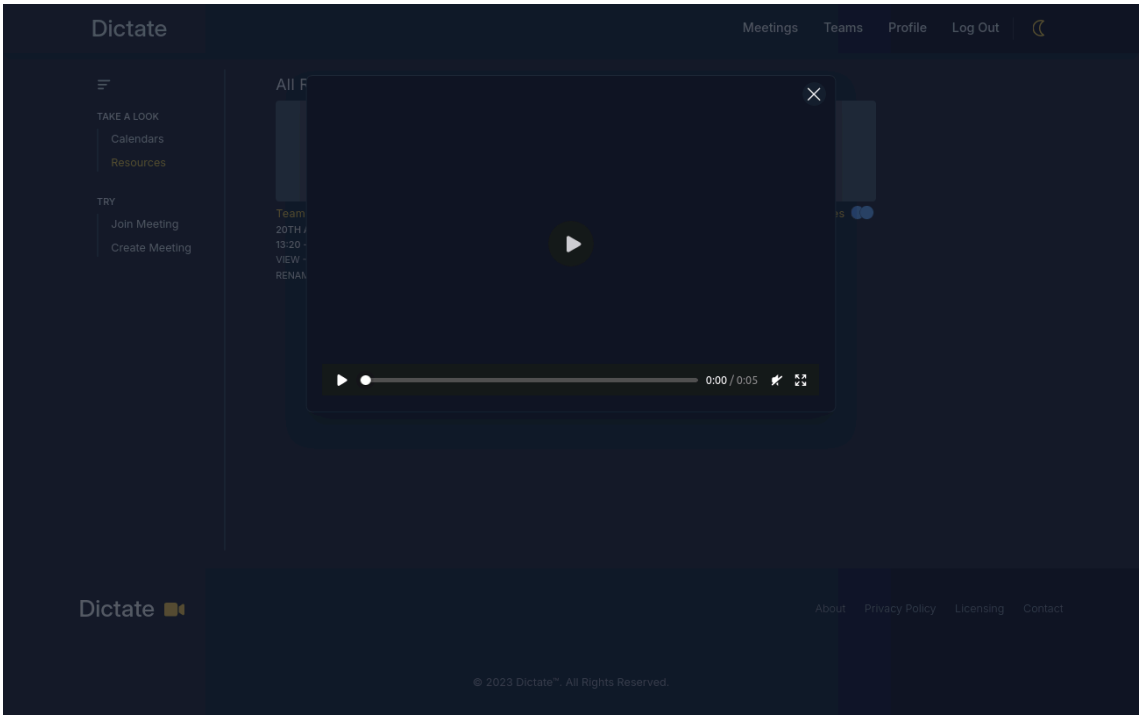
Resources Page - Notes Preview (Light Mode)



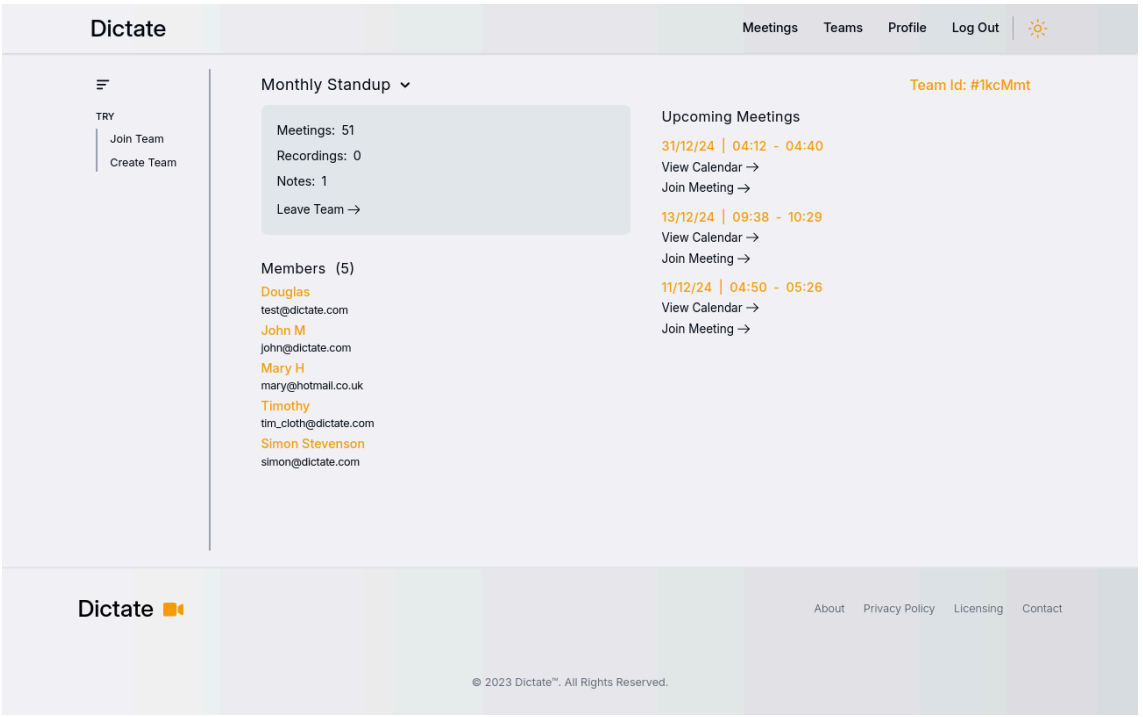
Resources Page - Notes Preview (Dark Mode)



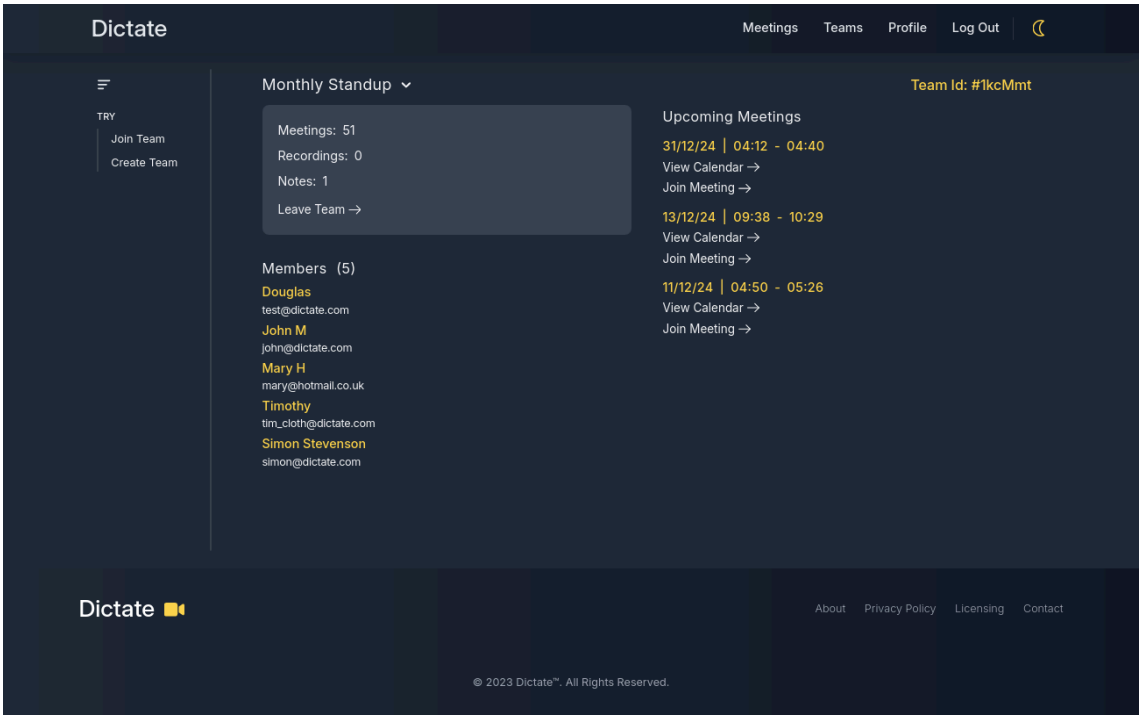
Resources Page - Recording Preview (Light Mode)



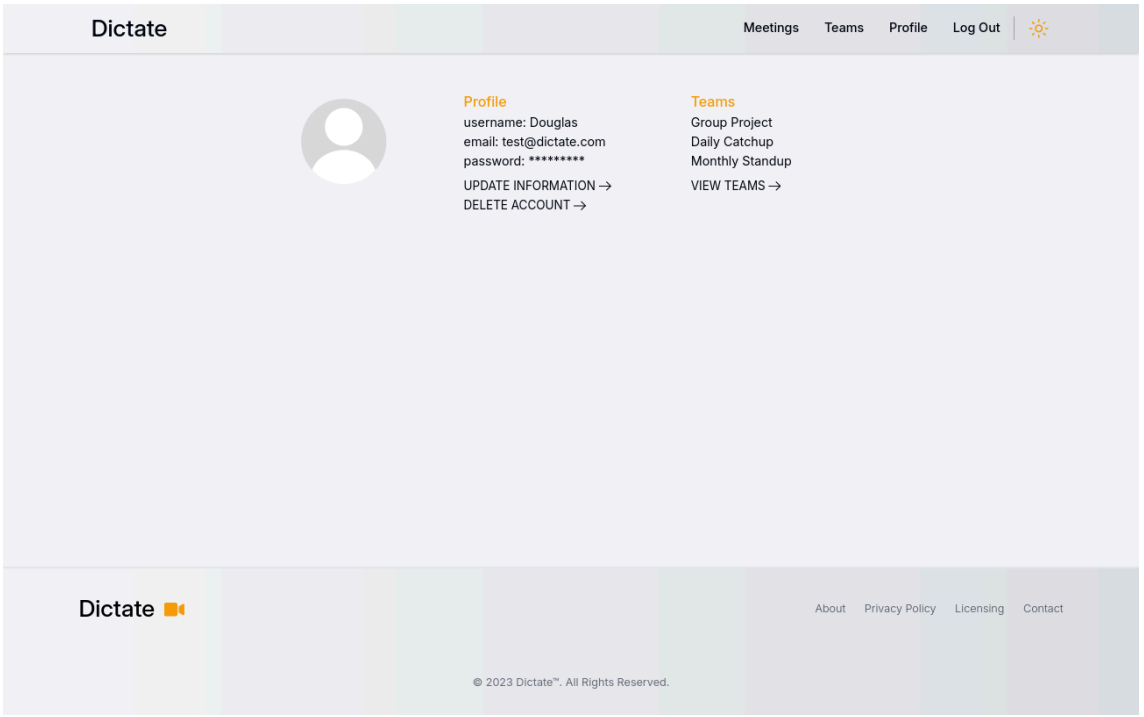
Resources Page - Recording Preview (Dark Mode)



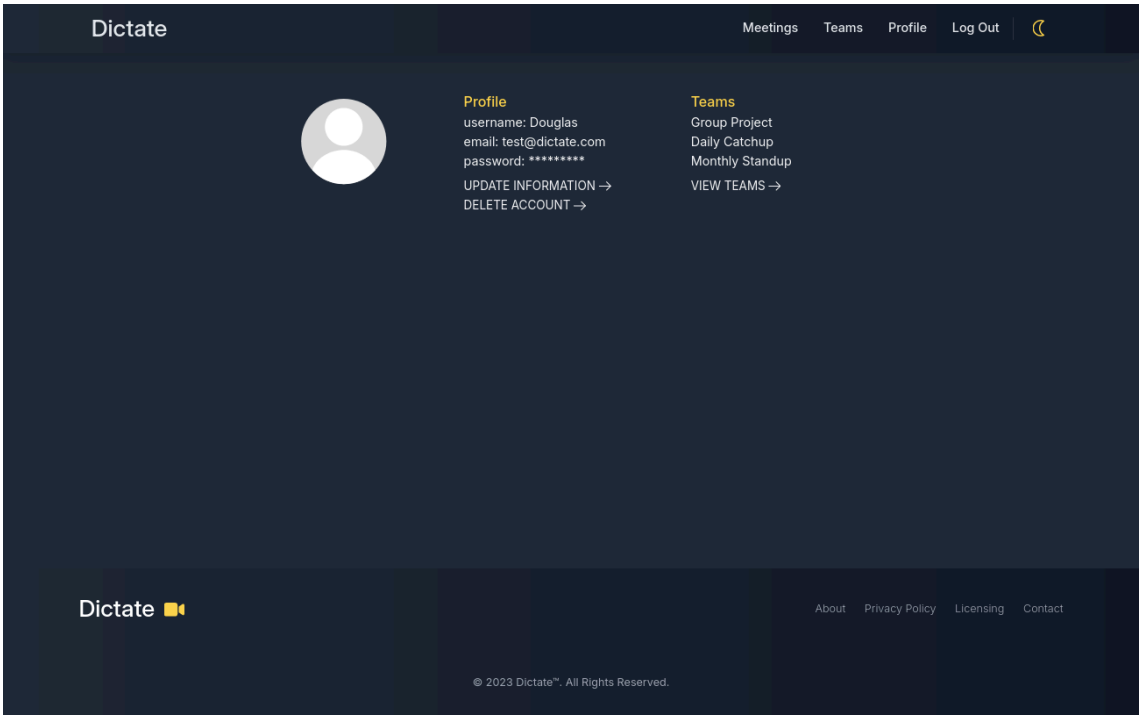
Teams Page (Light Mode)



Teams Page (Dark Mode)



Profile Page (Light Mode)



Profile Page (Dark Mode)

Appendix I: **Test Cases**

Test Case	Login
Frontend Unit Test(s)	describe("auth pages), it(“logs in successfully”)
	describe("auth pages”), it("shows error for bad password when logging in")
	describe("navbar”), it(“shows login and signup buttons if not logged in”)
	describe("navbar”), it(“navigates to login and signup pages successfully”)
	describe("navbar”), it(“shows page navigation if logged in”)
	describe("navbar”), it(“navigates to auth pages successfully”)
Backend Unit Test(s)	test_login_success()
	test_login_bad_credentials()
	test_login_bad_session()
Steps	1. Go to Login Page
	2. Enter email and password
	3. Click “Login”
Expected Result	1. Redirected to Calendars Page
	2. Access to protected routes (i.e. Calendars, Resources, Teams, Profile)
	3. ‘sessionid’ cookie set
	4. Error message displayed if bad email or password
	5. Error message displayed if already logged in
Pass/Fail	Pass

Test Case 1

Test Case	Logout
Frontend Unit Test(s)	describe("auth pages”), it(“logs out successfully”)
	describe("navbar”), it(“logs out successfully if logged in”)
Backend Unit Test(s)	test_logout_success()
	test_logout_bad_session()
Steps	1. Go to Navigation Bar
	2. Click Logout
Expected Result	1. Redirected to Home Page

	2. Denied access to protected routes (i.e. Calendars, Resources, Teams, Profile)
	3. ‘sessionid’ cookie unset
	4. Error if not logged in prior to attempting to log out
Pass/Fail	Pass

Test Case 2

Test Case	Signup
Frontend Unit Test(s)	describe("auth pages”), it(“signs up successfully”)
	describe("auth pages”), it(“shows error for bad email when signing up in”)
	describe("navbar”), it(“shows login and signup buttons if not logged in”)
	describe("navbar”), it(“navigates to login and signup pages successfully”)
	describe("navbar”), it(“shows page navigation if logged in”)
	describe("navbar”), it(“navigates to auth pages successfully”)
Backend Unit Test(s)	test_signup_success()
	test_signup_bad_required_fields()
	test_signup_bad_credentials()
Steps	1. Go to Signup Page
	2. Enter email, username, password and confirm password
	3. Click “Signup”
Expected Result	1. Redirected to Calendars Page
	2. Access to protected routes (i.e. Calendars, Resources, Teams, Profile)
	3. ‘sessionid’ cookie set
	4. Error message displayed if bad email, username, password and confirm password
	5. Error message displayed if already logged in
Pass/Fail	Pass

Test Case 3

Test Case	Get Profile
Frontend Unit Test(s)	describe("profile page”), it(“successfully loads”)
	describe("profile page”), it(“shows correct profile information”)
Backend Unit Test(s)	test_get_profile_success()
	test_get_profile_bad_session()
Steps	1. Go to Profile Page
Expected Result	1. Profile information should be correct
	2. Error if not logged in
Pass/Fail	Pass

Test Case 4

Test Case	Update Profile
Frontend Unit Test(s)	describe("profile page”), it(“opens update information modal successfully”)
	describe("profile page”), it(“updates profile information correctly”)
	describe("profile page”), it(“shows error when trying to update username to forbidden value”)
Backend Unit Test(s)	test_update_user_success()
	test_update_user_bad_session()
	test_update_user_bad_target()
Steps	1. Go to Profile Page
	2. Click Update Profile
	3. Update chosen field (e.g. email, username, password and/or confirm password)
	4. Click Go
Expected Result	1. Profile information should be updated
	2. Error message should be shown one of the fields contains a bad value
	3. Error if not logged in
Pass/Fail	Pass

Test Case 5

Test Case	Delete Profile
Frontend Unit Test(s)	describe("profile page”), it("successfully deletes profile")
Backend Unit Test(s)	test_delete_user_success()
	test_delete_user_bad_session()
	test_delete_user_bad_target()
Steps	1. Go to Profile Page
	2. Click Delete Profile
	3. Click Confirm in pop-up modal
Expected Result	1. Redirected to Home Page
	2. ‘sessionid’ cookie unset
	3. Unable to login with account details
	4. Error if not logged in
Pass/Fail	Pass

Test Case 6

Test Case	Get User
Frontend Unit Test(s)	describe("teams page”), it(“shows test user 2 as part of team members”)
Backend Unit Test(s)	test_get_user_success()
	test_get_user_bad_id()
Steps	1. Go to Teams Page
Expected Result	1. Details’ of team members should be visible
	2. Error if not logged in
Pass/Fail	Pass

Test Case 7

Test Case	Get Team
Frontend Unit Test(s)	describe("teams page”), it(“successfully loads”)
	describe("teams page”), it(“changes selected team successfully”)
	describe("teams page”), it(“shows current user as part of team members”)
	describe("teams page”), it(“shows meeting in upcoming meetings”)

	describe("resources page”), it(“filters by team successfully”)
	describe("calendars page”), it(“shows team dropdown filter on click”)
	describe("calendars page”), it(“filters calendar by team successfully”)
Backend Unit Test(s)	test_get_team_success()
	test_get_team_bad_id()
Steps	1a. Go to Teams Page
	1b. Click Select Team dropdown
	1c. Choose a team
	2a. Go to Calendars Page
	2b. Click on team filter
	2c. Choose a team
	2a. Go to Resources Page
	2b. Click on team filter
	2c. Choose a team
Expected Result	1. Teams Page should show correct details of selected team
	2. Calendars Page should show correctly filtered meetings of selected team
	3. Resources Page should show correctly filtered resources of selected team
	4. Error if not logged in
Pass/Fail	Pass

Test Case 8

Test Case	Join Team
Frontend Unit Test(s)	describe("teams page”), it(“opens join team modal successfully”)
	describe("teams page”), it(“successfully joins a team”)
Backend Unit Test(s)	test_join_team_success()
	test_join_team_bad_target()
Steps	1. Go to Teams Page
	2. Click on Join Team
	3. Enter team id
	4. Click Go
Expected Result	1. Team should be added to selector in dropdown

	2. Should be able to view team members of new team
	3. Error if not logged in
Pass/Fail	Pass

Test Case 9

Test Case	Leave Team
Frontend Unit Test(s)	describe("teams page"), it("opens leave team modal successfully")
	describe("teams page"), it("successfully leaves a team")
Backend Unit Test(s)	test_leave_team_success()
	test_leave_team_bad_target()
Steps	1. Go to Teams Page
	2. Choose a team
	3. Click on Leave Team
	4. Click on Yes in "Are You Sure?" Modal
Expected Result	1. Team should be removed from selector in dropdown
	2. Should not be able to view team details
	3. Error if not logged in
Pass/Fail	Pass

Test Case 10

Test Case	Create Team
Frontend Unit Test(s)	describe("teams page"), it("opens create team modal successfully")
	describe("teams page"), it("successfully creates a team")
Backend Unit Test(s)	test_create_team_success()
	test_create_team_bad_session()
Steps	1. Go to Teams Page
	2. Click on Create Team
	3. Enter team name
	4. Click Go
Expected Result	1. Team should be added to selector in dropdown
	2. Should be able to view team members of new team
	3. Should be able to see unique team id
	4. Error if not logged in

Pass/Fail	Pass

Test Case 11

Test Case	Update Team
Frontend Unit Test(s)	N/A
Backend Unit Test(s)	test_update_team_success() test_update_team_bad_session() test_update_team_bad_target()
Steps	1. Submit a PATCH request to “/api/teams/id/”
Expected Result	1. 200 status response 2. Error if trying to update field other than team name 3. Error if incorrect headers are sent 4. Error if not logged in
Pass/Fail	Pass

Test Case 11

Test Case	Delete Team
Frontend Unit Test(s)	N/A
Backend Unit Test(s)	test_delete_team_success() test_delete_team_bad_session() test_delete_team_bad_target()
Steps	1. Submit a DELETE request to “/api/teams/id/”
Expected Result	1. 204 status response 2. Error if trying to delete team not a member of 3. Error if not logged in
Pass/Fail	Pass

Test Case 12

Test Case	Get Meeting
Frontend Unit Test(s)	describe("calendars page”), it(“shows a meeting in the calendar”) describe("calendars page”), it(“shows meeting information when clicked”)

	describe("meetings page”), it(“opens meeting information modal successfully”)
	describe("meetings page”), it(“shows meeting id in information modal”)
	describe("meetings page”), it(“shows correct participants in information modal”)
Backend Unit Test(s)	test_get_meeting_success()
	test_get_meeting_bad_id()
Steps	1a. Go to Calendars Page
	1b. Click on a day of the month with meetings scheduled
	2a. Go to Home Page
	2b. Create a Meeting
Expected Result	1. Calendars Page should should correct meeting details
	2. Video-Conference Page should show correct meeting details in the information modal
Pass/Fail	Pass

Test Case 13

Test Case	Create Meeting
Frontend Unit Test(s)	describe("home page”), it(“opens start meeting modal successfully”)
	describe("home page”), it(“creates a new meeting successfully and redirects to page”)
	describe("calendars page”), it(“opens schedule meeting modal successfully”)
	describe("calendars page”), it(“schedules meeting successfully”)
	describe("calendars page”), it(“opens create meeting modal successfully”)
	describe("calendars page”), it(“creates meeting successfully from modal and redirects to video-conference page”)
	describe("resources page”), it(“opens create meeting modal successfully”)
	describe("resources page”), it(“creates meeting successfully from modal and redirects to video-conference page”)
Backend Unit Test(s)	test_create_meeting_success()
	test_create_meeting_bad_datetime()
Steps	1a. Go to Home Page/Calendars Page/Resources Page

	1b. Click Create Meeting
	2a. Go to Calendars Page
	2b. Click Schedule Meeting
	2c. Enter meeting details
	2d. Click Schedule Meeting
Expected Result	1. Creating a meeting should redirect to Video-Conference Page
	2. Scheduling a meeting should show scheduled meeting in calendar
	3. Error when scheduling meeting and not logged in
Pass/Fail	Pass

Test Case 14

Test Case	Update Meeting
Frontend Unit Test(s)	N/A
Backend Unit Test(s)	test_update_meeting_success()
	test_update_meeting_bad_session()
	test_update_meeting_bad_datetime()
Steps	1. Submit a PATCH request to “/api/meetings/id/”
Expected Result	1. 200 status response
	2. Error if trying to delete meeting not a participant of
	3. Error if not logged in
Pass/Fail	Pass

Test Case 15

Test Case	Delete Meeting
Frontend Unit Test(s)	N/A
Backend Unit Test(s)	test_delete_meeting_success()
	test_delete_meeting_bad_session()
	test_delete_meeting_bad_target()
Steps	1. Submit a DELETE request to “/api/meetings/id/”
Expected Result	1. 204 status response
	2. Error if trying to delete meeting not a participant of
	3. Error if not logged in

Pass/Fail	Pass

Test Case 16

Test Case	Join Meeting
Frontend Unit Test(s)	describe("home page"), it("opens join meeting modal successfully")
	describe("home page"), it("joins ongoing meeting successfully and redirects to page")
	describe("home page"), it("shows error when trying to join invalid meeting")
	describe("calendars page"), it("opens join meeting modal successfully")
	describe("calendars page"), it("joins meetings successfully from modal")
	describe("resources page"), it("opens join meeting modal successfully")
	describe("resources page"), it("joins meetings successfully from modal")
	describe("meeting page"), it("loads successfully")
	describe("meeting page"), it("shows set username modal first if not logged in")
	describe("meeting page"), it("shows local video stream in the corner of the page")
	describe("meeting page"), it("shows meeting controls at bottom of page")
Backend Unit Test(s)	test_websocket_connection_success_anonymous_user()
	test_websocket_connection_success_authenticated_user()
	test_websocket_connection_bad_id()
	test_websocket_connection_bad_meeting()
	test_websocket_user_joined_success_anonymous_user()
	test_websocket_user_joined_success_authenticated_user()
	test_websocket_webrtc_message_success()
	test_websocket_webrtc_message_bad_peer()
Steps	1. Go to Home/Calendars/Resources Page
	2. Click Join Meeting

	3. Enter Meeting Id
	4. Click Go
Expected Result	1. Redirected to Video-Conference Page
	2. Participant streams are visible on the page
Pass/Fail	Pass

Test Case 17

Test Case	Leave Meeting
Frontend Unit Test(s)	describe("meeting page"), it("loads successfully")
	describe("meeting page"), it("opens exit meeting modal successfully")
	describe("meeting page"), it("exits room successfully and redirects")
Backend Unit Test(s)	test_websocket_disconnect_success()
Steps	1. Join a Video-Conference
	2. Click on Exit Meeting in Controls
	3. Click on Yes in "Are You Sure?" Modal
Expected Result	1. Redirected to Home Page
Pass/Fail	Pass

Test Case 18

Test Case	Get Recording
Frontend Unit Test(s)	describe("resources page"), it("successfully loads")
	describe("resources page"), it("shows recording preview on click")
	describe("resources page"), it("filters by recordings successfully")
Backend Unit Test(s)	test_get_recording_success()
	test_get_recording_bad_id()
Steps	1. Go to Resources Page
	2. Click on a recording
Expected Result	1. Resources Page should show the initial frame of the recording as a preview along with resource details (name, date of meeting)
	2. Clicking on the recording should allow to play the whole recording

	3. Error if not logged in
Pass/Fail	Pass

Test Case 19

Test Case	Create Recording
Unit Test(s)	describe("meeting page"), it("opens start recording modal successfully")
	describe("meeting page"), it("prompts guest user to log in before recording")
	describe("meeting page"), it("successfully records meeting")
Backend Unit Test(s)	test_create_recording_success()
	test_create_recording_bad_file()
	test_task_transform_video_success()
	test_task_transform_video_error_success()
Steps	1. Join Video-Conference meeting
	2. Click Start Recording
	3. Click Stop Recording
Expected Result	1. "Successfully Captured Recording" Modal should appear
	2. Recording should be visible in Calendars Page
	3. Error if attempting to record and not logged in
Pass/Fail	Pass

Test Case 20

Test Case	Update Recording
Frontend Unit Test(s)	describe("resources page"), it("shows rename modal on click")
	describe("resources page"), it("successfully renames a resource")
Backend Unit Test(s)	test_update_recording_success()
	test_update_recording_bad_session()
Steps	1. Go to Resources Page
	2. Click Rename Resource
	3. Enter new name
	4. Click Go

Expected Result	1. Resource should be renamed 2. Error if not logged in
Pass/Fail	Pass

Test Case 21

Test Case	Delete Recording
Frontend Unit Test(s)	N/A
Backend Test(s)	test_delete_recording_success() test_delete_recording_bad_session() test_delete_recording_bad_target()
Steps	1. Submit a DELETE request to “/api/recordings/id/”
Expected Result	1. 204 status response 2. Error if trying to delete recording not belonging to profile 3. Error if not logged in
Pass/Fail	Pass

Test Case 22

Test Case	Get Notes
Frontend Unit Test(s)	describe("resources page”), it(“shows notes preview on click”) describe("resources page”), it(“filters by notes successfully”)
Backend Unit Test(s)	test_get_notes_success() test_get_notes_bad_id()
Steps	1. Go to Resources Page 2. Click on a note
Expected Result	1. Resources Page should show resource details (name, date of meeting) 2. Clicking on the note should show the contents 3. Error if not logged in
Pass/Fail	Pass

Test Case 23

Test Case	Create Notes
Frontend Unit Test(s)	describe("meeting page”), it(“hides meeting notes on first load”)
	describe("meeting page”), it(“opens meeting notes successfully”)
	describe("meeting page”), it(“writes text to group notes successfully”)
Backend Unit Test(s)	test_websocket_group_notes_message_success()
	test_websocket_group_notes_message_bad_content()
Steps	1. Go to Video-Conference Page
	2. Click Group Notes Button
	3. Add text to notes
Expected Result	1. Text should be persistent across screen re-renders
	2. Text should be seen by other users
	3. Notes should be available in Resources Page
Pass/Fail	Pass

Test Case 24

Test Case	Update Notes
Frontend Unit Test(s)	describe("resources page”), it(“shows rename modal on click”)
	describe("resources page”), it(“successfully renames a resource”)
Backend Unit Test(s)	test_update_notes_success()
	test_update_notes_bad_session()
	test_update_notes_bad_target()
Steps	1. Go to Resources Page
	2. Click Rename Resource
	3. Enter new name
	4. Click Go
Expected Result	1. Resource should be renamed
	2. Error if not logged in
Pass/Fail	Pass

Test Case 25

Test Case	Delete Notes
Frontend Unit Test(s)	N/A
Backend Unit Test(s)	test_delete_notes_success() test_delete_notes_bad_session() test_delete_notes_bad_target()
Steps	1. Submit a DELETE request to “/api/notes/id/”
Expected Result	1. 204 status response 2. Error if trying to delete note not belonging to profile 3. Error if not logged in
Pass/Fail	Pass

Test Case 26

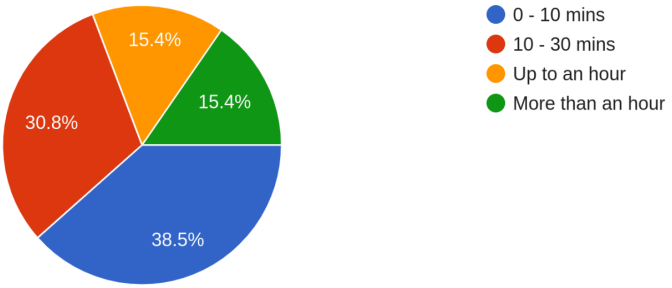
Test Case	Get Calendar
Frontend Unit Test(s)	describe("calendars page”), it(“loads successfully”) describe("calendars page”), it(“shows current month as default month”) describe("calendars page”), it(“toggles to next month successfully”) describe("calendars page”), it(“toggles to previous month successfully”)
Backend Unit Test(s)	N/A
Steps	1. Go to Calendars Page 2. Click Next/Previous Month
Expected Result	1. Calendar should load with correct meetings 2. Month should update appropriately after clicking Next/Previous Month
Pass/Fail	Pass

Test Case 27

Appendix J: **Feedback Survey**

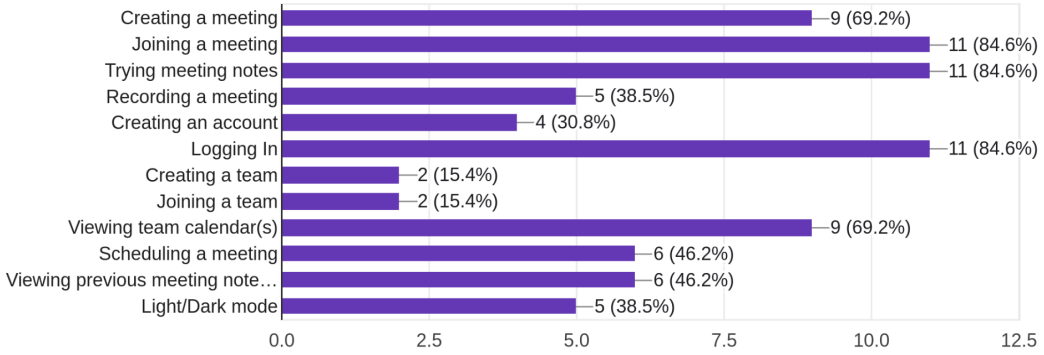
How long did you use the app for

13 responses



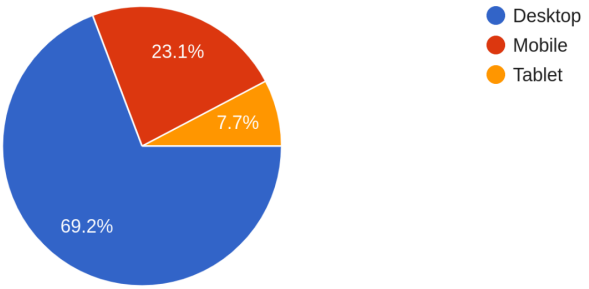
Which functionalities did you try?

13 responses



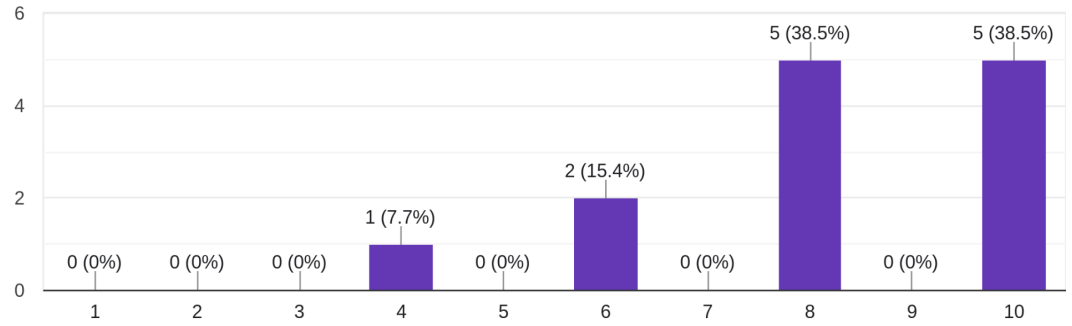
How did you access the platform?

13 responses



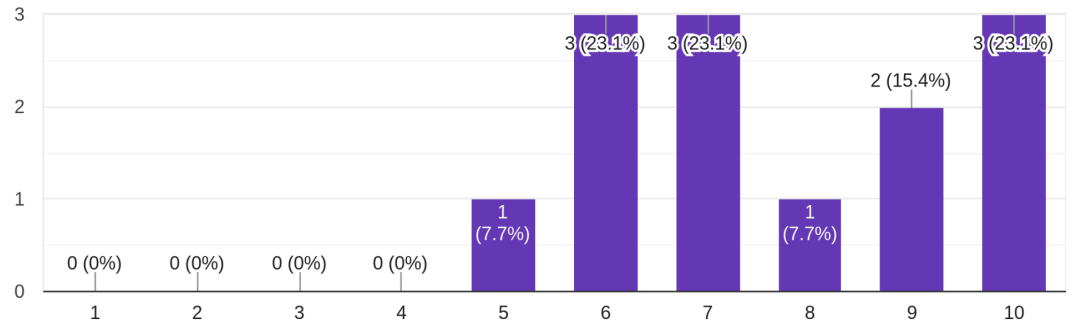
How did you find the platform

13 responses



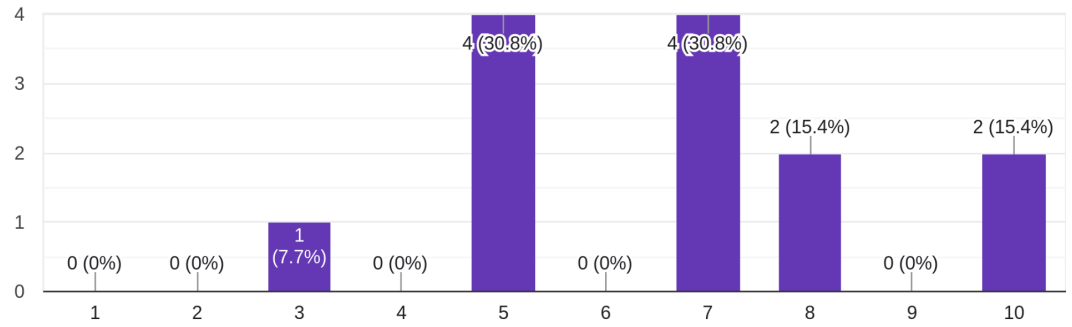
How likely would you be to recommend others to use the platform?

13 responses



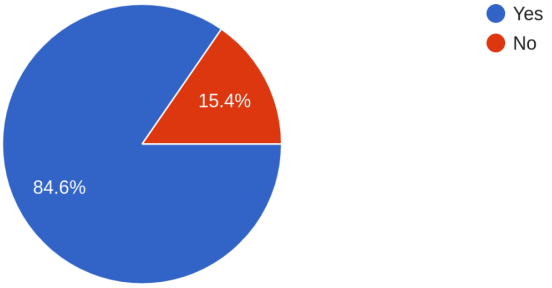
How would you compare the platform to other tools in the market?

13 responses



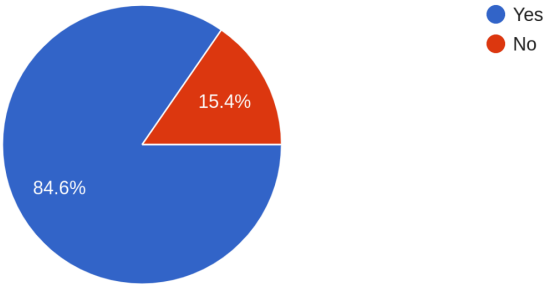
Did trying out the platform better your perception of video-conference technologies as a whole?

13 responses



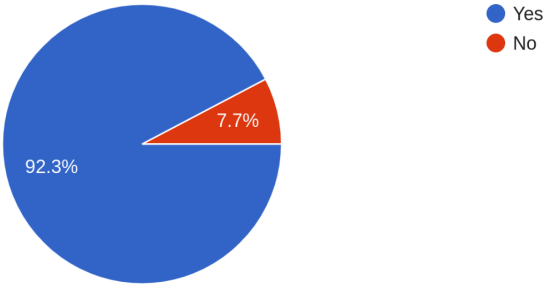
Did the platform increase your likelihood of selecting video-conferencing as opposed to a face-to-face meeting in the future?

13 responses



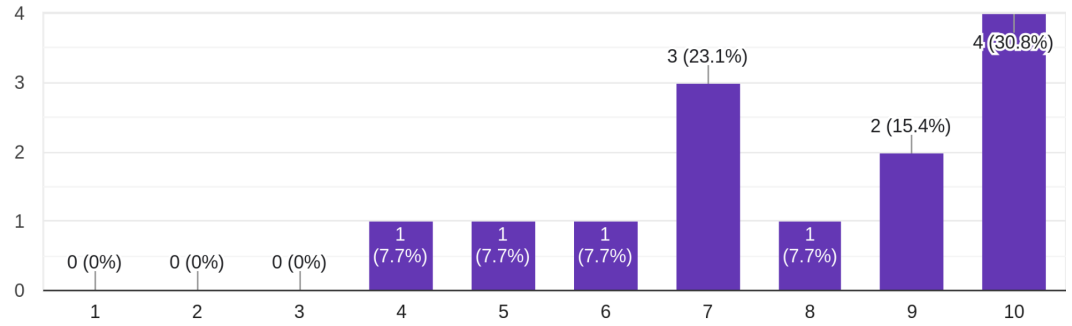
Did you like the design of the platform?

13 responses



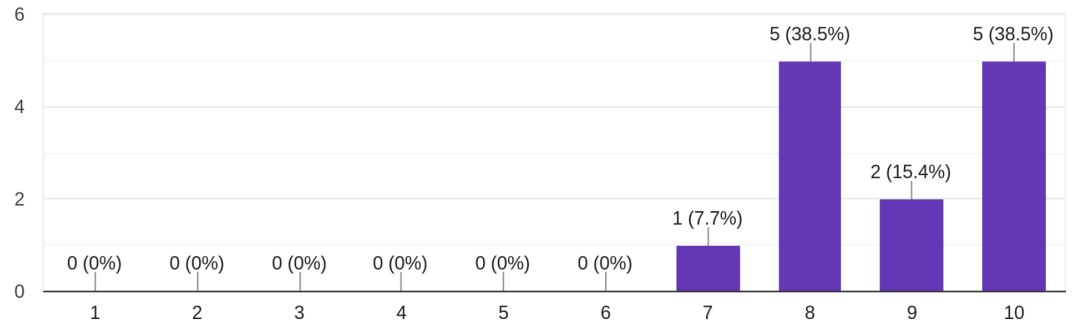
I found the design to positively affect my experience testing out the platform

13 responses



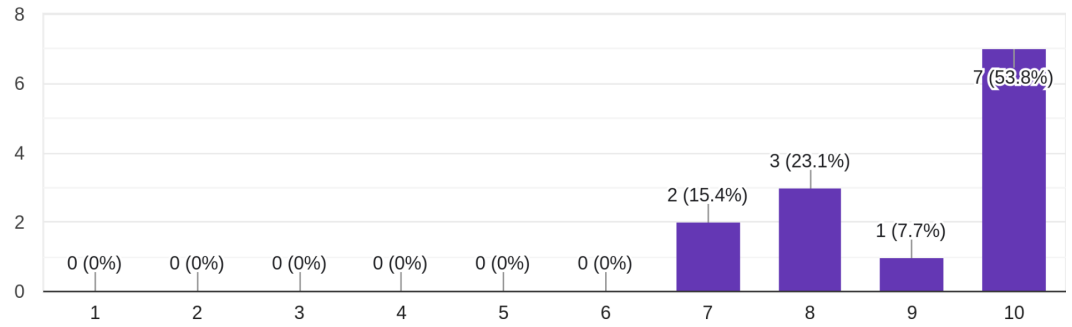
I found the platform easy to use

13 responses



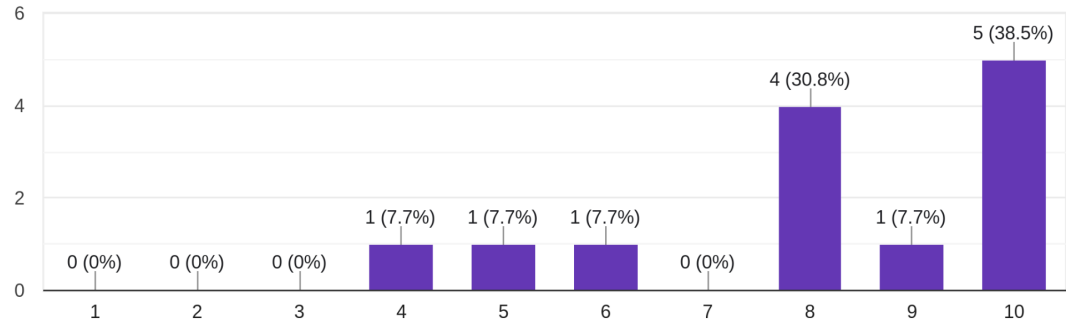
I found the platform fast and responsive

13 responses



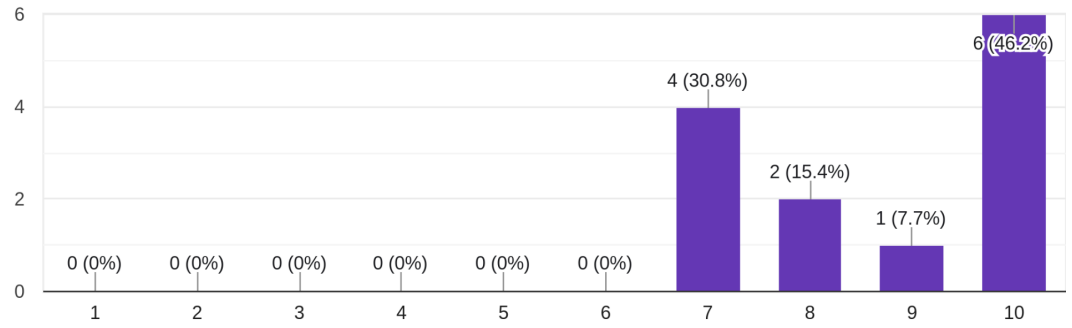
I found the design intuitive

13 responses



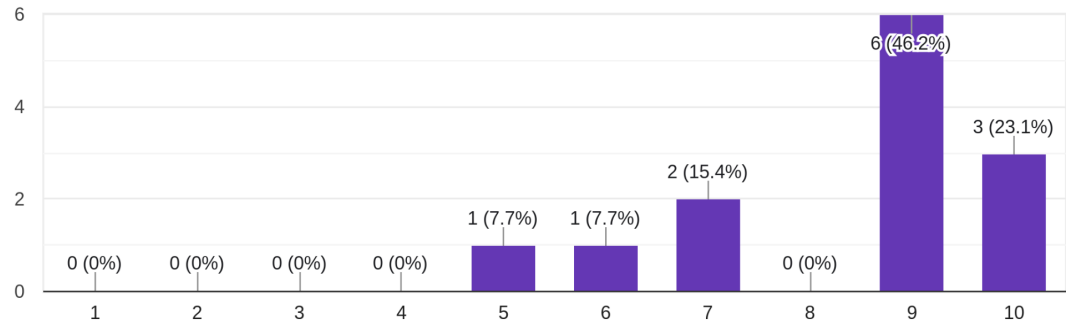
I found the video-conference to have good video and audio quality

13 responses



I found the platform to support group work

13 responses



What functionalities (if any) did you like about the platform?

13 responses

- Loaded quickly, chat was positioned well and worked well, both audio and video were surprisingly really good quality
- Writing on the side
- Shared notes
- I enjoyed the simplicity of the website, making it intuitive and straight forward.
- The meeting notes were very useful and you could use many functions within that. The calendar was also very useful and easy to see your schedule.
- calendar functionalities
- Simple UI
- Easy to join the call, no login or lengthy process to join, video and audio quality was great, tested the platform on both MS Edge and Chrome - both works fine
- Being able to write notes together and edit them
- Professional looking site that functions quickly.
- Users able to note take and allows users to edit during video conferencing. Brain storming/discussions captured live through note taking
- How easy it is to host a meeting
- I like the ease of logging in and adding your name. The ease of the entry
- The images and the way in which they are clear to see.

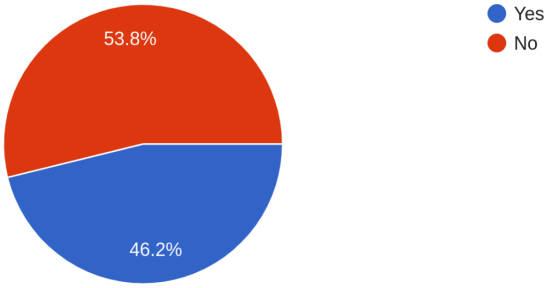
What functionalities (if any) did you dislike about the platform?

13 responses

- A little more work on ux
- Dosen"t make notes automatically
- Better experience on desktop compared to mobile
- None
- None.
- none
- UI could be better
- Visually not appealing, would be great to customise the background or have set background which prompts the user if they want to change it. When turning of camera, this also turned off the mic, there needs to be a function that turn camera off only - or a simple instruction showing the user what to do, on both Chrome and Edge when I turned this off the icon disappeared and took a while to navigate to the options to turn back on - something that wasn't user friendly and could be frustrating. Chat function doesn't allow for sharing images or emojis - it is also very flat/basic. The 4 icons are not clear - when hovering mouse over them would be good to get a prompt to allow the user to understand what the buttons do. No button to easily access the homepage to login/make a username etc
- Make the buttons in the call more clear
- No dislikes.
- Video and note taking layout needs to be improved by the option of customizing
- NA
- Lack of emotion signalling. The chat feature I would have preferred this at the side of the page as opposed to the top. I feel the chat feature should have a text box with a slightly different colour to make it clear that this is another section of the page.

Were there any problems/bugs found when trying out the platform?

13 responses



If so what were they?

6 responses

- Joining using meeting code alone did not work
- Input camera was using the wrong camera causing me to not load in properly
- couldn't hear peoples voices
- As per comments above
- The mic not working when recording a meeting
- When I went to send a message and hit return the screen took me back to the main menu. This is due to the fact there are no menu in this section of the page or separaters.

What feedback can you give me regarding the platform?

13 responses

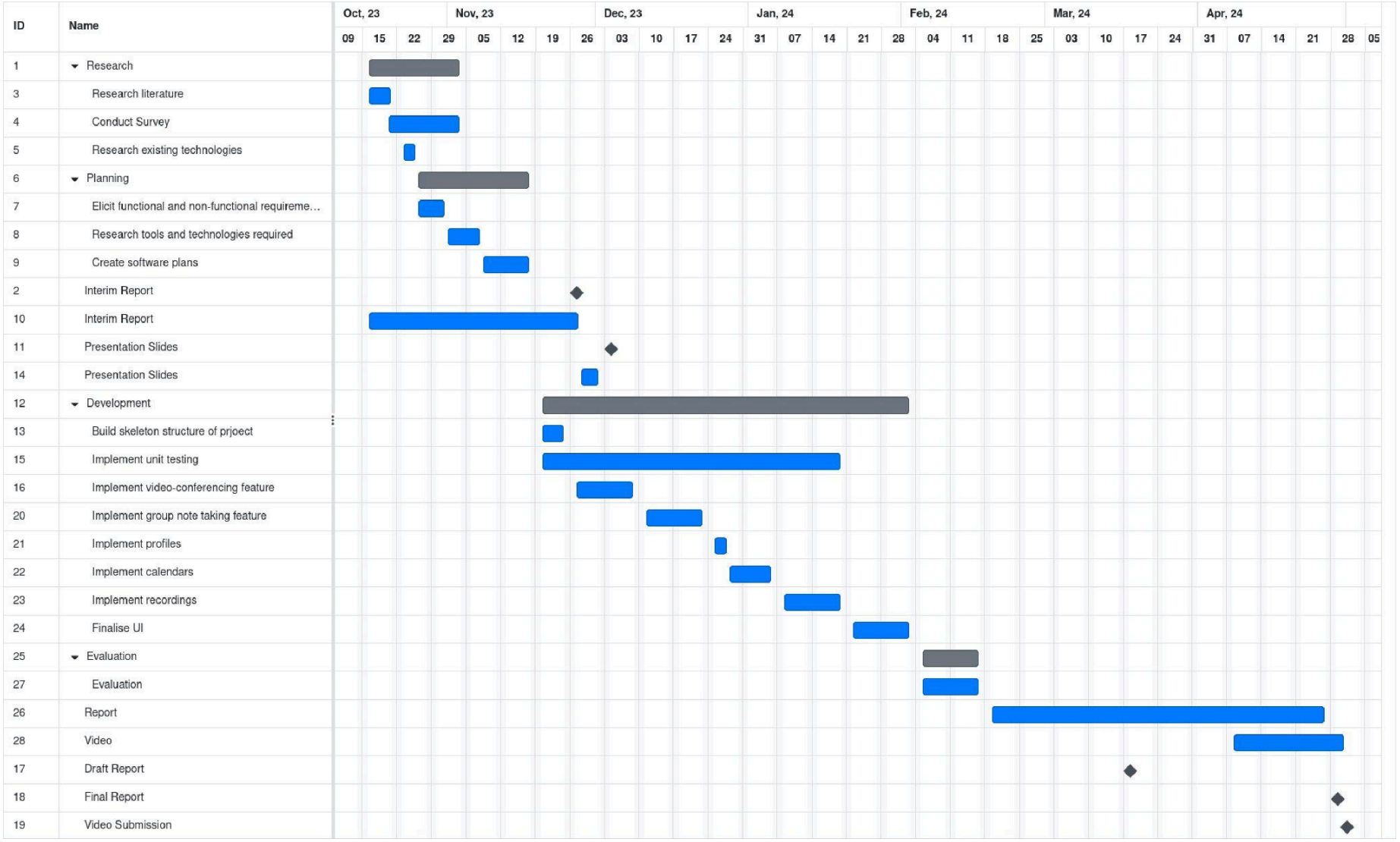
- I think it's a really good beta product, bug free, so is actually a good candidate for an MVP....a little more work on design layout/UX and you might be onto something!
- N/a
- Great start! Consider adding Animated Avatars in fitted versions
- I think the addition of the ability to mute and deafen would be beneficial
- The user interface was very well made and easy to use!
- make the leave call icon a red colour when hovered over
- improve UI
- Platform in principle is great, its fast, effective, and does what I initially expected, if the above comments can be implemented, this is something I would be happy to use as my go to platform
- Good platform which achieves its goals. Could perhaps improve the aesthetics of the website. Make the user interface clearer and more visually pleasing.
- Very well put together, has everything you would want for a site of this purpose.
- Advertise functionality specifically note taking to capture users. Overall I was able to easily use on browser and ios. Like the functionality of able to view past recordings/meetings on the site. I liked that users can access via a link without providing email or logging in via an account. I think improving the video layout when there are multi users and automatically adapt to users when speaking. Also, I would like note taking to have more functionality similar to that of word document. I found it helpful that you can schedule/store previous meetings in a calendar.
- Don't forget us when you make it

- Very nice idea . Needs more functionality when you are in the meetings and ways to show your feelings in the meeting. Need timer and possible show of names incase u are in a meeting where you do not know everyones names

Appendix K: **Risk Assessment**

Description of Risk	Description of Impact	Likelihood Rating	Impact Rating	Preventative Actions
Loss or corruption of work	Starting the project again from the beginning	Medium	High	Backing up data to the cloud or hardware device (e.g. USB)
Illness or unforeseen circumstances	Inability to work for a period of time	Medium	Medium	Spreading work out as much as possible and not leaving it to the last minute
Time constraints	Not enough time to finish project	High	Medium	Scheduling time effectively and prioritising correctly
Inability to conduct requirements survey	Not eliciting requirements effectively	Low	Medium	Plan out desired respondents and leave enough time for them to fill out results
Inability to find necessary libraries/apis for a working system	Not producing a working system	Medium	High	Choose a popular programming language with lots of resources and external help
Introducing breaking changes to the system	Not producing a working system	Low	Low	Using version control such as git
Project could break on deployment	Not producing a working system	High	Medium	Developing in a sandbox environment such as docker which can be replicated on deployment
Inability to learn new tools required	Not producing a working system	Medium	High	Have alternate options to potential libraries/apis in case the learning curve is too steep
Inability to conduct feedback survey	Inability to complete validation/evaluation sections	Medium	Medium	Plan out desired respondents and leave enough time for them to fill out results

Appendix L: **Original Work Plan**



Appendix M: **Revised Work Plan**