Planning Report

[Requirements] Elicitation

> Find 2-3 people to interview as target users. Target users are people who currently use a tool like Streams, or intend to. Collect their name and email address.
> Develop a series of questions to ask these target users to understand what problems they might have with teamwork-driven communication tools that are currently unsolved by Streams. Give these questions to your target users and record their answers.
>
> Once you have done this, think about how you would solve the following problem and write down a brief description of a proposed solution.

Questions

1. What features would you like to see in team communication tools such as Microsoft Teams?
2. What do you dislike about Microsoft Teams?
3. What do you like about Microsoft Teams?
4. What might you change about Microsoft Teams?
5. Are there any features on similar tools that you like/dislike?
6. Why do you use Microsoft Teams?

Problems

Response - Matthew Kim

Email: jongjeh@gmail.com

Q1 - Missing Features

- Customising theme + reacts
- LaTeX support in chat
- Archiving/exporting chat messages
- More channel settings
- Participants drawing on shared screen

- Using other services eg. (Google + Facebook) to login

Q2 - Dislikes

- Pretty laggy
- Doesn’t suit use case as a student (Too many features unused)

Q3 - Likes

- Integration with Office 365
- Replying to threads

Q4 - Changes

- Different versions for different use cases (Eg. Enterprise vs Education)
- Somehow make it run faster

Q5 - Features from Similar Tools

- Zoom has a less bulky UI/UX

Q6 - Use

- Very popular
- Compulsory for accessing course content

Response - Trent Zeng

Email: z.trent.w@gmail.com

Q1 - Missing Features

- Joining a call from two devices
- Archiving chat messages
- Drawing on shared screen
- Customising theme

Q2 - Dislikes

- The share screen option is confusing while you are sharing

- Doesn’t show when your microphone sound is picked up
- Lags

Q3 - Likes

- It automatically decides to mute your microphone when there’s a lot of people in a meeting already
- File tab is useful
- Adding own reacts
- Uploading files

Q4 - Changes

- Combine the sidebar when you’re in a team with the top one to make it easier to find things.
- UI/UX Fixes

Q5 - Features from Similar Tools

- Zoom tends to have better video quality but the chat function outside of calls can get confusing with the way replying to other messages is laid out.
- Improve video quality so it matches Zoom

Q6 - Use

- University uses it for lectures and to have access to student communications in some classes

---

List of Problems

- Missing features
- Video Call
- Drawing on shared screen
- Chat
- Archiving messages
- LaTeX support
- Threads
- Customisation
- Themes

- Reacts
- Experience
- UI/UX
- Lag
- Integration
- Login with Google
- Microsoft 365

Proposed Solutions

> Missing Features: create an upcoming features list and slowly release this feature throughout development.

Missing Feature - Video Call

- Joining a call from two devices
- Participants drawing on shared screen
- Video call

Missing Feature - Chat Features

- Archiving chat messages
- Customising theme + reacts
- LaTeX support in chat
- Threads

Missing Feature - Customisation

- Themes
- Reacts

Missing Feature - Integration

- Login with Google
- Microsoft 365

General Problem - UI/UX

- Rework UI to be easy to use/customisable

- Create another survey and ask specifically what areas of UI/UX needs work and why they might prefer it, then refactor the front-end design.

General Problem - Lag

- Identify potential areas that might contribute to lag and refactor them.
- Potential locations include:
    - Datastore
    - Front-End

[Requirements] Analysis & Specification - Use Cases

> Once you've elicited this information, it's time to consolidate it.
> Take the responses from the elicitation and express these requirements as user stories. Document these user stories. For each user story, add user acceptance criteria as notes so that you have a clear definition of when a story has been completed.
>
> Once documented, generate at least one use case that attempts to describe a solution that satisfies some of or all the requirements elicited. You can generate a visual diagram or a more written-recipe style, as per lectures.

User Story 1

As a student, I want to be able to interact with the class and lecturers in a fun and productive environment, especially through a video call.

User Acceptance Criteria

- There is a button to start a call in a channel
- The call starts once it is pressed, alerting all currently logged in channel members
- Users can then elect to join the call via the popup
- There are options to toggle video and microphone input
- The call ends either when everyone leaves, or when the channel owner/user who
- started the call ends the call

User Story 2

As a student, I want to be able to customise my application so it doesn’t look boring.

User Acceptance Criteria

- The user can change the 'theme' of their application.
- Given that I have created an account, I can edit the primary colours of the page.
- The user can add custom Reacts.
- Given that I am in a specific channel, I can add custom reacts to messages.

User Story 3

As a student, I find it annoying having to remember multiple accounts for university so I’d like it if we could use other details, such as our google accounts or even university details to login.

User Acceptance Criteria

- The user can create an account with a Google account.
- I should be able to press a button that takes me to a Google authorisation page.
- This should then create an account associated with that Gmail account after I have been authorised.

User Story 4

As a student, I would like to be able to archive/export chat files and messages as well as upload images to the channel.

User Acceptance Criteria

- The user can upload images to a channel.
- The user can press a button to upload an image to a channel.
- This should be posted as an image.
- The user can download the files and messages associated with a channel.
- The user can press a button to download a .zip file with a .txt of all messages and a separate folder with a list of images.

User Story 5

As a UNSW maths student, I would like to be able to send LaTeX formatted messages so I can share and discuss math problems with my peers.

User Acceptance Criteria

- There is a LaTeX button on the message bar
- When clicked a popup with a text box opens, allowing LaTeX to be inputted
- When the user clicks 'ok' their message is added to the buffer in the message bar
- Additionally by wrapping their message with '$$' it will automatically be formatted in LaTeX when sent
- Editing messages will update LaTeX accordingly

Use Cases

Use Case 1 - Login with Google

MAIN SUCCESS SCENARIO

1. User clicks signup with Google
2. Webpage redirects to Google Login
3. Google asks for password
4. User successfully logins
5. An account is registered for the user
6. User successfully logs in

Goal in Context:
Ability to register and login with another service account

Scope:
Google's authorisation servers

Level:
Subfunction.a

Preconditions:
The user has a Google account

Success End Condition:
The customer has created an account associated with aGoogle account and can login with it

Failed End Condition:
The customer can’t generate or login with a Google account

Primary Actor:
User

Trigger:
User clicks the register or login button

Use Case 2 - VC (video/voice call) channel

MAIN SUCCESS SCENARIO

1. User clicks the Join Voice call button
2. Server checks whether the user has a valid token and is a member of channel
3. Server allows user to enter video/voice call
4. User successfully joined

Goal in Context:
Ability for users to have the option to enter a VC

Scope:
Servers

Level:
Subfunction.a

Preconditions:
The user has an account on streams and is a member of the same channel/dm with the call recipient

Success End Condition:
The user can enter a video/voice call with one or multiple desired members from a channel

Failed End Condition:
N/A

Primary Actor:
User

Trigger:

User clicks the ‘join call’ button.

Use Case 3 - Ability to upload files/photos/latex integration

MAIN SUCCESS SCENARIO

1. User clicks button to send files/photos
2. User is prompted to upload file
3. User uploads file
4. Server checks user has valid token and is a member of same channel/dm as the recipient
5. Server sends file on successful request
6. User and recipient(s) can view file

Goal in Context:
Ability for users to have the option to send files/photos

Scope:
Servers

Level:
Subfunction.a

Preconditions:
The user has an account on streams and is a member of the same channel/dm with the desired message recipient

Success End Condition:
The user can share a file with the desired recipients from a channel/dm

Failed End Condition:
The user fails to share

Primary Actor:
User

Trigger:
User clicks the share file button.

[Requirements] Validation

> With your completed use case work, reach out to the 2-3 people you interviewed originally and inquire as to the extent to which these use cases would adequately describe the problem they're trying to solve. Ask them for a comment on this, and record their comments in the PDF.

Use case 1

Matthew

Being able to login with google would fix my major gripe with using team communication tools and its issues concerning ease of use. It doesn’t make it a more compelling choice for everyday use, however, it makes it less tedious to use when the uni does force you to use it.

Trent

Logging in with google wouldn't really change any current interactions I have with team communication tools. It would sometimes be more convenient, ie, logging in from a new device, but it doesn’t really address my current issues with teams. I mainly just wanted to see an external indicator that relays whether or not the microphone is in use. I would be much less paranoid if other people could hear me whilst I'm on another screen.

Use case 2

Matthew

The ability to hold voice calls with other people would assist in teamwork efficiency and general productivity. In fact, I think that it would cut down a lot of time spent fumbling around and struggling to organise things. It's a much more efficient medium to convey ideas compared to text.

Trent

An issue that I had with some other team communication tools such as teams was that the screen share option felt clunky and non-intuitive. I’m assuming that the implementation of a vc would include a screen share option so hope that it includes features that would set it apart from other team communication tools. I’m really just looking to see if it has an external microphone use indicator and an intuitive screen share interface.

Use case 3

Matthew

I personally think that latex integration has been the most compelling feature so far. Due to my math heavy workload, there were multiple occasions that I wished that teams software came with latex integration; as it would've made my life a whole lot easier.

The ability to send an equation with integrated latex formatting just makes the ideas and questions I send so much more legible, saving the inevitable need to clarify later. Sending images also would make communicating with others a lot easier, saving the need to explain things with text. Altogether, I think these features seem to be the most useful and helpful out of everything so far.

Trent

Being able to upload files and images is a pretty important feature on most other communication services, thus, its utility for students isn't something that really needs to be stated. Personally, I wouldn't really use the latex integration feature, but I suppose it would be useful for a student majoring in maths. Regardless of how file/photo upload is implemented, it would still be a useful feature that would see extensive use. I think file/photo upload is one of the features where you won’t really notice until you need to use it, but you’ll really miss it if your communication tool doesn't have it.

[Design] Interface Design

> Now that we've established our problem (described as requirements), it's time to think about our solution in terms of what capabilities would be necessary. You will specify these capabilities as HTTP endpoints, similar to what is described in 6.2. There is no minimum or maximum of what is needed - it will depend on what problem you're solving.

call/start/v1

- Description:
  - Starts a voice call in the respective channel or dm. The creator automatically joins the call with all currently logged in members of the DM/channel. The call automatically ends when there are no existing members in

the call. `channel_id` is `-1` if the call is starting in a DM, and `dm_id` is `-1` if the call is starting in a channel.

- Method:
  - `POST`
- Parameters:
  - `{ token, channel_id, dm_id }`
- InputError:
  - Both `dm_id` and `channel_id` are invalid
  - Neither `dm_id` nor `channel_id` are `-1`
  - If the pair of `channel_id` and `dm_id` are valid and there is already a call active in the DM
  - `token` is invalid
- AccessError:
  - The pair of `channel_id` and `dm_id` are valid (i.e. one is `-1`, the other is valid) and the authorised user has not joined the channel or DM they are trying to start a call in
- Return Type `{ call_id }`

call/mute/v1

- Description:
  - Given a token, a call_id and that the user's microphone is unmuted, the user's microphone is muted; meaning, anything the user says through their microphone will not be heard by the other participants in the call.
- Method:
  - `POST`
- Parameters:
  - `{ token, call_id }`
- InputError:
  - `call_id` is invalid
  - The user is already muted
  - `token` is invalid
- AccessError
  - `call_id` is valid and user is not a member of the call
- Return Type `{}`

call/unmute/v1

- Description:

    - Given a token and a call_id, and that the user is already muted, the user's microphone is unmuted; meaning, anything the user says through their microphone can be heard by the other participants in the call.
- Method:
    - `POST`
- Parameters:
    - `{ token, call_id }`
- InputError:
    - `call_id` is invalid
    - The user is already unmuted
    - `token` is invalid
- AccessError
    - `call_id` is valid and user is not a member of the call
- Return Type `{}`

call/invite/v1

- Description:
    - Invites a user to join a voice call. The user does not have to be a member of the channel or dm the voice call is hosted in.
- Method:
    - `POST`
- Parameters:
    - `{ token, call_id, u_id }`
- InputError
    - `token` is invalid
    - `call_id` is invalid
    - `u_id` is invalid
- AccessError:
    - `call_id` is valid and user is not a member of the call
- ReturnType: `{}`

call/listmembers/v1

- Description:
    - Lists all the members currently present in a voice call.
- Method:
    - `GET`
- Parameters:

    - `{ token, call_id }`
- InputError
    - `token` is invalid
    - `call_id` is invalid
- AccessError
    - `call_id` is valid and user is not a member of the call
- ReturnType: `{ users }`

call/active/v1

- Description:
    - For a given `channel` or `dm_id`, returns whether there is a call active in it. `channel_id` is `-1` if they're checking the `dm_id`, and `dm_id` is -1 if the user is checking the channel_id.
- Method:
    - `GET`
- Parameters:
    - `{ token, channel_id, dm_id }`
- InputError:
    - Both `dm_id` and `channel_id` are invalid
    - Neither `dm_id` nor `channel_id` are `-1`
- `token` is invalid
- AccessError
    - The pair of channel_id and dm_id are valid (i.e. one is -1, the other is valid) and the authorised user has not joined the channel or DM they are checking
- ReturnType: `{ is_active }`

call/join/v1

- Description:
    - Given a call_id that the authorised user can join, adds them to the call.
- Method:
    - `POST`
- Parameters:
    - `{ token, call_id }`
- InputError:
    - `token` is invalid
    - `call_id` is invalid

  - User is already a member of the call
- AccessError:
  - `call_id` is valid but user is not a member of the channel/dm the call is hosted in
- ReturnType: `{}`

call/whiteboard/v1

- Description:
  - Given a call_id that the authorised user is a member of, start a collaborative whiteboard that all members can draw on.
- Method:
  - `POST`
- Parameters:
  - `{ token, call_id }`
- InputError:
  - `token` is invalid
  - `call_id` is invalid
  - There already exists a collaborative whiteboard
- AccessError:
  - `call_id` is valid but the user is not a member of the call
- Return Type: `{ whiteboard_id }`

call/sharescreen/v1

- Description:
  - Given a `call_id` that the authorised user is a member of and that the user is not currently sharing their screen, the user starts sharing their screen.
- Method:
  - `POST`
- Parameters:
  - `{ token, call_id }`
- InputError:
  - `token` is invalid
  - `call_id` is invalid
  - The user is already sharing their screen
- AccessError:
  - `call_id` is valid but the user is not a member of the call
- ReturnType: `{}`

call/stopsharescreen/v1

- Description:
  - Given a call_id that the authorised user is a member of and that the user is currently sharing their screen, the user stops sharing their screen.
- Method:
  - Post
- Parameters:
  - `{ token, call_id }`
- InputError:
  - `token` is invalid
  - `call_id` is invalid
  - The user is not currently sharing their screen
- AccessError:
  - `call_id` is valid but the user is not a member of the call
- ReturnType: `{}`

Call/end/v1

- Description:
  - Given a call_id, a member with owner permissions can end the call. All collaborative whiteboards are deleted and sharescreen is stopped.
- Method:
  - `POST`
- Parameters:
  - `{ token, call_id }`
- InputError:
  - `token` is invalid
  - `call_id` is invalid
- AccessError:
  - User is not a member of the channel/dm the call is hosted in
  - User does not have owner permissions in the channel/dm the call is hosted in
- ReturnType: `{}`

message/sendlatex/v1

- Description:
  - Given a string of raw LaTeX code, return a formatted string that can be displayed in html via MathMl
- Method:

    - `POST`
- Parameters:
    - `{ token, channel_id, message }`
- InputError:
    - length of message is less than 1 or over 1000 characters
    - Improper latex syntax
    - Invalid `channel_id`
    - Invalid `token`
- AccessError
    - `channel_id` is valid and the authorised user is not a member of the channel
- ReturnType:
    - { message_id }

message/sendfile/v1

- Description:
    - Send a file from the authorised user to the channel specified by `channel_id`. Note: Each message should have its own unique ID, i.e. no messages should share an ID with another message, even if that other message is in a different channel.
- Method:
    - `POST`
- Parameters:
    - `{ token, channel_id, file }`
- InputError:
    - File size > 25 mb
    - Invalid channel_id
    - Invalid token
- AccessError
    - `channel_id` is valid and the authorised user is not a member of the channel
- ReturnType:
    - `{ message_id }`

message/sendimage/v1

- Description:
    - Uploads an img_url then displays the image in the channel as a message from the authorised user to the channel specified by `channel_id`. Note: Each

message should have its own unique ID, i.e. no messages should share an ID with another message, even if that other message is in a different channel.

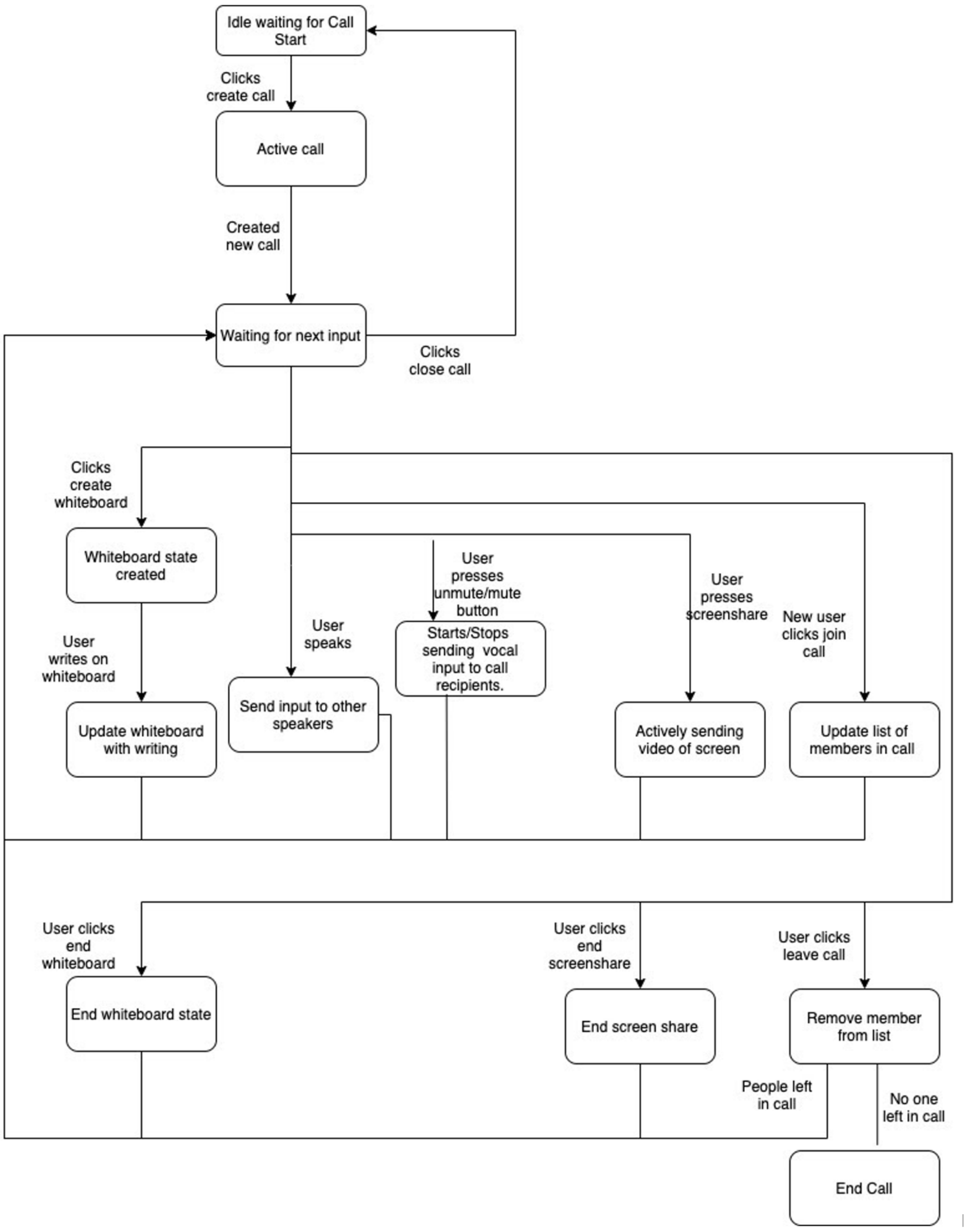
- Method:
  - `POST`
- Parameters:
  - `{ token, channel_id, img_url }`1
- InputError:
  - length of message is less than 1 or over 1000 characters
  - image uploaded is not a JPG
  - img_url returns an HTTP status other than 200
  - Invalid `channel_id`
  - Invalid `token`
- AccessError
  - `channel_id` is valid and the authorised user is not a member of the channel
- ReturnType:
  - `{ message_id }`

[Design] Conceptual Modelling (State)

> Now that you have a sense of the problem to solve, and what capabilities you will need to provide to solve it, add at least one state diagram to your PDF to show how the state of the application would change based on user actions. The aim of this diagram is how to a developer understand the different states the user or application.

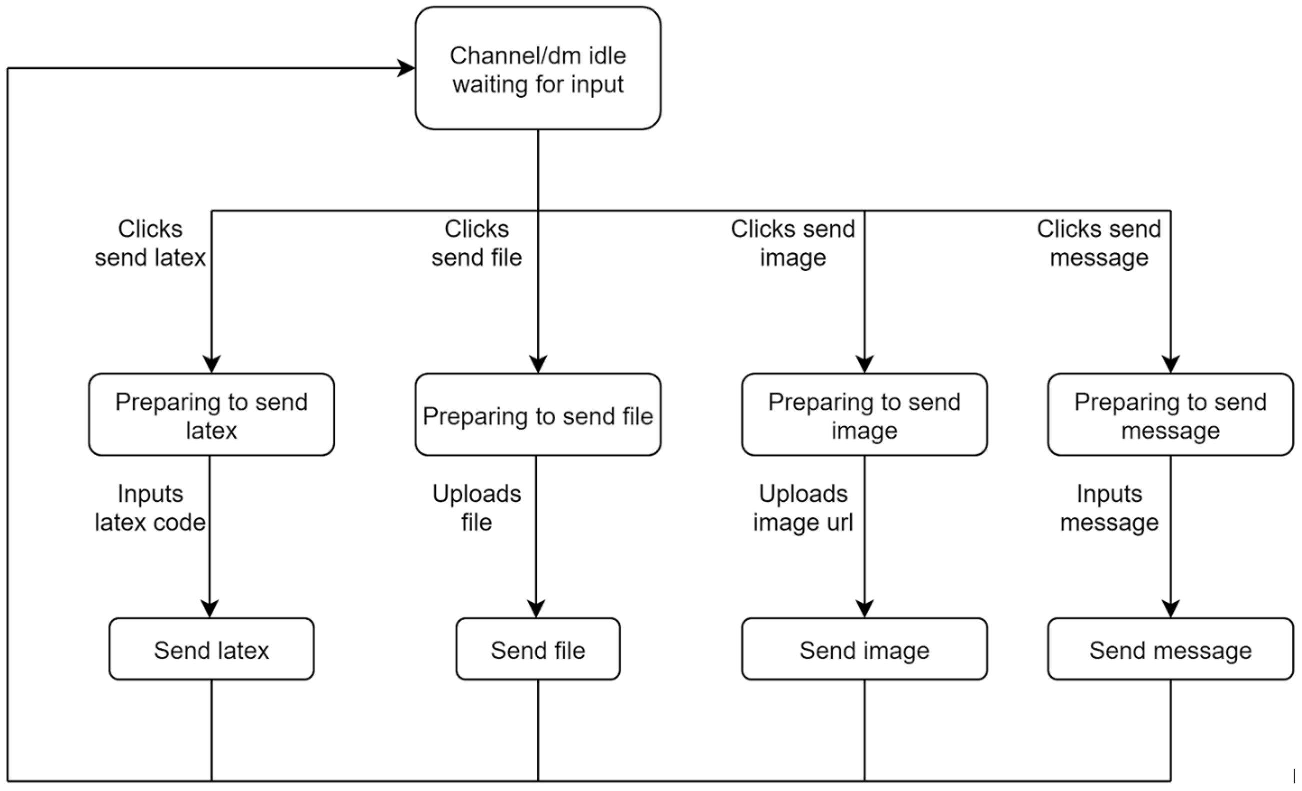
Call Feature State Diagram

LaTeX Feature State Diagram



Idle waiting for Call Start
Clicks create call
Active call
Created new call
Waiting for next input
Clicks close call
Clicks create whiteboard
Whiteboard state created
User writes on whiteboard
Update whiteboard with writing
User speaks
Send input to other speakers
User presses unmute/mute button
Starts/Stops sending vocal input to call recipients.
User presses screenshare
Actively sending video of screen
New user clicks join call
Update list of members in call
User clicks end whiteboard
End whiteboard state
User clicks end screenshare
End screen share
User clicks leave call
Remove member from list
People left in call
No one left in call
End Call

LaTeX Feature State Diagram


Channel/dm idle waiting for input
Clicks send latex
Clicks send file
Clicks send image
Clicks send message
Preparing to send latex
Preparing to send file
Preparing to send image
Preparing to send message
Inputs latex code
Uploads file
Uploads image url
Inputs message
Send latex
Send file
Send image
Send message