User requirements

Elicitation

Interview

Interviewees:
a) Danielle Wu
   i) Email: daniellewu@live.com.au
b) Victoria Chen
   i) Email: vicwombat15@gmail.com

Questions:
1) **What problems do you have when accessing Streams?**
   a) 'I can't access Streams from my phone, because the display is messed up when I use safari.'
   b) 'When I'm outside, I can't communicate with my team, since I can't use my laptop.'
2) **What problems do team communication tools have with regards to the user-friendliness?**
   a) 'Using Streams through a browser is pretty annoying since I need to remember the URL.'
   b) 'I don't really like the design of Streams, since it's pretty bland.'
3) **What problems do you face when trying to communicate in Streams?**
   a) 'I find it difficult to engage with my team members through just text messages and completing work together is really inefficient. I also can't have quick conversations about work since I only get notified if I'm in the app.
   b) 'It's hard to gauge whether teammates are actually paying attention when I'm messaging them.
4) **What problems do you face when receiving notifications?**
   a) 'Sometimes I get flooded with notifications from channels that I'm not really active in'
   b) 'I want to stop seeing notifications while I'm busy with other stuff.'
5) **What problems do you face when sending and receiving messages?**
   a) "It's hard to keep track of multiple messages in a channel and I can't send photos or videos."
   b) "I can't send pictures, videos or files, which makes explaining tasks really difficult. Also sometimes I have to send multiple messages because the character limit is too small."

Proposed Solutions

Problem	Proposed Solution
Not cross-platform	Streams should be run through both a Streams app (for mobile users) and a browser (for computer/PC users).
Not user friendly	
Can only send text messages	Messages should have attachments such as photos, videos, files and recordings.
Hard to keep track of opinions of the members especially in larger groups	Users should be able to generate polls in a channel or dm.
Cannot engage with team members	Users can start a voice/video call inside of a channel or dm.
	The user_profile function should have additional fields such as gender, age and pronouns
The interface is too bland	The UI of Streams should be customisable, adding the ability to change wallpapers and themes.
Receiving unwanted notifications	Users should be able to set their current status and individual mute channels which hide notifications from that channel
	Channel owners should be able to mute certain users within their channels.

Analysis & Specification - Use Cases

User Stories and Acceptance Criteria

User story	Acceptance criteria
As a **member of a team working on a task**, I want to **send documents such as text files, images and videos** so that **I can share my work with my teammates.**	- When a user sends a file, everyone else in the channel or dm is able to retrieve and download that file. - Given that the user is a member of a channel or dm, when they receive a file, the file size is no more than 8MB.
As a **member of multiple teams,** I want to **mute specific channels or dms** so that **I am not distracted when I am working on something else.**	- As a member of the given channel/dm, when I mute the channel/dm, I expect to be able to unmute the channel/dm. - As a member of the given channel/dm, when I mute the channel/dm and then unmute, I do not receive the notifications I missed while the channel/dm was muted.
As a **member of a team working on a task,** I want to **show my current status** so that I can **show my availability to other group members.**	- Has default status icons such as 'active', 'do not disturb', 'hidden'… - Can also have a custom status with a character limit.
As the **leader of a team,** I want to **be able to mute users in my channel/dm** so that **they do not disrupt the flow of communication.**	- As a member of a channel with owner permissions, I can mute other users. - Any member of a dm can mute other users. - Muted user is still a member of the channel/dm and can access everything except for sending messages.
As a **member of a team environment,** I want to **know how to address my team members** so that I can **work cohesively with my team.**	- Additional user details are optional, such as age and gender. - Can be hidden from the public if chosen.
As the **leader of a large group,** I want to be **able to start a poll** so that **I can easily collate the opinions of my team members.**	- Polls can be initiated by anyone - Polls can have multiple options - Polls can be closed by the owner or the user who initiated the poll - Polls can only be accessed by members of the channel/dm
As a **member of a team,** I want to **access this communication tool via an app** so that **the tool is more easily accessible on mobile devices and I can receive push notifications**	Access to the same or majority of function routes on mobile devices compared to desktop
As a **member of a team,** I want to **be able to hold a voice/video call** so that **we can communicate with each other in real-time.**	Given the member has started a video/voice call, when another user joins the call
As **a member of a team,** I want to **share my screen with my team** so that **we can work collaboratively on a single issue.**	Given the member shared his screen, when another user joined the call, then the system served a video stream of the first user's screen to the second user.

Use Case

Use Case:	Starting a voice call and sending a photo
Goal in Context	Users need to start a voice call on their mobile and send a photo, after adding members to a dm and eliminating potential distractions.
Scope	DM, Voice call, Muting, Status, Polls, Sending images
Level	Primary Task
Precondition	The user and their group members have existing accounts on Streams.
Success End Condition	The user starts the voice call, sends a photo and ends the voice call.
Failed End Conditions	The voice call fails to start or the photo fails to send.
Primary Actor	User
Trigger	User logs onto Streams

VOICE CALL SUCCESS SCENARIO:

STEP 1: System asks user for email and password.

STEP 2: User enters email and password and selects login.

STEP 3: System accepts the email and password and then generates and returns a unique token to the user.

STEP 4: User selects 'create dm' button.

STEP 5: System serves a drop down menu with a list of other users.

STEP 6: User selects other users they want to add to the DM and then selects the 'create' button.

STEP 7: System accepts a selection of members and then generates a new DM, which they add the selected members to.

STEP 8: User mutes notifications from other channels/DMs

STEP 9: System changes the settings of the user to ignore notifications from the selected channels/DMs

STEP 10: User selects 'set status' button and selects the 'do not disturb' option.

STEP 11: System sets user's status to 'do not disturb'.

STEP 12: User selects the 'make poll' button, and enters 'Is everyone ok to have a voice call right now?' as the query with 'yes' or 'no' as the possible responses.

STEP 13: System sends a poll to the message stream in the DM.

STEP 14: All users select 'yes'.

STEP 15: System accepts poll responses and displays results to poll creators.

STEP 16: User selects 'start a voice call' button.

STEP 17: System begins the call and informs other users of an ongoing voice call.

STEP 18: Users select 'join voice call' button.

STEP 19: User selects 'leave voice call' button.

STEP 20: User selects 'upload photo' and selects a JPG image of their math homework.

STEP 21: System receives image, and stores it with a URL.

STEP 22: System posts the URL to the DM and formats the URL into a JPEG image.

STEP 23: All users select 'leave voice call' button.

STEP 24: System ends the call.

STEP 25: All users select 'Logout' from the top-right corner.

STEP 26: System serves a logged out screen and invalidates all user tokens.

Validation

After inquiring with each interviewee, the general reception was that our use cases adequately addressed the problems they had raised with our current system. To sum each interviewee's finishing comments:

a) "Yeah I think it's good overall, since it covers most of the problems that I thought of."
   i) "The ability to send photos, videos, files and recordings is especially nice and having push notifications is a good way to notify me when other people send me messages."
   ii) "Adding a voice call feature is also super helpful when holding meetings online, since you can tell if people are actually paying attention or not."
b) "It deals with most of the issues I had previously"
   i) "The muting notifications feature resolves my issue with getting distracted by notifications when I'm concentrating on a task."
   ii) "Adding a polling feature to Streams is actually a great way to receive feedback from my team members."
   iii) "The use case acknowledges my problem with not being able to send photos and additional types of files."

Design

Input/Output Types

has suffix **_url**	string
has suffix **_id**	integer
has suffix **_message**	string
has suffix **_code**	string
status_code	string
vote_option	string

poll_option	Dictionary of shape {
	     title: 'title'
	     users: [u_ids]
	 }

vote_options	List of poll_options, where each poll option contains the titles and users which have voted.
votes	Dictionary of shape {vote_option: votes_received} where each Key is the vote option, and the Key Value is the votes received for that vote option as a list of u_ids.
poll_details	Dictionary of shape {poll_name, votes, is_closed, creator_id}
sound_clip	Mp3 clip (10 milliseconds long)
sound_clips	List of sound_clips, where each sound clip is taken from the users currently in the call.

Interface Design

user/profile/v2 For a valid user, returns information about their user_id, email, first name, last name, handle, img_url, gender, age, pronouns.	GET	**Parameters:** { token } **Return Type:** { users }	**InputError** when: ● u_id does not refer to a valid user
message/send/photo/v1 Send a photo from the authorised user to the channel specified by channel_id. Note: Each photo should have its own unique ID, i.e. no photo should share an ID with another photo, even if that other photo is in a different channel.	POST	**Parameters:** { token, channel_id, img_url } **Return Type:** { photo_id }	**InputError** when: ● channel_id does not refer to a valid channel ● img_url returns an HTTP status other than 200 **AccessError** when: ● channel_id is valid and the authorised user is not a member of the channel
message/send/video/v1 Send a video from the authorised user to the channel specified by channel_id. Note: Each video should have its own unique ID, i.e. no video should share an ID with another video , even if that other video is in a different channel.	POST	**Parameters:** { token, channel_id, video_url } **Return Type:** { video_id }	**InputError** when: ● channel_id does not refer to a valid channel ● video_url returns an HTTP status other than 200 **AccessError** when: ● channel_id is valid and the authorised user is not a member of the channel
message/send/file/v1 Send a file from the authorised user to the channel specified by channel_id. Note: Each file should have its own unique ID, i.e. no file should share an ID with another file , even if that other file is in a different channel.	POST	**Parameters:** { token, channel_id, file_url } **Return Type:** { file_id }	**InputError** when: ● channel_id does not refer to a valid channel ● file_url returns an HTTP status other than 200 **AccessError** when: ● channel_id is valid and the authorised user is not a member of the channel

message/send/recording/v1 Send a recording from the authorised user to the channel specified by channel_id. Note: Each recording should have its own unique ID, i.e. no recording should share an ID with another recording , even if that other recording is in a different channel.	POST	**Parameters:** { token, channel_id, recording_url } **Return Type:** { recording_id }	**InputError** when: • channel_id does not refer to a valid channel • recording_url returns an HTTP status other than 200 **AccessError** when: • channel_id is valid and the authorised user is not a member of the channel
message/senddm/photo/v1 Send a photo from the authorised user to the channel specified by dm_id. Note: Each photo should have its own unique ID, i.e. no photo should share an ID with another photo, even if that other photo is in a different channel.	POST	**Parameters:** { token, dm_id, img_url } **Return Type:** { photo_id }	**InputError** when: • channel_id does not refer to a valid channel • img_url returns an HTTP status other than 200 **AccessError** when: • dm_id is valid and the authorised user is not a member of the channel
message/senddm/video/v1 Send a video from the authorised user to the channel specified by dm_id. Note: Each video should have its own unique ID, i.e. no video should share an ID with another video , even if that other video is in a different channel.	POST	**Parameters:** { token, dm_id, video_url } **Return Type:** { video_id }	**InputError** when: • dm_id does not refer to a valid channel • video_url returns an HTTP status other than 200 **AccessError** when: • dm_id is valid and the authorised user is not a member of the channel
message/senddm/file/v1 Send a file from the authorised user to the channel specified by dm_id. Note: Each file should have its own unique ID, i.e. no file should share an ID with another file , even if that other file is in a different channel.	POST	**Parameters:** { token, dm_id, file_url } **Return Type:** { file_id }	**InputError** when: • dm_id does not refer to a valid channel • file_url returns an HTTP status other than 200 **AccessError** when: • dm_id is valid and the authorised user is not a member of the channel

message/senddm/recording/v1 Send a recording from the authorised user to the dm specified by dm_id. Note: Each recording should have its own unique ID, i.e. no recording should share an ID with another recording , even if that other recording is in a different channel.	POST	**Parameters:** { token, dm_id, recording_url } **Return Type:** { recording_id }	**InputError** when: - dm_id does not refer to a valid channel - recording_url returns an HTTP status other than 200 **AccessError** when: - dm_id is valid and the authorised user is not a member of the channel
channel/mute/v1 Given a channel with ID channel_id that the authorised user is a member of, toggle the notifications from that channel on or off. I.e. When notifications/get is called no notification related to the muted channel, should be returned. Note: Muted notifications should be callable once the user unmutes the channel.	POST	**Parameters:** { token, channel_id } **Return Type:** {}	**InputError** when: - channel_id does not refer to a valid channel **AccessError** when: - channel_id is valid and the authorised user is not a member of the channel
dm/mute/v1 Given a dm with ID dm_id that the authorised user is a member of, toggle the notifications from that dm on or off. I.e. When notifications/get is called no notification related to the dm, should be returned. Note: Muted notifications should be callable once the user unmutes the dm.	POST	**Parameters:** { token, dm_id } **Return Type:** {}	**InputError** when: - dm_id does not refer to a valid dm **AccessError** when: - dm_id is valid and the authorised user is not a member of the dm
notifications/mute/v1 Toggle notifications on or off. When notifications/get is called no notification should be returned. Note: Muted notifications should be callable once the user unmutes their notifications	POST	**Parameters:** { token } **Return Type:** {}	N/A
user/status/change/v1 Given a valid status_code, change the status of the currently logged in user to the new status.	PUT	**Parameters:** { token, status_code } **Return Type:** {}	**InputError** when: status_code does refer to a valid status
user/status/get/v1 Given a u_id, return the current status of the user.	GET	**Parameters:** { token, u_id } **Return Type:** {}	**InputError** when: u_id does not refer to a valid user

voicecall/start/v1 Given a channel_id or dm_id, the user associated with the token initiates a new voice call in the channel/dm. channel_id is the channel that the voice call is being started in, and is -1 if it is being started in a DM. dm_id is the DM that the voice call is being started in, and is -1 if it is being started in a channel. The user who created the voice call automatically joins the voice call.	POST	**Parameters:** { token, channel_id, dm_id} **Return Type:** {call_id}	**InputError** when: • dm_id and channel_id are both -1 • Either dm_id or channel_id are not -1 • Invalid channel_id or dm_id **AccessError** when: • The user is not a member of the given channel/dm
voicecall/join/v1 Given a call_id, the user associated with the given token joins the active voice call.	POST	**Parameters:** { token, call_id} **Return Type:** {}	**InputError** when: • call_id does not refer to a valid call **AccessError** when: • The user associated with token is not a member of the channel/dm which the voice call is in
voicecall/leave/v1 Given a call_id, the user associated with the given token leaves the voice call. If the only user left in the voice call leaves, the voice call ends.	POST	**Parameters:** { token, call_id} **Return Type:** {}	**InputError** when: • call_id does not refer to a valid call **AccessError** when: • The user associated with token is not a member of the channel/dm which the voice call is in
voicecall/send_sound/v1 Sends a sound clip from the authorized user to the voice call at the given call_id.	POST	**Parameters:** { token, call_id, sound_clip} **Return Type:** {}	**InputError** when: • User is not part of the voice call • call_id does not refer to a valid voice call **AccessError** when: • The user associated with token is not a member of the channel/dm which the voice call is in
voicecall/recieve_sound/v1 Gets a collection of sound_clips from the voice call with the given call_id. This should return a collection of sound clips from each user currently in the call.	GET	**Parameters:** { token, call_id} **Return Type:** {sound_clips}	**InputError** when: • User is not part of call • call_id does not refer to a valid voice call **AccessError** when: • The user associated with

			token is not a member of the channel/dm which the voice call is in
poll/create/v1 Creates a new poll with the given poll name and vote options. channel_id is the channel that the poll is being started in, and is -1 if it is being started in a DM. dm_id is the DM that the poll is being started in, and is -1 if it is being started in a channel.	POST	**Parameters:** { token, channel_id, dm_id, poll_name, vote_options} **Return Type:** {poll_id}	**InputError** when: ● dm_id and channel_id are both -1 ● Either dm_id or channel_id are not -1 ● Invalid channel_id or dm_id **AccessError** when: ● The user does not have owner permissions in the channel (does not apply to dm)
poll/vote/v1 Add a user's vote to the vote option of the poll with the given poll_id. The user should not be able to vote more than once.	PUT	**Parameters:** { token, poll_id, vote_option} **Return Type:** {}	**InputError** when: ● invalid poll_id ● vote option is not available **AccessError** when: ● valid poll_id, but user is not a member of the channel/dm in which the poll is in
poll/unvote/v1 Removes a user's vote from the vote option of the poll with the given poll_id. The user should not be able to remove more than a single vote.	PUT	**Parameters:** { token, poll_id, vote_option} **Return Type:** {}	**InputError** when: ● Invalid poll_id ● User has not voted for this poll **AccessError** when: ● valid poll_id, but user is not a member of the channel/dm in which the poll is in
poll/details/v1 Given a poll with ID poll_id, provide basic details about the poll.	GET	**Parameters:** { token, poll_id} **Return Type:** {poll_details}	**InputError** when: ● Invalid poll_id **AccessError** when: ● Valid poll_id, but user is not a member of the channel/dm in which the poll is in

poll/close/v1 Closes the poll associated with poll_id. The results of the poll are then sent as a single message to the channel by the poll creator.	DELETE	**Parameters:** { token, poll_id} **Return Type:** {}	**InputError** when: • Invalid poll_id **AccessError** when: • The user does not have owner permissions in the channel (does not apply to dm)
notifications/receive/v1 Given a push_token, authenticate the user corresponding to the token to receive notifications from their device.	PUT	**Parameters:** {token, push_code, device_id} **Return Type:** {}	**InputError** when: • Device_id does not correspond to any user. • Invalid push_code **AccessError** when: • The user associated with token is not logged in or does not exist
notifications/push/v1 Given a notification_message, push it to the device corresponding to the device_id.	POST	**Parameters:** {token, notification_message, device_id} **Return Type:** {}	**InputError** when: • Notification_message is empty • Device_id does not correspond to any user. **AccessError** when: • The user associated with token is not logged in or does not exist

Errors for all functions

Either an InputError or AccessError is thrown when something goes wrong. All of these cases are listed in the Interface table. If input implies that both errors should be thrown, throw an AccessError.

One exception is that, even though it's not listed in the table, for all functions except auth/register, auth/login, auth/password_reset/request and auth/password_reset/reset, an AccessError is thrown when the token passed in is invalid.

Conceptual Modelling

"Enter email and password"

User enters invalid email or password

User logs out

User logs in

"Application Homepage"

User selects "change status"

User selects a new status

"Choose status or create a custom status"

User unmutes notification

User mutes notification

"Muted Notifications"

User chooses "create dm"

"Select the users to create a new dm"

"Displaying poll results"

User chooses dm members

User goes to home

User closes poll

All users selects a poll option.

"Waiting for poll votes"

User leaves voice call

All users leave voice call

"Voice call active"

User starts a voice call

"Displaying Dm homepage"

User enters a query and poll option

User selects "Make Poll"

User mutes notification

User unmutes notification

User joins voice call

"Muted given DM's Notifications"

"Waiting for poll details"