Planning

Elicitation

Interviewee #1:

Name: Luke Macauley
Email: lukemacauley99@gmail.com

Interviewee #2:

Name: Alex Cain
Email: alexcain904@gmail.com

Alex and Luke are both university students who use team messaging tools such as slack for their group assignments. Alex is also a manager who has regularly used such tools during the COVID isolation period.They do not have a particular preference for one platform, and would be more than open to the idea of using Flockr.

How much do you currently use a teamwork-driven communication tool?

Luke (L): Depends on how many group assignments I have, about three times a week during the week before an assignment would be due.

Alex (A): I use Microsoft Teams for every shift at work, which is roughly four days a week, and this doesn't include the times for group university projects.

What are some good things about platforms such as Flockr that should be maintained in the future?

L: I like that it is simple and straightforward. The problem with a lot of team applications is that it can be too complicated for a new user to become used to it, and as a result, may never get in the habit of using it.

A: Similar to Luke - some team messaging tools are hard to navigate around, and can be frustrating especially when you want to do something simple. I think a good idea in the future is to add functionality without overcomplicating the original interface

What are some limitations of all or most team-messaging platforms?

L: Sometimes I'd like to be able to embed certain videos in the group chat. It would be easier to simply send a video to the team that can be opened on the main channel, so that people can discuss the video in real time, instead of watching it on another desktop and coming back to the channel afterwards.

A: I think an inconsistency across the different platforms is perhaps the most annoying part. A reason why I use a variety of different tools is that I simply don't get all the functionality I require from any singular tool. For example, tools such as slack or flockr are great for simple messaging, however I find I often resort to going on Discord for group calls and screen-sharing. I would prefer everything I need to be in one place.

In regards to Flockr specifically, what are some of the shortcomings of team-working functionality?

L: Definitely the ability to do group calls and screen-sharing. The simple messaging is great, however, sometimes people just need to talk face-to-face, or be able to see what someone else is doing on a computer - it's just not possible to be able to explain everything effectively over text.

A: A frequent topic of conversation in group assignments is mainly what needs to get done and when. Apps like Flockr can only allow for discussions and regardless of how easy it is to communicate within the app, sometimes team objectives can get lost in all the chatter. It would be nice if there was a way to more clearly define what needs to be done.

What would make you use Flockr more?

L: I would definitely use Flockr more if I would be able to use it, or even just receive notifications, from my phone. Although it doesn't sound like much, having to be on your computer, and log-in everytime just to use it is somewhat of a deterrent. Like most people, I carry my phone everywhere and would be far more connected with my team members if it was available as an app on my phone.

A: If there was some way to integrate a group "to-do" list, I feel as if that would significantly improve the experience. Then there would be less wasted discussion, and everyone could visualise what needs to be done in a much clearer way.

Analysis & Specification - Use Cases

1. "As a university student, I want to be able to create a to-do list that is shared amongst multiple people so that I would be able to set team-oriented goals and work clearly for course projects and assignments."

   User Acceptance Criteria:

   Scenario: Assignment is due soon
   Given: The user navigates to the project channel
   When: The user selected the to-do list option
   And: Added multiple assignment objectives to the channel to-do list for all team members, including the user.
   Then: The system adds the to-do list to all channel members

2. "As people who are involved in a lot of team-based projects, we would prefer it if there were some sort of voice communication software. It would definitely make it easier to get in contact with our groupmates and allow for weekly voice meetings that are generally more effective than the standard texting."

   User Acceptance Criteria:

   Scenario: Weekly meetings
   Given: The user and their group mates have created a channel together
   When: The user selects the call button
   And: The user has allowed their microphones to be used by flockr
   Then: The system will connect their channel through a voice call
   And: Allows input from one's microphone to be heard from other's speakers.

3. “As manager, I want to be able to share my screen during a call and also for my employees to be able to share their screen so I can help them with issues that they run into on the job.”

   User Acceptance Criteria:

   Scenario: Employee needs assistance using company software
   Given: The manager is on a call with the employee
   When: The employee selects the ”share screen” option
   Then: The manager is able to see what the employee sees from their screen
   And: Communicate what needs to be done over the call.

4. “As someone who takes their phone everywhere, it would be great if Flockr were available on the go. As I am not always on a computer 24/7, updates will be more frequently accessible for me as long as some sort of notification was sent to my phone, telling me I have unread messages.”

   User Acceptance Criteria:

   Scenario: No access to computer, require communication with group members
   Given: The user has downloaded the mobile application for Flockr on their phone
   When: The user opens the application
   Then: They have access to most of the functionality of Flockr, i.e. sending and receiving messages, video calls, and are up to date with the current discussions, from their smartphone.

Use Case: Calling

Goal in context: Users need to call all channel members together.
Scope: Flockr
Level: Primary task
Preconditions: The user has access to the internet, has an existing account and is in a channel with groupmates.
Success end condition: The user and his/her channel members are able to conduct a group call together
Failed end condition: The user is unable to call their channel members
Primary actor: User initiating the call
Trigger: User presses “call” button from within a channel

Main success scenario

Step 1: User presses “call” button from within a channel
Step 2: System adds user into a “call” in current channel, where their microphones are active
Step 3: Other member(s) receive a notification saying they have an incoming call
Step 4: Other member(s) accept the call
Step 5: All members who accept the call are able to hear and talk to other members in the call.

Use Case: Group To-Do List

Goal in context: Users need a place to store tasks delegated to others

Scope: Flockr
Level: Primary task
Preconditions: The user has access to the internet, has an existing account and is in a channel with groupmates
Success end condition: The users are able to keep track of existing tasks that need to be done, tasks that are currently being solved, and tasks which are already done in their respective channels
Failed end condition: The channel members do not have access to the to-do list, and are unable to view or edit it
Primary actor: User adding to the to-do list
Trigger: User types in a task and optionally adds a person to delegate it to.

Main success scenario

Step 1: User presses “to-do list” button from within a channel
Step 2: System asks user what action they wish to take
Step 3: User asks to add objectives to the to-do list
Step 4: System asks which members in the channel the objectives should be assigned to
Step 5: User adds whichever members they wish to assign the task to
Step 6: System appends objectives to existing channel to-do list

Validation

Use Case: Call

Alex: Yes, this is how a basic call function should work.
Luke: This would definitely encourage me to use it more. However, it would be nice if I could mute my microphone if I was in a noisy place though.

Use Case: To-do list

Alex: This implementation of the to-do list would definitely allow for more concrete planning and visualisation of goals, as well as reduce the need to use another application. I.e. Flockr would allow me to have everything I need in one place.
Luke: Looks good to me, this is exactly how I wanted the to-do list to be implemented. Nice!

Interface Design

Use Case: To-do list

Function Name	HTTP Method	Parameters	Return Type	Exceptions	Description
channel/todolist/details	GET	(token, channel_id)	{**tasks**}	**InputError** when any of: - channel_id does not refer to a valid channel. **AccessError** when any of: - the authorised user is not already a member of the channel.	Given a Channel with ID channel_id that the authorised user is part of, provide basic details about their to-do list.
channel/todolist/clear	DELETE	(token, channel_id)	{}	**InputError** when any of: - channel_id does not refer to a valid channel. **AccessError** when any of: - when the authorised user is not an owner of this channel	Given a Channel with ID channel_id that the authorised user is an owner of, resets the internal data of the channel's to-do list to its initial state
task/add	POST	(token, channel_id, task_message)	{task_id}	**InputError** when any of: - Message is more than 1000 characters - channel_id does not refer to a valid channel. **AccessError** when any of: - the authorised user has not joined the channel they are trying to add tasks to	Add a task from the authorised_user to the channel specified by channel_id. The task is delegated to the caller of the function, and set to 'todo' status.
task/remove	DELETE	(token, task_id)	{}	**InputError** when any of: -Task (based on ID) no longer exists **AccessError** when none of the following are true: -Task with task_id was delegated to the authorised user making this request -The authorised user is an owner of this channel or the flockr	Given a task_id for a task, this task is removed from the channel's to-do list, assuming that the caller is the delegated user.
task/delegate	POST	(token, task_id, u_id)	{}	**InputError** when any of: -Task (based on ID) no longer exists - u_id does not refer to a valid user. **AccessError** when none of the following are true: -Task with task_id was delegated to the authorised user making this request -The authorised user is an owner of this channel or the flockr	Given a task with ID task_id that the authorised user is delegated to, delegate it to the user with u_id

task/edit	POST	(token, task_id, task_message)	{}	**InputError** when any of: - Message is more than 1000 characters -Task (based on ID) no longer exists **AccessError** when none of the following are true: -Task with task_id was delegated to the authorised user making this request -The authorised user is an owner of this channel or the flockr	Given a task with ID task_id that the authorised user is delegated to, update it's text with new text.
task/move	POST	(token, task_id, task_status)	{}	**InputError** when any of: -Task (based on ID) no longer exists - task_status does not refer to a valid value task_status **AccessError** when none of the following are true: -Task with task_id was delegated to the authorised user making this request -The authorised user is an owner of this channel or the flockr	Given a task with ID task_id that the authorised user is delegated to, update its task_status to one of either: 0 for to-do 1 for doing 2 for done
task/details	GET	(token, task_id)	{**task**}	**InputError** when any of: -Task (based on ID) no longer exists **AccessError** when any of: - the authorised user is not already a member of the channel.	Given a task with ID task_id in the channel which the authorised user is from, provide basic details about the task.

(outputs only) name ends in **task** - Dictionary containing task_id, channel_id, u_id_delegated, task_status and task_message
(outputs only) name ends in **tasks** - List of dictionaries containing task_id, channel_id, u_id_delegated, task_status and task_message

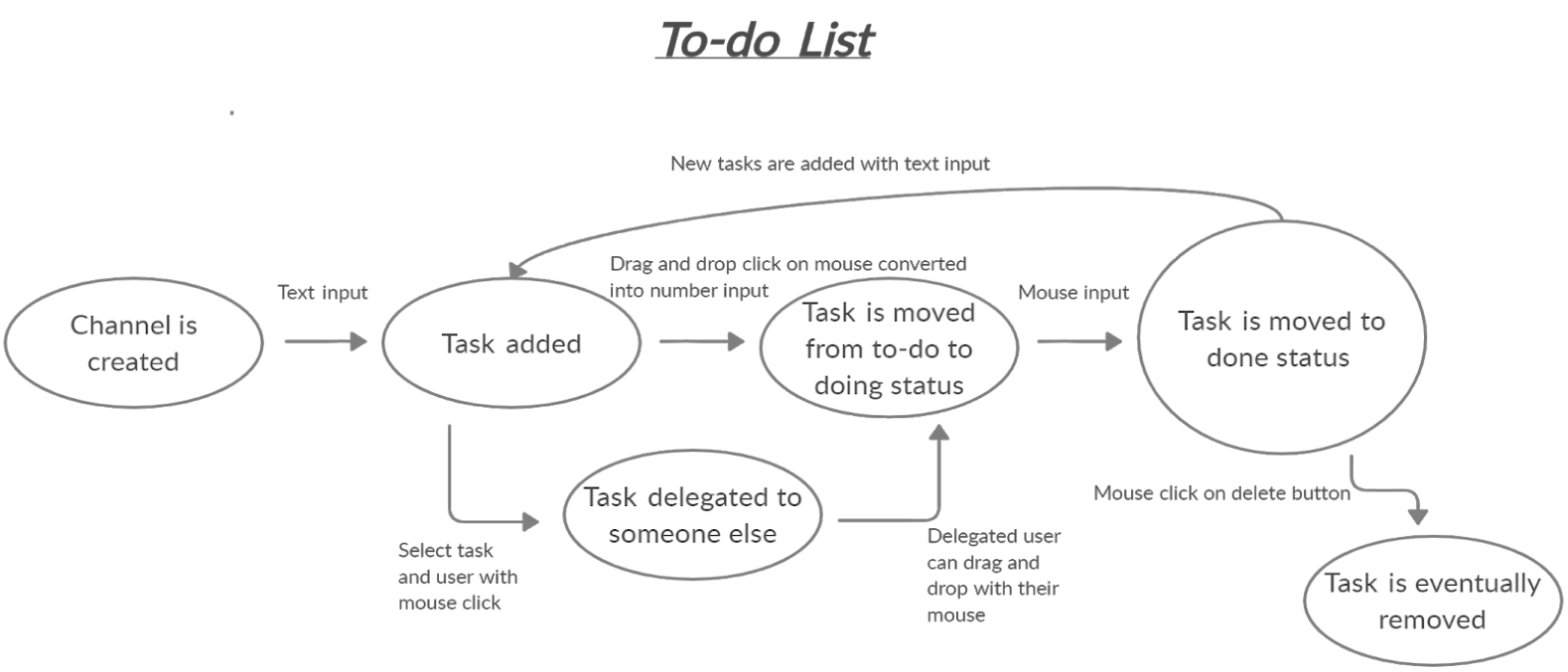
Use Case: Call

Function Name	HTTP Method	Parameters	Return Type	Exceptions	Description
channel/call/create	POST	(token, channel_id)	{call_id}	**InputError** when any of: -channel_id does not refer to a valid channel -u_id does not refer to a valid user **AccessError** when any of: -the authorised user is not already a member of the channel	Creates a call in a channel with ID channel_id. Once created the user with the given token is added to the call immediately, and all other members are invited to join the call. A unique call_id generated for the call is returned.
call/mute	POST	(token, call_id)	{}	**InputError** when any of: -call_id does not refer to a valid call **AccessError** when any of: -the user is not part of the call with call_id	Mutes microphone of a user (with given token). The user becomes muted.
call/unmute	POST	(token, call_id)	{}	**InputError** when any of: -call_id does not refer to a valid call **AccessError** when any of: -the user is not part of the call with call_id	Unmutes microphone of a user (with given token). The user becomes unmuted.
call/invite	POST	(token, call_id, u_id)	{}	**InputError** when any of: -u_id does not refer to a valid user -call_id does not refer to a valid call **AccessError** when any of: -the authorised user is not already a member of the channel	Invites a user (with user u_id) to join the call with ID call_id. Once invited the user is added to the call immediately with their microphones muted.
call/join	POST	(token, call_id)	{}	**InputError** when any of: -call_id does not refer to a valid call **AccessError**r when any of: -the authorised user is not already a member of the channel that the call with call_id originated from	Given the call_id of a call that originated from a channel that the user is already a part of, adds them to that call.
call/leave	POST	(token, call_id)	{}	**InputError** when any of: -call_id does not refer to a valid call **AccessError** when any of: -user with token is not part of the call with call_id	Given a call ID, the user is removed as a member of this call. (no longer part of this call)
call/listall_members	GET	(token, call_id)	{**call_members**}	**InputError** when any of: -call_id does not refer to a valid call **AccessError** when any of: -user with token is not part	Provide a list of all members (and their associated details) in the call with call_id.

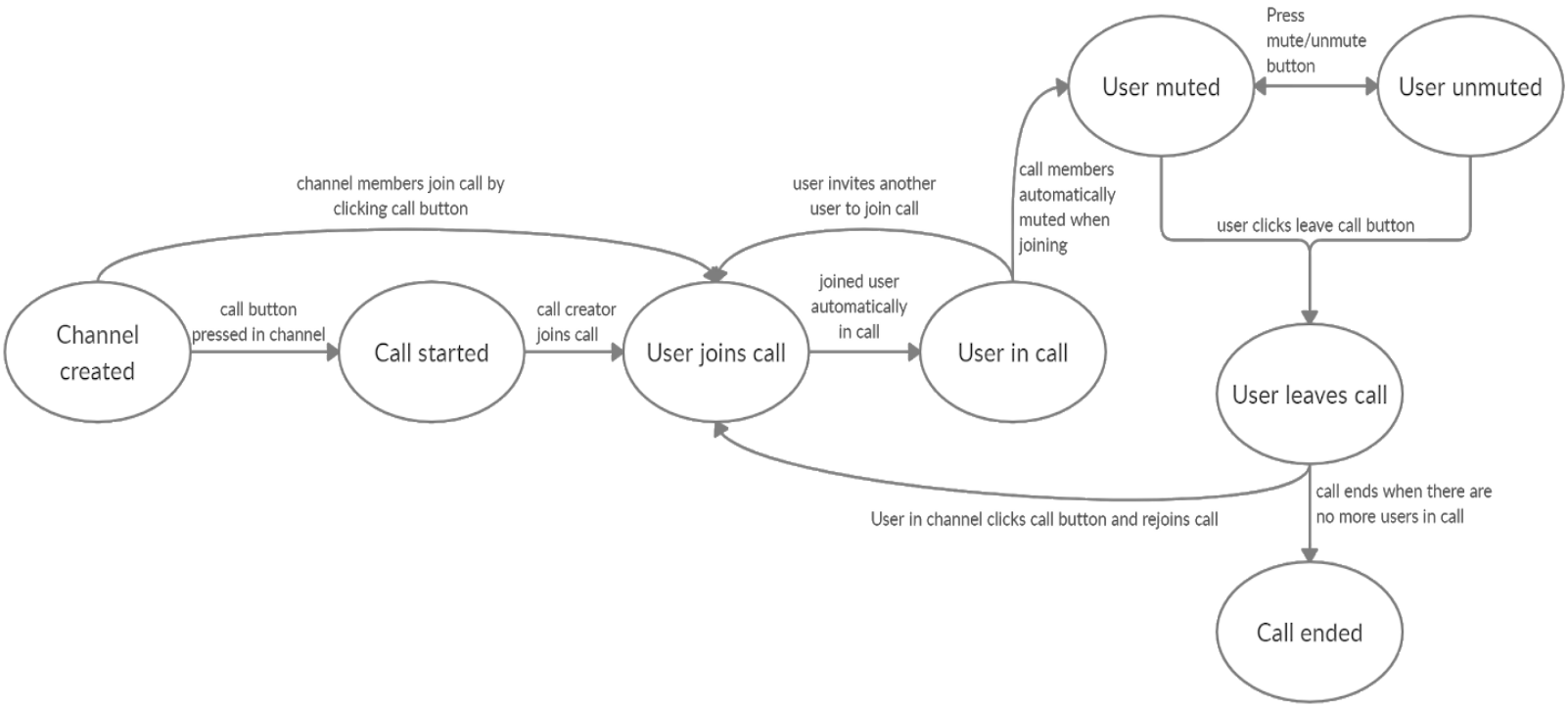
				of the call with call_id	
call/end	DELETE	(token, call_id)	{}	**InputError** when any of: -call_id does not refer to a valid call -call_id has 1 or more members	Given a call_ID with no existing members, ends the call and deletes the call_id.
microphone/add	POST	(token, mic_name)	{}	**InputError** when any of: -mic_name does not belong to a valid existing microphone on the device	Given mic_name, searches device for audio input device(with name mic_name) and connects it to user (with token)
microphone/remove	POST	(token)	{is_success}	N/A	Given an active token, checks if the user has any existing microphone connected. If the user does have an existing microphone and the microphone is no longer linked to the user, it returns true. Otherwise, it returns false.

(outputs only) **call_members** - List of dictionaries where each dictionary contains types { u_id, name_first, name_last, profile_img_url }

Conceptual Modelling



Call


Channel created
call button pressed in channel
Call started
call creator joins call
User joins call
joined user automatically in call
User in call
channel members join call by clicking call button
user invites another user to join call
call members automatically muted when joining
User muted
Press mute/unmute button
User unmuted
user clicks leave call button
User leaves call
call ends when there are no more users in call
Call ended
User in channel clicks call button and rejoins call