Elicitation

1. What processes would you undertake in order to send
   files to each other and keep them updated constantly between all users?

   (1): Have a database (externally) which everyone has access to or which select members of the group have access to. A role is delegated to updating and archiving old documents, In the event of a conference call the presenter presenting the material must be able to access it from the database to ensure the material is the most up to date and current.

   (2): Google docs to keep the collaboration updated

2. How many people would you consider as too many users in one channel if there is no limit?

   (1): Depends on the actual applications, e.g. certain programs cant handle more than 3 - 4 participants before losing either audio or video connection or both. We have had unit video conferencing to potentially 60 staff because of the widespread range of staff in regional areas. You would have a designated person to check the list of those who should be there and does a roll call for each person attending at the beginning of the conference call. After any question for feedback that person asks only one person to speak and moves on to the next person. This is a very laborious process with 60 odd staff but is the fairest process to ensure that everybody's voice is heard. If you want them to talk via video conference, then not ideal but if its participation through polling then that is possible to have as many as you like.

   (2): Depends on the meeting, if there is one or two people presenting that fine for unlimited. But if it's a project team where everyone needs to contribute, 6 - 8 is a good amount.

3. In what way would you deal with many users requesting to join multiple private channels that they have not been invited too for a specific task/team they have been assigned to?

   (1): Contacting the lead of each channel to provide access to those particular individuals, if it was only for a substantial period of time (limited access). If it was a on off or a short time period I would just get them to phone in.

(2): I would email all the users with all the links that have a description about what each task meeting is about. Relying on people to know what task (link) to join.

(1)
Izabela Pomes
izabela.pomes@health.nsw.gov.au

(2)
Andrew Pomes
apomes115@bigpond.com

Use Cases w/ User Acceptance Criteria

1. As the owner of the channel, I want to be able to always have permission to updating a shared database or document during and away from a conference call.

- A designated shared database file add button is implemented within the channel
- Document / Spreadsheet / Slide / etc optional choices
- A settings button with a dropdown on selecting who has permission to either
- view or edit the document
- Settings are saved as is until changed

2. As a channel owner, I should have the ability to control how many users can join a channel/conference call for a specific task that requires x amount of users/participants.

- Channel settings should contain a integer input for the max
- number of participants that can join the channel/call.
- Storing presets for specific room sizes in relation to a
- certain task

3a. As a role coordinator, I would like to have the ability to send a batch of links to all users with a description of each role
3b. As a role coordinator, I would like to have the ability to privately message each user with a link to a specific role that they are assigned to.

- A feature on the user's profile links to a page with checkboxes
- in a table with role description and channel link
- A select all users button, or exclusive selection

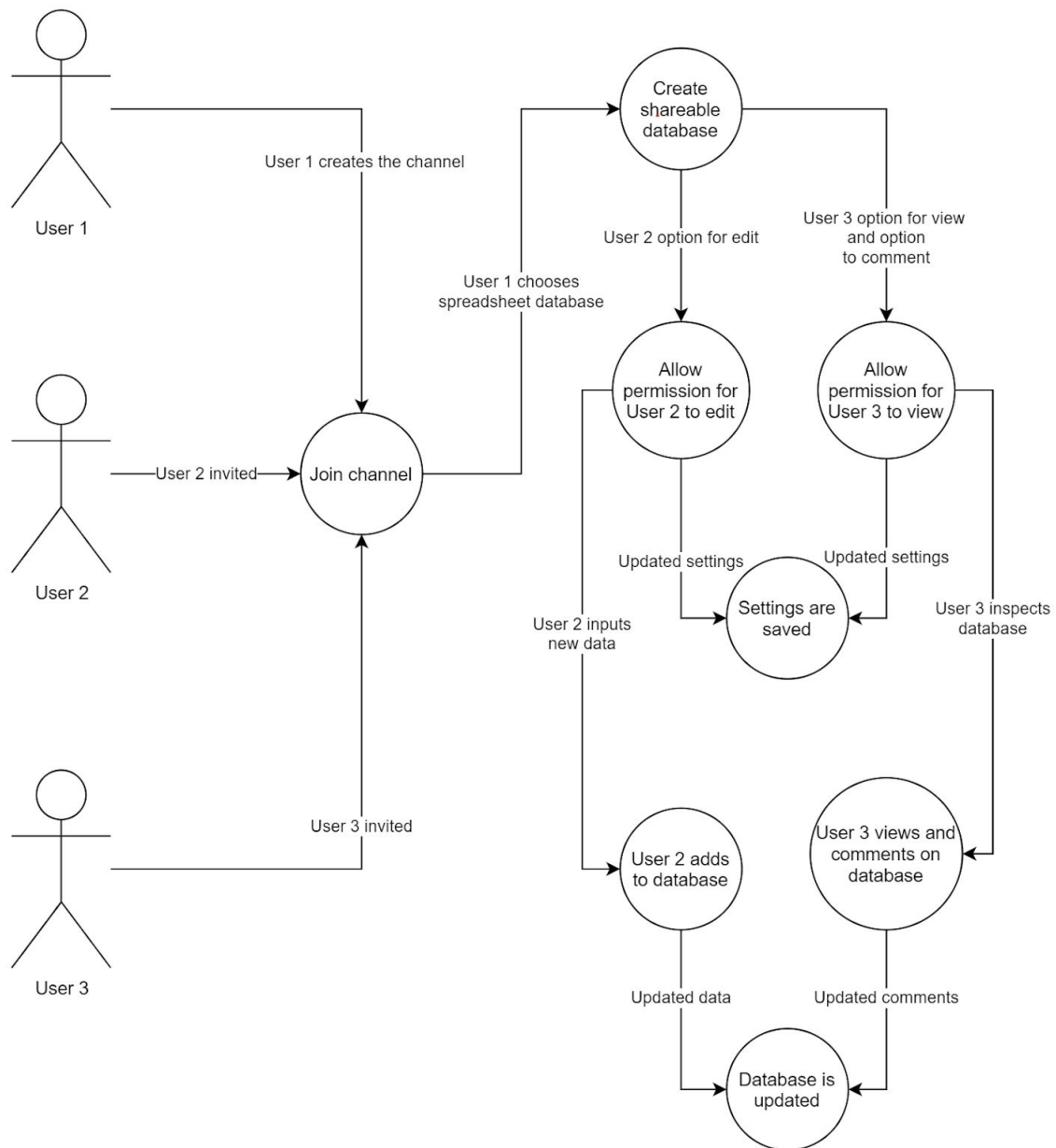● Choose between private or public sharing of link to user(s)

Figure A: Use Case Diagram for scenario 1

Validation

(1) *1. Very good idea*

    2. That would work very well

*3. Sounds like a lot of prework to set this up. May not work
with limited time but a longer lead in would work well.*

*(2) 1. Link to the document is saved in the channel which is adequate. The link is
emailed to all designated users of that channel.*

*2. Only should be used for a very public channel. Should have
a matrix of all channels and their user access to channels.
Shouldn't be worried about maximum users, more about who has
access to the channels.*

*3. Agree with this use case setup. Good way to secure access to
channel links and documents.*

Interface Design

Function Name	HTTP Method	Parameters	Return type	Exceptions
channel/database/create	POST	(token, channel_id)	{dbase_id}	**InputError** when any of: - channel_id does not refer to a valid channel. **AccessError** when any of: - token is not admin
channel/database/add	POST	(token, channel_id, dbase_id, u_id)		**InputError** when any of: - u_id does not refer to a valid user. - channel_id does not refer to a valid channel. - dbase_id does not refer to a valid database. **AccessError** when any of: - token is not admin - u_id is not part of channel
channel/database/remove	POST	(token, channel_id, dbase_id, u_id)		**InputError** when any of: - u_id does not refer to a valid user. - channel_id does not refer to a valid channel. - dbase_id does not refer to a valid database. **AccessError** when any of: - token is not admin - u_id is not part of channel
channel/database/userlist	GET	(token, channel_id, dbase_id)	{users}	**InputError** when any of: - channel_id does not refer to a valid channel. - dbase_id does not refer to a valid database. **AccessError** when any of: - token is not admin or has permission to edit database
channel/maximum	PUT	(token, channel_id, value)	{}	**InputError** when any of: - channel_id does not refer to a valid channel. - value is below 3 **AccessError** when any of: - token is not admin
role/create	POST	(token, channel_id)	{role_id}	**InputError** when any of: - channel_id does not refer to a valid channel. **AccessError** when any of: - token is not admin
role/send	POST	(token, channel_id, u_id, role_id, is_private)	{}	**InputError** when any of: - channel_id does not refer to a valid channel. - u_id does not refer to a valid user. - role_id does not refer to a valid user. **AccessError** when any of: - token is not admin

Figure B: Interface Design table

Conceptual Modelling (State)

Channel owner admin currently using one of their channels

Role coordinator obtains information brief about the project and the roles

Adding users to roles specified by brief

Choose either public or privately

Send roles to users

Users join those specific private channels

Determine maximum number of users for specific role for that channel

Users then begin working on project specifications

Create databases for the channel so users can view, comment and or edit

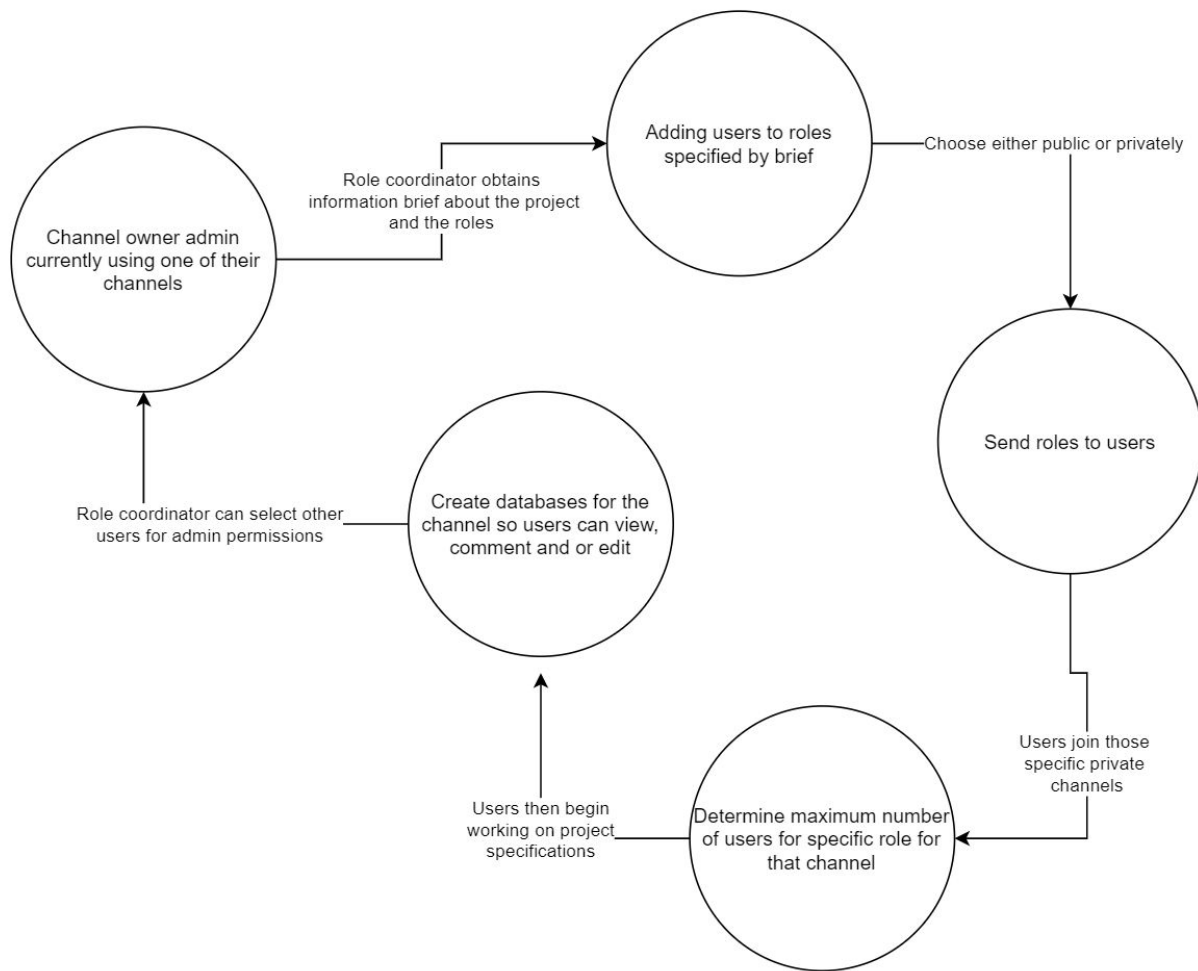Role coordinator can select other users for admin permissions

Figure C: State flow diagram displaying process of actions that combine all 3 use cases