Elicitation questions:

1. Do you encounter difficulties expressing ideas in a team due to the limits of the platform you're using? If so, what are they?
2. How concerned are you with security and privacy on the platforms you use? What reassures your security and privacy on a platform?
3. In a teamwork platform do you have struggles with the organisation of tasks? Why?
4. How do you like to communicate on message platforms?
5. Tell me about a time when your team was let down by a platform?
6. What do you often discuss on a messaging platform?
7. Do you struggle with reading messages?
8. Do you struggle with keeping up with messages?

Person 1:

Name: Angeni Bai

Email: angeni.bai@unsw.edu.au

Answers:

1. Sometimes ideas will get lost in a chat if the chat is moving quickly.

2. Not too concerned. Having 2FA reassures me that it would be difficult to hack into my account. Although I do find it weird that Discord somehow always knows what game I'm playing.

3. It's hard to see the current state of all the projects at once.

4. By texting?

5. Whenever I forget that I have dnd turned on and miss important notifications. Also when I click on a channel and make a mental note to come back and then I forget.

6. Use messaging platforms to give updates on progress and invite input from other team members

7. Not really

8. Sometimes

Person 2:

Name: Ryan King

Email: r.r.king@unsw.edu.au

Answers:

1. No. Although Discord has a messaging limit it's easy just to break it up into smaller messages

2. Not very concerned.

3. No, because many platforms have channels and threads which make it easy to organise conversations and ideas. Search functionalities are helpful as well.

4. Sending a message.

5. Slack has limits on how many messages it saves (10,000) unless you upgrade.

6. Casual stuff on a platform like Messenger and discord, important stuff on something like slack or teams.

7. No.

8. If I have a lot of unread messages, yes.

User stories

1. **As a student, I want to be able to read all past messages in a channel so that I don’t miss out on any previous information sent by my peers.**

   Given I have joined a channel/DM, when I view the message history, I can see all previous messages that have been sent into the channel/dm.

2. **As a student, I want to be able to send long messages to express complex ideas without having to spend time manually breaking up the message into smaller components.**

   Given I have joined a channel/DM, when I send a message over 1000 characters, the message will be automatically broken up into smaller parts and sent into the channel/dm.

3. **As a student, I want to be able to draw peers' attention to the channel so that I can invite their input into the discussion.**

   Given I have joined a channel/DM, when I send a message including a users name with an ‘@’ in front of it, I can tag the user in the channel/dm.

4. **As a student, I would like to be able to mark a channel/dm so that I can have a reminder to read the channel/dm later.**

   Given I can see the list of channels/DMs I’ve joined, when I hover over a channel/dm and press a button, I can flag the channel to read later.

5. **As a user, I want to be able to have two-factor authentication so that I can feel more secure from potential hackers or from someone that has my password.**

   Given I have registered with a mobile number, when I try and log in with the correct details, I should be prompted to enter a code that has been sent to my phone via SMS.

User stories that are already addressed:

1. Streams currently stores all previous messages sent free of charge.

3.Streams has a tagging feature implemented already.

Use Cases

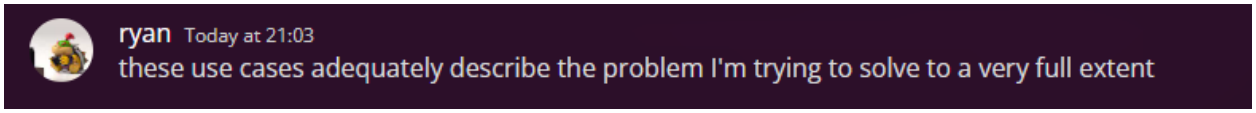
2. Sending messages greater than 1000 characters
Step 1: User selects channel/dm they have joined
Step 2: User inputs their message greater than 1000 characters
Step 3: Backend receives the message
Step 4: Backend calculates the number of messages to break the message into
Step 5: Backend divides the message into equal sections
Step 6: Each section is added to the message list one after the other
Step 7: Messages sent in the user's stats increases by the number of messages the original message was divided into

4. Marking a channel/DM
Step 1: User hovers over a channel/dm they have joined
Step 2: Next to the channel/DM name, a grey flag is displayed
Step 3: User clicks on the grey flag
Step 4: The backend registers the channel/dm as being flagged in the user's dictionary
Step 5: The flag turns red
Step 6: The flag remains next to the channel when the user stops hovering on the channel/dm
Step 7: User clicks on the flag again
Step 8: Backend registers the channel/dm as not being flagged in the users' dictionary

5. Two-factor authentication
Step 1: Front end asks for the user to enter their email and password to login
Step 2: User enters their email and password
Step 3: Backend verifies the email and password
Step 4: Backend generates a 4 digit code
Step 5: Backend contacts SMS service to send the code to the user's mobile number
Step 6: Frontend prompts the user to enter a code sent to their mobile number
Step 7: User receives the code via SMS
Step 8: User enters the code
Step 9: Backend verifies the code
Step 10: The user is logged into their account

Validation

Ryan King:



Angeni Bai:

2. Sending messages greater than 1000 characters

Very useful!! Y'all can't escape my rants now

4. Marking a channel/DM

Big fan of the flagging feature - I will definitely use it a lot to make sure I don't forget to come back to stuff.

5. Two-factor authentication

Appreciate this security feature. Will 2fa be mandatory or optional?

Interface Design

Sending messages greater than 1000 characters

Name and description	HTTP Method	Data Types	Exceptions
message/send/v2 Send 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. If the message length is greater than 1000, split the message into equal portions (to nearest ' ') and send them individually. If the message length is odd, the first half will contain an extra character.	**POST**	**Parameters:** **{ token, channel_id, message }** **Return Type:** **{ message_id }**	**InputError** when: ● channel_id does not refer to a valid channel ● length of message is less than 1 character **AccessError** when: ● channel_id is valid and the authorised user is not a member of the channel
message/senddm/v2 Send a message from the authorised user to the dm specified by dm_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 dm. If the message length is greater than 1000, split the message into equal portions (to nearest ' ') and send them individually. If the message length is odd, the first half will contain an extra character.	**POST**	**Parameters:** **{ token, dm_id, message }** **Return Type:** **{ message_id }**	**InputError** when: ● dm_id does not refer to a valid channel ● length of message is less than 1 character **AccessError** when: ● dm_id is valid and the authorised user is not a member of the dm

Marking a channel/DM

Name and description	HTTP Method	Data Types	Exceptions
channel/getflag/v1 Given a channel with id channel_id that the authorised user is a member of, return the flag status of the channel in the user's dictionary.	**GET**	**Parameters:** **{ token, channel_id}** **Return Type:** **{ is_flagged}**	**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
channel/flag/v1 Given a channel with id channel_id that the authorised user is a member of, add the channel to list of flagged channels in the user's dictionary.	**POST**	**Parameters:** **{ token, channel_id}** **Return Type:** **{}**	**InputError** when: ● channel_id does not refer to a valid channel ● channel_id is already flagged in user's dictionary **AccessError** when: ● channel_id is valid and the authorised user is not a member of the channel
channel/unflag/v1 Given a channel with id channel_id that the authorised user is a member of, remove the channel from the list of flagged channels in the user's dictionary.	**DELETE**	**Parameters:** **{ token, channel_id}** **Return Type:** **{}**	**InputError** when: ● channel_id does not refer to a valid channel ● channel_id is already unflagged in user's dictionary **AccessError** when: ● channel_id is valid and the authorised user is not a member of the channel
dm/getflag/v1 Given a DM with ID dm_id that the authorised user is a member of, return the flag status of the dm in the user's dictionary.	**GET**	**Parameters:** **{ token, dm_id}** **Return Type:** **{is_flagged}**	**InputError** when: ● dm_id does not refer to a valid channel **AccessError** when: ● dm_id is valid and the

			authorised user is not a member of the dm
dml/flag/v1 Given a dm with id dml_id that the authorised user is a member of, add the dm to list of flagged dms in the user's dictionary.	**POST**	**Parameters:** **{ token, channel_id}** **Return Type:** **{}**	**InputError** when: ● dm_id does not refer to a valid channel **AccessError** when: ● dm_id is valid and the authorised user is not a member of the dm
dml/unflag/v1 Given a dm with id dm_id that the authorised user is a member of, remove the dm from the list of flagged dms in the user's dictionary.	**DELETE**	**Parameters:** **{ token, channel_id}** **Return Type:** **{}**	**InputError** when: ● dm_id does not refer to a valid channel ● dm_id is already unflagged in the user's dictionary **AccessError** when: ● dm_id is valid and the authorised user is not a member of the dm

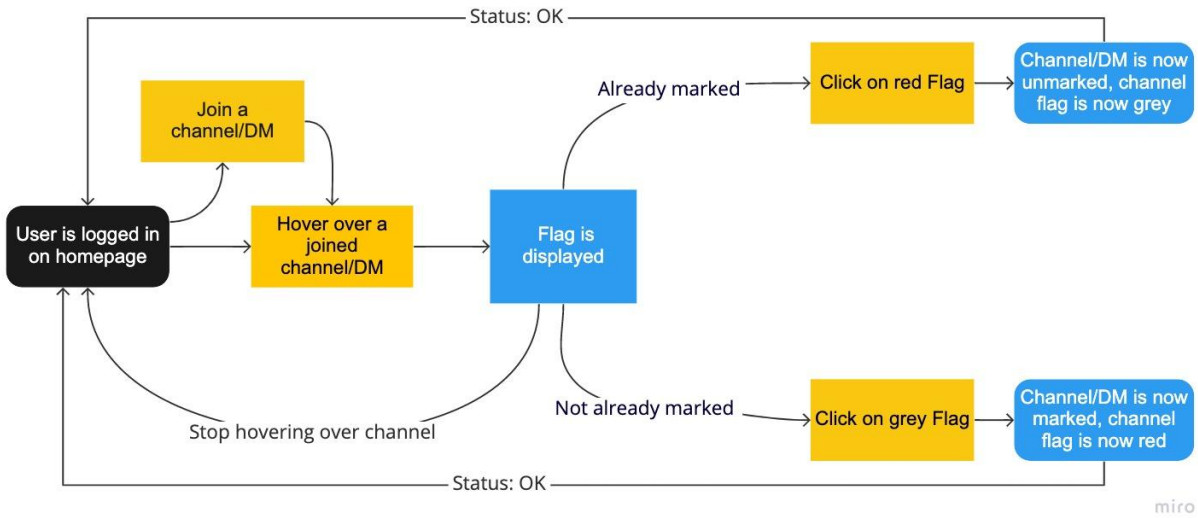
Two-factor authentication

Name and description	HTTP Method	Data Types	Exceptions
auth/register/v3 Given a user's first and last name, email address, and password, create a new account for them and return a new `token`. A handle is generated that is the concatenation of their casted-to-lowercase alphanumeric (a-z0-9) first name and last name (i.e. make lowercase then remove non-alphanumeric characters). If the concatenation is longer than 20 characters, it is cut off at 20 characters. Once you've concatenated it, if the handle is once again taken, append the concatenated names with the smallest number (starting from 0) that forms a new handle that isn't already taken. The addition of this final number may result in the handle exceeding the 20 character limit (the handle 'abcdefghijklmnopqrst0' is allowed if the handle 'abcdefghijklmnopqrst' is already taken). User may optionally input their mobile number if they want to enable two factor authentication.	**POST**	**Parameters:** **{ email, password, name_first, name_last, mobile_number}** **Return Type:** **{ token, auth_user_id }**	**InputError** when any of: ● email entered is not a valid email (more in section 6.4) ● email address is already being used by another user ● length of password is less than 6 characters ● length of name_first is not between 1 and 50 characters inclusive ● length of name_last is not between 1 and 50 characters inclusive ● **Mobile number includes non-integer characters** ● **Mobile number does not contains more than 0 but not 10 characters**
auth/login/v3 Given a registered user's email and password, returns their `token` value if no mobile number is associated with their account. Otherwise, redirects the user to SMS verification	**POST**	**Parameters:** **{ email, password }** **Return Type:** **{}** **Or** **If no mobile number is associated with the account** **{token, auth_user_id}**	**InputError** when any of: ● email entered does not belong to a user ● password is not correct
auth/sms_verf/v1 Given a code that has been sent to the user via SMS, returns their 'token' value	**POST**	**Parameters:** **{code}** **Return Type:** **{token,**	**InputError** when: ● no code is entered ● length of code is not 4 ● code includes non-integer characters

		auth_user_id}	• code entered is not correct

Conceptual Modelling

Marking a channel/DM



Sending messages greater than 1000 characters