Requirements & Design for Future Work

H09B Elephant

[Requirements] Elicitation

We used a Google Form to gather interview responses from 5 people.

Interview Questions:

1. What do you use MS Teams for?
2. How often do you use MS Teams?
3. Why do you use Teams as little or as much as you do currently?
4. How useful do you find MS Teams as a method of communication?
5. What do you think can be implemented to make the platform a safer space?
6. How can we increase engagement on a platform like MS Teams?
7. What is one feature that bugs you about MS Teams, or a feature that you would like to see implemented?
8. Any final thoughts on tools like MS Teams?

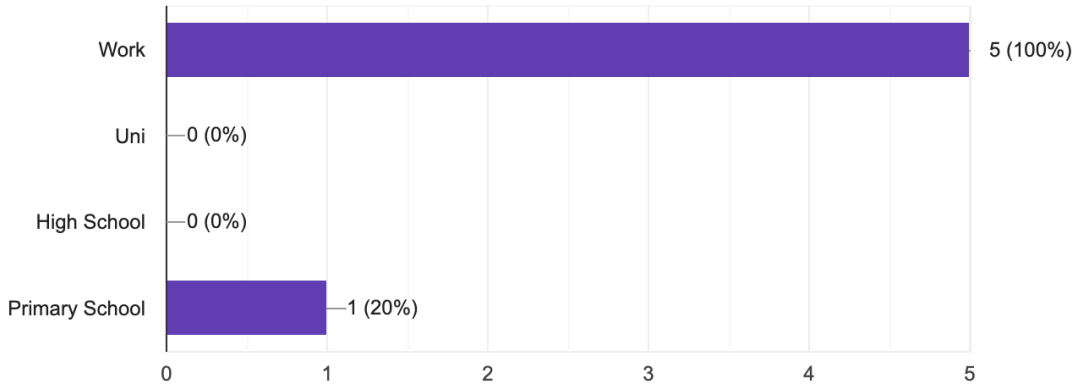
Responses:

Name 5 responses	What is your email address? 5 responses
Andrew Wood	andewood1991@gmail.com
Katy Glover	katyglover464@gmail.com
Siska Kadir	siska.kadir@gmail.com
Karen	kaz.zhang99@gmail.com
Nidhi	nidhivijay@hotmail.com

What do you use MS Teams for?

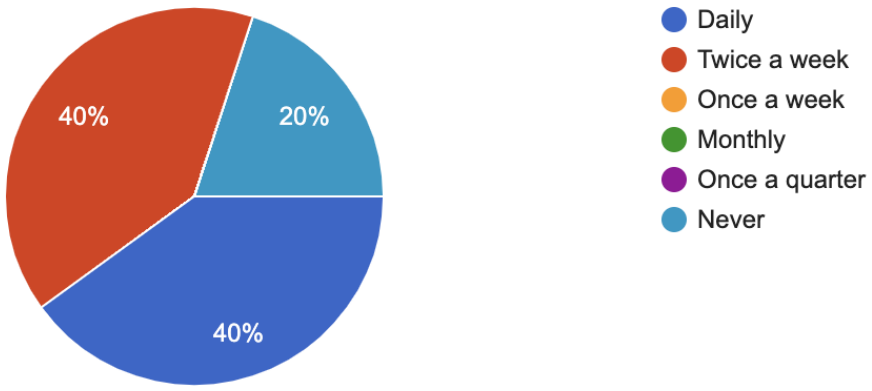
Copy

5 responses



How often do you use MS Teams?

5 responses



Why do you use Teams as little or as much as you do currently?

5 responses

I use it to manage my team of software engineers.

Just to keep tabs on the work roster

Team collaboration and meetings

Used to use it for work

I use Teams to contact clients. However my son uses Teams for school.

How can we increase engagement on a platform like MS Teams?

5 responses

I'm not sure.

Maybe make the app faster so it's quicker to open. Less friction this way.

Better search with the app

Add more features in MS Teams (something that looks and functions like discord or slack), or get rid of other platforms xd

This is a great question, and something that I think is relevant if you would like Teams to be a social environment for less formal engagement. At the moment, Teams is structured very formally.

Perhaps some sort of chat feature that would increase the likelihood of people communicating on teams for social reasons.

What do you think can be implemented to make the platform a safer space?

5 responses

Maybe some toggle for offensive language

Maybe an option to report messages.

Making it more secure when guest signing

Implement a permission required to message someone feature

Because of the digitisation occurring in schools recently, and lockdowns have made school students move to online platforms of communication. I remember my son who is in Year 7 who had to communicate on Teams during lockdown and I thought of how unregulated the environment is. If something goes wrong it's very difficult to manage team members. I think it would be great to ban swear words for young members, to make it a safe learning environment.

What is one feature that bugs you about MS Teams, or a feature that you would like to see implemented?

4 responses

I'd like some typesetting features like _italics_ and **bold** so that I can emphasise some of my messages.

Probably some kind of commands to give me more control over messages.

Not sure

see above

Any final thoughts on how tools like MS Teams can be improved?

5 responses

That's it :)

Nope.

NA

Already does the job, maybe just add more features

n/a

Proposed solution:

A channel bot that implements a language filter, poll function, and ability for channel owners to time channel members out from messaging for a certain period of time.

1) **To improve the channel environment safety** we propose implementing a language filter. It will check messages to be sent for any swear words. If a swear word exists in a message that will be sent, the language filter will prevent it from being sent.

2) **To improve Seams engagement**, we propose implementing a poll function. It will give channel members the ability to create a poll and have channel members vote in it.

3) **To improve channel owner control** and also channel environment safety, we propose implementing a timeout function. This will allow channel owners to prevent a certain user from sending messages into a channel for a certain period of time.

[Requirements] Analysis & Specification - Use Cases

User Stories:

User Story 1: (language filter)
As a channel member, I want there to be settings in place that limit profanity to ensure that the channel is a safe environment where profanity is not condoned.

Rule-based acceptance criteria:

- The language filter is only activated when the channel bot is active.
- Channel bot default state is active
- The filter works on every member in the channel when the bot is active.
- If a user uses profanities in messages, that message will be removed and a warning given to the user.
- If a user edits a message and the edited message contains profanities, then the message will also be removed and a warning given to the user.
- After 3 warnings, the user is timed out for 60 seconds.
- When timed out, a user cannot send any messages to the channel.
- Bot can be deactivated by sending a message "/dbot" into the channel

Scenario-based acceptance criteria:
Scenario: the user is a channel member and is trying to use expletives in messages
Given: the user is a channel member and has navigated to the channel page
And: the channel bot is activated
When: the user tries to send a message including any expletives into the channel
Then: the channel bot will remove the message and replace it with "This message has been removed due to profanity", and a warning will be given to the user saying the number of remaining warnings before being timed out.
Given: the user uses expletives 3 times
When: the third time the bot gives a warning,
Then: the warning will say "<user handle> has been timed out for 60 seconds for profanity!" and the user will be timed out for 60 seconds.

User Story 2: (timeout)

As a channel owner, I want to be able to prevent disrespectful members from messaging for a certain period of time, so that the channel environment is safer.

Rule-based acceptance criteria:

- The timeout command will prevent a specified user from sending any messages into the channel for a specified number of seconds.
- The timeout command is only able to be used when the channel bot is active.
- Channel bot default state is active
- Only channel owners are able to time a channel member out
- The timeout command can be applied to any member in the channel when the bot is active.
- Even if the bot is deactivated, if a timeout is in place, it will continue until the end of the specified timeout period.
- The timeout command will prevent the indicated user from sending or editing any messages in the channel for however many seconds the user is timed out for.
- Bot can be deactivated by sending a message "/dbot" into the channel

Scenario-based acceptance criteria:
Scenario: A channel member is being disrespectful to others in the channel and a channel owner wants to do something about it.
Given: the user is an owner of the channel and the channel bot is activated
When: the user types into the channel chat "/timeout <handle of the disrespectful member> <number of seconds>
Then: the disrespectful channel member is prevented from sending messages in the channel for however many seconds they have been timed out for.

User Story 3: (Poll)
As a channel member, I want to be able to make a poll, so that I can gather the opinions of channel members quickly.

Rule-based acceptance criteria:

- A poll can only be started when the channel bot is activated.
- Channel bot default state is active, so when the channel is created, the bot is automatically activated
- A poll can be started by any channel member once the bot is active.
- A user can only vote for 1 option at a time.
- Only 1 poll can run at any time within a channel
- Bot can be deactivated by sending a message "/dbot" into the channel
- Poll can only be ended by the user who started the poll.

Scenario-based acceptance criteria:
Scenario: Channel member wants to make a poll
Given: the user is a channel member and has navigated to the channel page
And: the channel bot is activated and there is no existing poll running
When: the user types "/startpoll <question> <option1> <option2> ..." into the message box
Then: the channel bot will start a poll with that question and options for everyone in the channel to vote.
Given: the poll is still in progress
When: a channel member types "/vote <option>" into the chat,
Then: the poll will update showing a vote next to whichever option the user has voted
Given: the user has already voted
When: the user types "/vote <option>" again,
Then: their vote will change and the poll will update to reflect their change of vote.
generate at least one use case that attempts to describe a solution that satisfies some of or all the requirements elicited.
Given: the user is the user who started the poll
When: they send "/endpoll" into the chat
Then: the channel bot will end the poll.

Use Cases:

Use case 1: Language filter (Background)
Use Case: prevent users from swearing in a channel
Goal in context: channel members can choose to operate in a safe channel environment where profanity is not condoned
Scope: Seams channel bot language filter infrastructure
Level: Primary task
Preconditions: users are channel members, the channel exists, the bot is active
Success End Condition: any messages channel members want to send containing expletives are removed as soon as they try to send it. Users are timed out for 60 seconds if they send 3 messages containing expletives.
Failed End Condition: messages containing expletives are able to be sent into the channel
Primary actor: channel member trying to send a message containing expletives
Trigger: channel member trying to send a message containing expletives into the channel whilst the channel bot is active

Use case 1: Language Filter (Main Success Scenario)

Step 1: channel bot is active
Step 2: channel member tries to send a message containing an expletive into the channel
Step 3: channel bot detects an expletive in the message.
Step 4: channel bot will send a warning message into the channel, naming the user and asking the user to stop swearing, and says they have 2 remaining warnings before they are timed out.
Step 5: channel bot replaces the message containing expletives with “This message has been removed due to profanity”
Step 6: channel member attempts to send another message into the channel containing expletives
Step 7: channel bot detects an expletive in the message.
Step 8: channel bot will send a warning message into the channel, naming the user and asking the user to stop swearing, and says they have 1 remaining warning before they are timed out.
Step 9: channel bot replaces the message containing expletives with “This message has been removed due to profanity”
Step 10: channel member attempts to send a third message into the channel containing expletives
Step 11: channel bot detects an expletive in the message.
Step 12: channel bot will send a message into the channel, naming the user and declaring that they are being timed out for 60 seconds.
Step 13: channel bot replaces the message containing expletives with “This message has been removed due to profanity”
Step 14: channel member is timed out for 60 seconds and prevented from sending any messages during that period.

Use case 2: timeout (Background)

Use Case: channel owner can prevent a specified user from messaging in a channel for a certain number of seconds
Goal in context: channel owners have more control over who can message in a channel for a specified period of time.
Scope: Seams channel bot timeout infrastructure
Level: Primary task
Preconditions: the channel exists, user requesting the timeout is a channel owner or there are no channel owners in the channel, the bot is active
Success End Condition: the timed out user cannot send or edit any messages in the channel for the period for which they are timed out.
Failed End Condition: the timed out user can still send/edit messages in the channel during the period for which they are timed out.
Primary actor: channel owner timing out a channel member

Trigger: channel owner sending a command in a channel “/timeout <target handle> <seconds to timeout>”

Use case 2: timeout (Main Success Scenario)
Step 1: channel bot is active
Step 2: channel owner sends a command in the channel “/timeout <target handle> <seconds to timeout>”
Step 3: channel bot detects command
Step 4: channel bot will timeout the specified user <target handle> for however many seconds <seconds to timeout> so that channel member cannot send/edit any messages during the timeout period
Step 5: channel member attempts to send a message into the channel
Step 6: server raises an error saying “You have been timed out, there is X seconds left”
Step 7: channel member attempts to edit a message in the channel

Step 9: server raises an error saying “You have been timed out, there is Y seconds left”
Step 10: time out period passes and server lets the user send/edit messages in the channel now
Step 11: user can send/edit messages in the channel now

Use case 3: poll (Background)
Use Case: creating and voting in a poll in a channel
Goal in context: channel members can start, vote in and end polls in a channel
Scope: Seams channel bot poll infrastructure
Level: Primary task
Preconditions: the channel exists, user requesting the poll commands is a channel member, the bot is active
Success End Condition: a poll is created, channel members can vote, and the poll results can be seen after every poll update.
Failed End Condition: a poll is not created, users cannot vote, poll results are not updated after every change.
Primary actor: any channel member
Trigger: channel member sends a command in the channel to start the poll with the poll question and options “/startpoll <question> <option1> <option2>...”

Use case 3: poll (Main Success Scenario)
Step 1: channel bot is active
Step 2: channel member sends a command in the channel to start the poll with the poll question and options
Step 3: channel bot detects command

Step 4: channel bot will start a poll sending a message in channel displaying the poll state as poll started, including the poll question and options
Step 5: channel member votes for an option typing the command "/vote <option>"
Step 6: channel bot updates the poll by sending a message in the channel with the updated poll statistics with an emoji next to the option the user has just voted for representing 1 vote
Step 7: channel member can add as many poll options as they want by sending the command "/addpolloption <option1> <option2>..." in the channel
Step 8: channel bot updates the poll by sending a message in the channel with the updated poll statistics including the poll question, existing options and the newly added options and their votes
Step 9: channel member changes their vote by sending the vote command again but with a different option
Step 10: channel bot updates the poll by sending a message in the channel with the updated poll statistics with an emoji next to the option the user has just voted for representing 1 vote
Step 11: channel member starts a new poll
Step 12: chanel bot will not allow a new poll to start without a user ending the existing poll
Step 13: channel member can end poll by sending the command "/endpoll"
Step 14: channel bot will end the poll and send a message in the channel with the final poll results and highlighting the most popular choice

[Requirements] Validation

Interviewees' Comments on the Use Cases:

Karen:
Original request:

> Add more features in MS Teams (something that looks and functions like discord or slack), or get rid of other platforms xd

Comments on our use cases:
These use cases definitely make Teams similar to Discord with the bot feature! I think it will be great to add more flexibility to what rules you want in your channel!

Siska:
Original request:

> Because of the digitisation occurring in schools recently, and lockdowns have made school students move to online platforms of communication. I remember my son who is in Year 7 who had to communicate on Teams during lockdown and I thought of how unregulated the environment is. If something goes wrong it's very difficult to manage team members. I think it would be great to ban swear words for young members, to make it a safe learning environment.

Comments on our use cases:
I love the language filter! I would feel much less worried about my son being on Teams, if MS Teams actually implements this!

Andrew Wood:
Original request:

> Probably some kind of commands to give me more control over messages.

Comments on our use cases:
Interesting how you made the channel owners have more control over messages. The timeout function would be useful for channel owners to have something to leverage over misbehaving channel members. Good for a school or university environment, but might be a little over the top for corporate use. Though I understand all these features will be included in the bot which can be turned off, so if corporate environments don't want to use it, it's their choice to turn the bot off. Great idea to make a bot.

[Design] Interface Design

Name & Description	**HTTP Method**	**Data Types**	**Exceptions**
/channel/activatebot/v1 Activates the bot. If channel has no owners then bot can be activated by anyone.	POST	**Parameters:** { token, channel_id} **Return Type:** { }	**InputError** when any of: ● channel_id does not refer to a valid channel **AccessError** when: ● channel_id is valid and the authorised user

			does not have owner permissions in the channel
/channel/deactivatebot/v1 Deactivates the bot	POST	**Parameters:** { token, channel_id} **Return Type:** { }	**InputError** when any of: ● channel_id does not refer to a valid channel **AccessError** when: ● channel_id is valid and the authorised user does not have owner permissions in the channel
/channel/timeout/v1 Times out the user u_id for a specified period of time. The user who is timed out cannot send/edit messages during the timeout period.	POST	**Parameters:** { token, channel_id, u_id, timeout_length} **Return Type:** { }	**InputError** when any of: ● channel_id does not refer to a valid channel ● timeout_length is a negative integer ● u_id does not refer to a valid user ● u_id refers to a user who is not a member of the channel **AccessError** when: ● channel_id is

			valid and the authorised user does not have owner permissions in the channel
/channel/languagefilter/v1 Runs every time a message is edited or sent. Checks messages for any swear words listed in swear_words list. If a swear word exists in a message that will be sent, replace the message with “This message has been removed due to profanity”. If a user edits a message to a new message containing profanity, this message will also be removed and replaced with “This message has been removed due to profanity”. If the user swears 3 times, they will be timed out for 20 seconds for every attempted message containing a swear word. I.e. if they swear 3 seconds, they will be timed out for 60 seconds. If they swear another 3 times, they will be timed out for 120 seconds.	POST	**Parameters:** { token, channel_id, message_id} **Return Type:** {profanity_found }	**InputError** when any of: ● message_id does not refer to a valid message within a channel/DM that the authorised user has joined ● channel_id does not refer to a valid channel
/channel/startpoll/v1 Initiates a poll in the specified channel, with the poll question and with as many options as inputted.	POST	**Parameters:** { token, channel_id, question, options} **Return Type:** {poll_id }	**InputError** when any of: ● channel_id does not refer to a valid channel ● Question is of length zero ● Options is of length zero

			AccessError when: ● Token is not a member of the channel
/channel/vote/v1 Takes in one option, considering that the option inputted is valid (case sensitive), adds a vote to that option. A user can only choose 1 option. If this user has already voted, change their vote to the new option.	POST	**Parameters:** { token, channel_id, poll_id, option} **Return Type:** {vote_id }	**InputError** when any of: ● channel_id does not refer to a valid channel ● poll_id does not refer to a valid poll ● option does not refer to an existing option **AccessError** when: ● Token is not a member of the channel
/channel/addpolloption/v1 Adds options to existing poll, will read as many options as inputted	POST	**Parameters:** { token, channel_id, poll_id, options} **Return Type:** { }	**InputError** when any of: ● channel_id does not refer to a valid channel ● poll_id does not refer to a valid poll ● options is of length zero **AccessError** when: ● Token is not a member of the channel
/channel/endpoll/v1	POST	**Parameters:** { token,	**InputError** when any of:

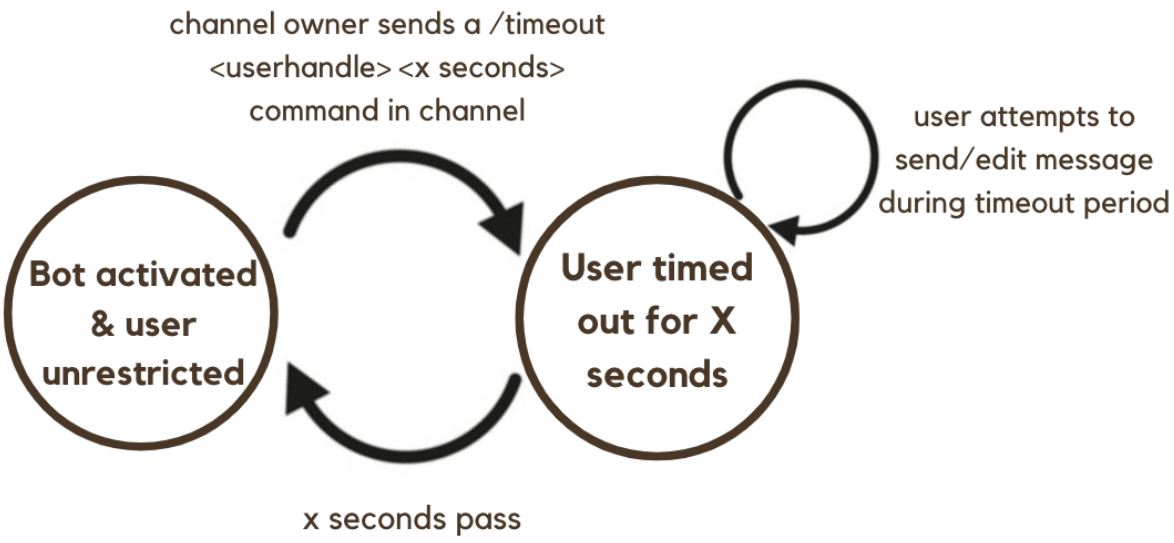
Ends the existing poll.
Returns the highest voted option(s).
		channel_id, poll_id}
		
		Return Type:
		{
		most_popular
		}
			• channel_id does not refer to a valid channel
			• poll_id does not refer to a valid poll
			
			AccessError when:
			• Token is not a member of the channel

/channel/help/v1	POST	**Parameters:** { token, channel_id} **Return Type:** { }	**InputError** when any of: • channel_id does not refer to a valid channel **AccessError** when: • Token is not a member of the channel

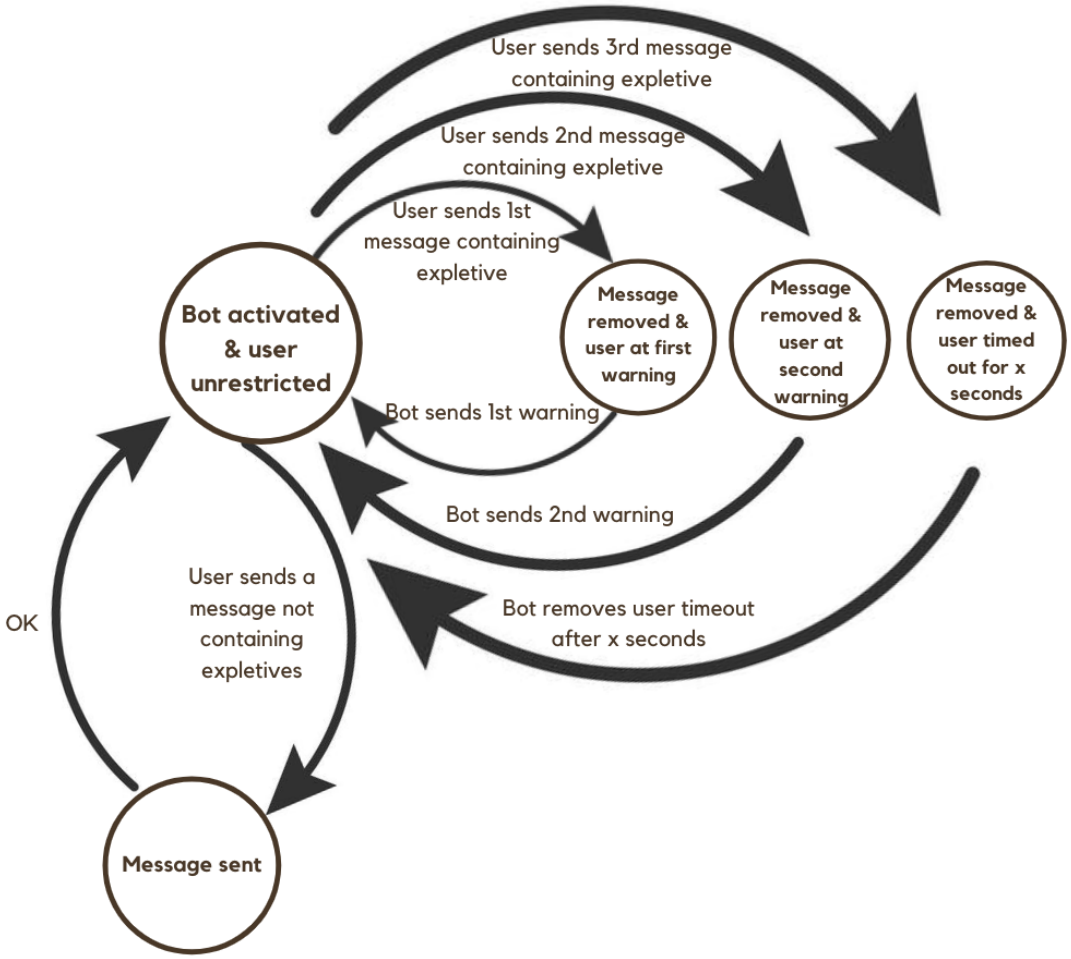
[Design] Conceptual Modelling (State)

State Diagrams:

TIMEOUT IN CHANNEL



LANGUAGE FILTER IN CHANNEL



POLL IN CHANNEL