Planning

Elicitation

Questions asked:

1. Do you use any messaging platforms to work on projects together and if so, which platform?
2. Why did you choose to use the platform you stated in question 1? What do you look for in a messaging platform?
3. Have you experienced any issues or restrictions on these platforms?
4. When working together as a team on these platforms, what issues have you had with collaboration?
5. What do you think are the best features in these platforms that allow for efficient collaboration work?

Responses:

Name: Luka Ryan
Email: lukaryan01@gmail.com
Response to Questions:

1. Yes, Facebook messenger, Microsoft teams
2. Well known, easy, work with pre-existing accounts
3. I often struggle to realise who is an admin or a tutor in Microsoft Teams when working on team projects.
4. It can be easy to miss calls on teams
5. screen sharing

Name: Tyrone Salomon

Email: tyroneaquinosalomon@gmail.com

Response To Questions:

1)Yes, I have used Microsoft Teams and Discord,

2)Used Microsoft Teams because it is used by UNSW for many courses. As a result, all group members will have access to it. Also, it allows for a professional social environment to talk with other group members and tutors who don't want to mix their outside life with Uni. Moreover, Discord is used often because the majority of people know how to use it and it's free. I look for a messaging platform that has a neat interface, is accessible, easy to use.

3)In Discord, there is a limit to how much code can be sent before it is converted into a file.

4)The main problem when using Microsoft Teams is that people are not active on it, and as a result, they often miss messages and calls. On the other hand, some people don't use Discord, making it impossible to collaborate on.

5)Specifically, in Discord, I like the feature that channels can be created in a server which can be dedicated for different sections for a project. Secondly, an owner of a server can ban users and prevent them from joining back. Lastly, I have the ability to mute people and channels that I find disruptive when collaborating with my project members. For Microsoft Teams, I like the feature where the group can collectively work on a file together simultaneously.

Analysis & Specification - Use Cases

Tyrone - Use Case One

Tyrone's User story

User acceptance criteria:

Tyrone should be able to send chunks of code without having to send it as a file and also have a platform with all of his team members available on. If these conditions are met, his first user story is resolved.

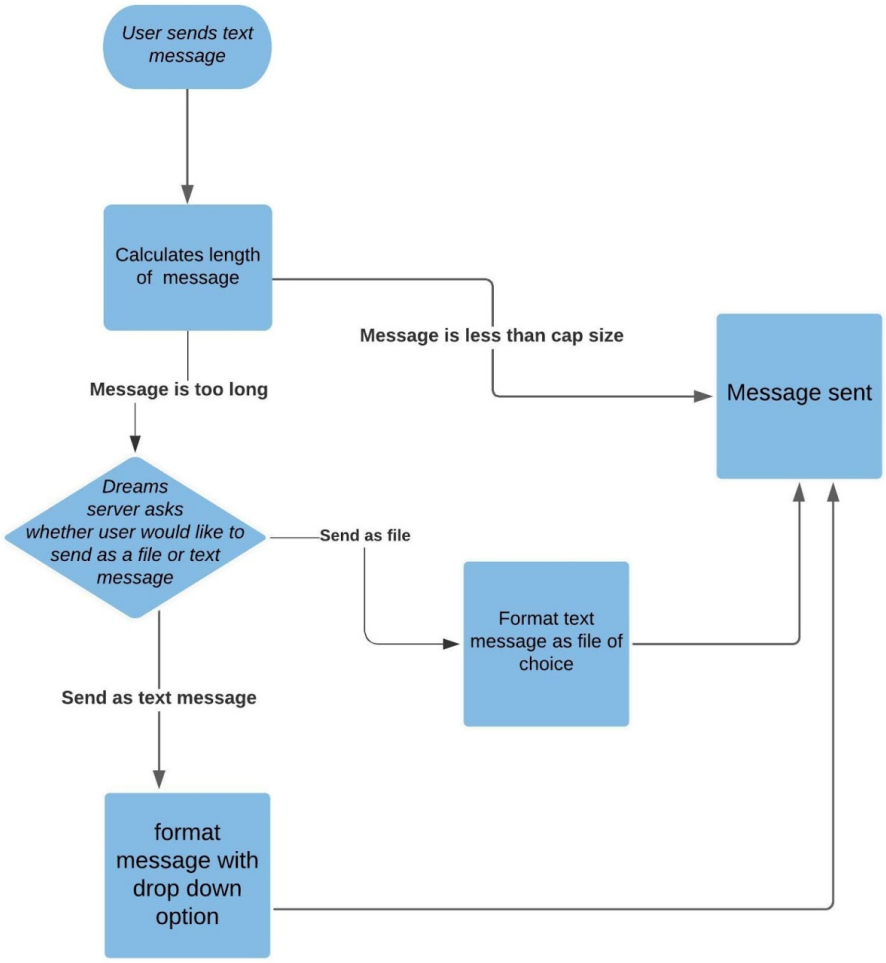
User story:

Tyrone is a university student at UNSW who uses Discord and Microsoft Teams to collaborate and work on team projects. The main issue Tyrone found when using discord for collaborative work was that he was capped at the amount of characters that could be in a text message. This would be an issue when sending paragraphs or sending chunks of code to team members as it would take a lot more time to review a peers work.

Tyrone's use case:

- **Use Case:** Have no cap on message size.
- **Goal in Context:** When sending work for peer review for team mates, Tyrone should be able to send this without having a cap on the message size. Therefore the person receiving the message doesn't have to open the work in a file opener and can stay on the dreams platform making teamwork and collaboration more efficient and simpler.
- **Scope:** messages and dm
- **Preconditions:** User is registered and a part of a channel involving other team members.
- **Level:** Subfunction
- **Success End Condition:** Messages are able to be sent without a cap and have the option to choose whether they want to convert a message to a file of their choice. Messages that pass the cap value will have a drop down option so it doesn't take too much space in the viewing of channel messages.
- **Failed End Condition:** When messages reach a certain character length, dreams will either not allow messages to be sent or convert messages to a text file.
- **Primary Actor:** User (Tyrone)
- **Trigger:** User wants to send a piece of their to a team member for peer review

MAIN SUCCESS SCENARIO:



Validation:

Question: How well do these use cases address the issue you are trying to solve?

I think that the use case addresses the issue very well and I like how there is an option to convert it to a file as well making the message have a drop down option so it doesn't make it hard to see other messages.

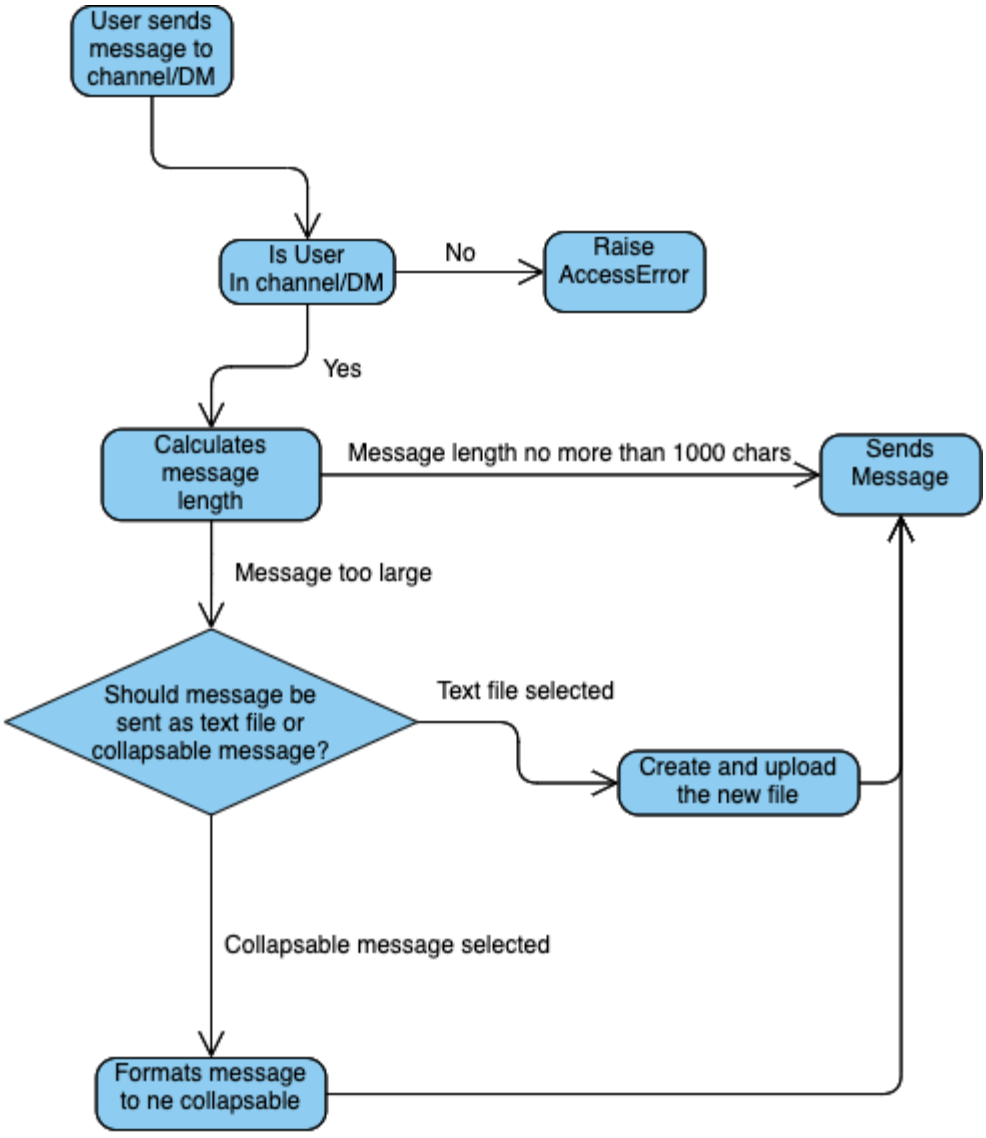
Interface Design

The problem that Tyronne's user story addressed involves sending a message larger than 1000 characters. If the message is, however, more than 1000 characters, the message would instead be sent as a text file generated from the message the user intends on sending. The HTTP endpoints would following the following structure:

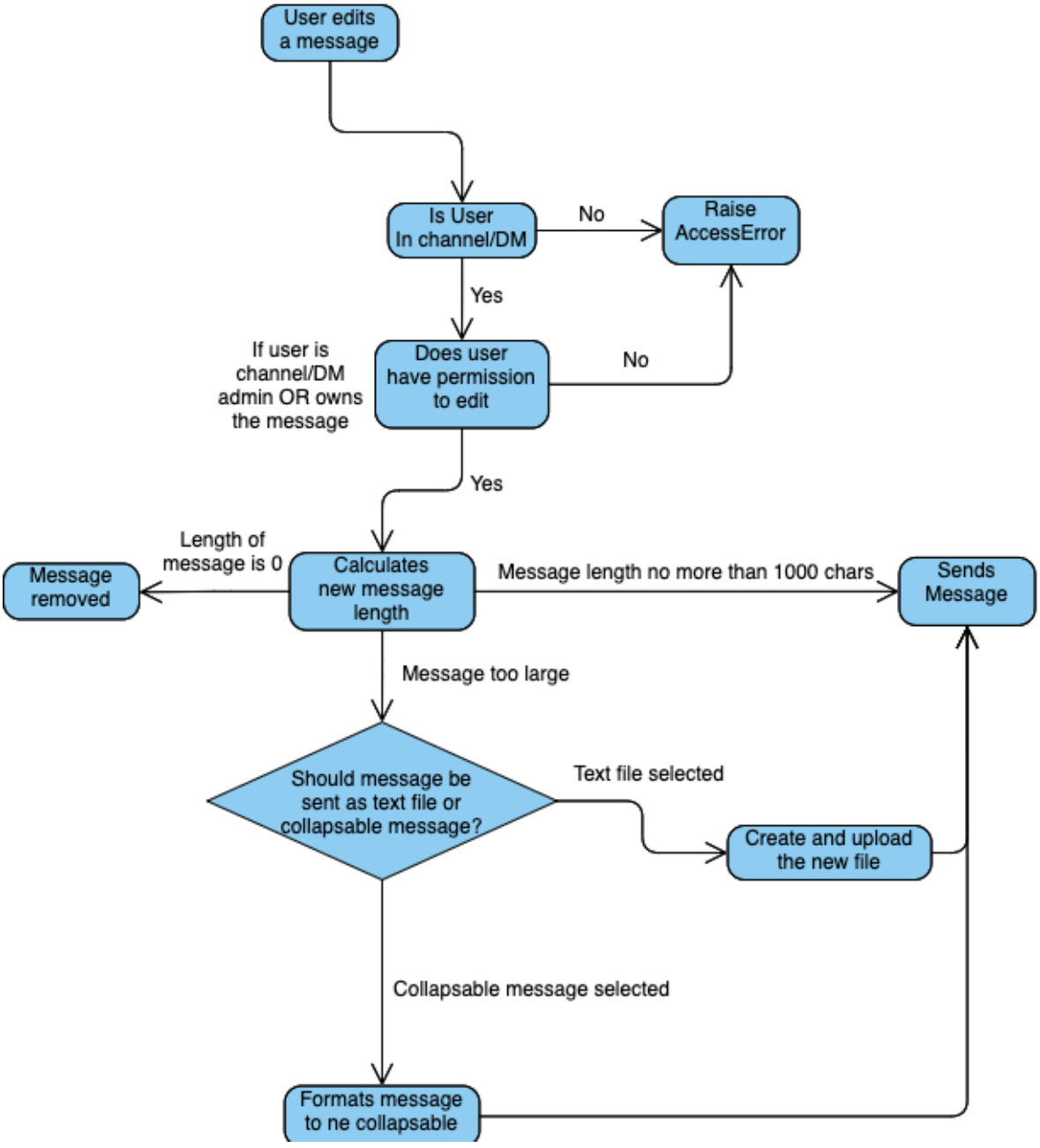
Name & Description	**HTTP Method**	**Data Types**	**Exceptions**
message/send/v3 Send a message from authorised_user to the channel specified by channel_id. If the message is 1 to 1000 characters, send it as a normal message. If the message is greater than 1000 characters, the user should be given the option to either send the message as a text file, or to send as a collapsed textbox represented when displaying the message on the front end.	POST	**Parameters:** (token, channel_id, message) **Return Type:** { message_id }	**AccessError when:** the authorised user has not joined the channel they are trying to post to
message/senddm/v2 Send a message from authorised_user to the DM specified by dm_id. If the message is greater than 1000 characters, the user should be given the option to either send the message as a text file, or to send as a collapsed textbox represented when displaying the message on the front end.	POST	**Parameters:** (token, dm_id, message) **Return Type:** { message_id }	**AccessError when:** the authorised user has not joined the channel they are trying to post to
message/edit/v3 Given a message, update its text with new text. If the new message is an empty string, the message is deleted. If the new message is greater than 1000 characters, the user should be given the option to either send the message as a text file, or to send as a collapsed textbox represented when displaying the message on the front end.	PUT	**Parameters:** (token, message_id, message) **Return Type:** {}	**InputError when:** **AccessError** when none of the following are true: Message with message_id was sent by the authorised user making this request The authorised user is an owner of this channel (if it was sent to a channel) or the **Dreams**

Conceptual Modelling (State)

Below is the updated message send and message senddm function with the new states and features



Below is a similar implementation for the edit function



Luka - Use Case Two

Luka's User story:

User acceptance criteria:

If Luka could quickly and effortlessly see what status users have on the platform's server during interface in the server his story will be resolved. This means he should click a user profile to see their role/status.

User story:

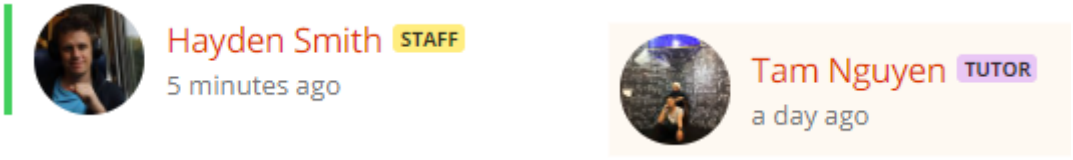
As a student in Microsoft Teams, Luka wants to be able to easily and effortlessly identify who people are. For example, when reading discussions in Teams, it should be clear to see which users are tutors, lecturers and students. This is so he can see which information is credible and understand the context of the discussion more clearly. Currently, Luka has to click each user and view their profile to see their status in the server.

Luka's Use Case:

- **Use Case:** Be able to view user's role as a tag next to their name
- **Goal in Context:** When surfacing the collaborative platform's interface, Luka should be able to easily recognise all the user's role/status in the server. This should be displayed wherever the user(Luca) can see other user's names. i.e server owners, tutors etc.
- **Scope**: User profiles,
- **Preconditions**: The user is logged into the collaborative platform and can view the interface.
- **Level**: Minor task
- **Success End Condition:** The user(Luca) can view all the discussions in the messaging platform whilst receiving feedback on who the owners and admins are and knows which information is credible or not. User names should appear as the following:

  Before-

After-



- **Failed End Condition:** Luca will view the channels or direct messages interface in Dreams and not be able to easily see which comments are made by tutors, lecturers, admins and owners.
- **Primary Actor:** User(Luca)
- **Trigger:** Luca wants to access the channels or direct message interface in Dreams.

MAIN SUCCESS SCENARIO

1. User(Luka) wants to read channel/dm messages
2. User calls ‘channel/messages/v2’ or ‘dm/messages/v2’ to view messages
3. All the users who are an admin or channel owner will have a special tag next to their name.

Validation:

Question: How well do these use cases address the issue you are trying to solve?

The use cases effectively address the issue I have raised. However, it would be desirable if the tagged user name could be highlighted, making it more noticeable. Nevertheless, the main issue has been addressed.

Interface Design

When a channel is created and there are various people who are members inside the channel, the channel owner and the other channel admins should be able to assign roles to other team members to distinguish their roles between each other. Initially when a channel is created, the individual who created it is automatically labelled as an owner and admin however when other individuals join they are labelled as a member. If the channel owner makes another user admin they are then labelled as admin. The admin and owners have access to a wide range of labels including, tutor, lecturers, students etc.

Name & Description	HTTP Method	Data Types	Exceptions
channel/label/v1 Given an active admin token, labels a user in the channel to a specific label. Will be given special treatment to be displayed on the frontend visually.	POST	**Parameters:** (token, channel_id, u_id, label_id) **Return Type:** { }	**AccessError when:** the authorised user is not an owner or admin in the channel **InputError when:** the label_id is invalid
dm/label/v1 Given an active admin token, labels a user in the dm to a specific label. Will be given special treatment to be displayed on the frontend visually.	POST	**Parameters:** (token, dm_id, u_id, label_id) **Return Type:** { }	**AccessError when:** the authorised user is not an owner or admin in the dm **InputError when:** the label_id is invalid

Conceptual Modelling (state)