Planning

CS1531_FRI09B_ECHO

Team Members:

Wang Zhang

Siddh Rawal

Sunny Tang

Sarth Bishnoi

Amin Ghasembeigi

[Requirements] Elicitation

Target User 1

Name: Roberto Gallardo

Email Id: z5261474@ad.unsw.edu.au

Q1 Having used Dreams, what are some features that require improvement?

Color scheme is monochromatic and a bit boring. It also makes it hard to differentiate the different functionalities of the app. The lack of usernames makes it unintuitive to tag people, and even if they are tagged, this won't show up in notifications, which is also a frustrating aspect.

Q2 Having used Dreams, is there a similar application that you use, which serves a team communications purpose?

Microsoft Teams, but a bit different as this one can also be used for meetings and screen sharing etc.

Q3 How often do you rely on this application to communicate?

Maybe three to four times a week. My main use however is for video meetings and editing shared documents, not chatting per se.

Q4 What common tasks do you carry out on this application?

Editing documents, creating presentations, sharing presentations, and sometimes reaching people I don't have on messenger.

Q5 What are some features on this application that simplify your experience doing your tasks?

Live editing is very helpful, the fact that you can tag people in chats and call them, also you can see if people are busy or active or not.

Q6 Did you experience any of the same issues from Dreams on your application?

Teams has a clearer color scheme and title setup, which facilitates the user interface. It also has more subdivisions, such as teams, classes, chats and subgroups. Notifications clearly shows up as a little red thing in whatever compartment. Essentially, Teams does not host the same problems I found while using Dreams.

Problem Raised:

It is not intuitive to tag people, more so the format in which we need to type the user's name. Interface of the Dreams can be made better (in terms of color, design etc.).

Proposed solution:

Add a new feature below the message bar which is a Tag bar. In this we can search the names or usernames of channel members and click on their name to include '@<First Name><Last Name>' in the message. This process can be repeated multiple times to tag multiple people.

We can improve the colour selection of the page to make the site more appealing to look at and simultaneously make the functionality of the different buttons more immediately clear.

Target User 2

Name: Ankit Gautam
Email Id: Ankit.Gautam@team.telstra.com

Q1 Having used Dreams, what are some features that require improvement?

The remove dm button doesn't work and there is no way to know if a person has seen my message.

Q2 Having used Dreams, is there a similar application that you use, which serves a team communications purpose?

Teams.

Q3 How often do you rely on this application to communicate?

I use it daily.

Q4 What common tasks do you carry out on this application?

Dm people and communicate in main channels.

Q5 What are some features on this application that simplify your experience doing your tasks?

Teams has feature that shows status of people, it works on the phone, and you can message people directly easily.

Q6 Did you experience any of the same issues from Dreams on your application?

No.

Problem Raised

Remove Dm is currently not working and there is no feature provided to know if the message sent has been seen by the other user.

Proposed solution.

Fix remove Dm and add a new feature that shows the status of a message (Sent, received and seen).

[Requirements] Analysis & Specification - Use Cases

Target User 1

User Story 1

As a user, I want to be able to distinguish between different functionalities of the interface so that it is easy to work with.

Acceptance Criteria 1

- Login on Dreams app.
- It should be easy to differentiate between different functionalities.

User Story 2

As a user, I want to be able to see a list of filtered names of people that match my typed input so that it's easier to tag people.

As a user, I want to be made aware of the user handle format so that I can intuitively tag people in channels.

Acceptance Criteria 2

- In the message box (inside a Dm or channel) type '@'.
- list of people(filtered) that match the following letters should be shown above the message box.

Use Case 1

Environment:

- Use case: Improve the interface.
- Goal in context: User's need to see a better interface to make things more organised.
- Scope: Frontend.
- Level: Primary task.
- Preconditions: The user is already registered.
- Success End condition: The interface is good enough to intuitively differentiate the functionality on it.
- Failed End condition: The interface is not good enough to differentiate the functionality on it
- Primary Actor: User.
- Trigger: User logs in.

Main success Scenario:

- Step1. Frontend asks for an action.
- Step2. User logs in.
- Step3. Frontend asks backend to verify the user.
- Step4. Backend verifies the user's membership.
- Step5. Frontend shows the interface on the screen.

Use Case 2

Environment:

- Use case: Provide a list of filtered names using user's input into tag box.
- Goal in context: User's need to see a list of filtered names to make it easy to tag other users.
- Scope: frontend and backend
- Level: Primary level
- Preconditions: User is logged in and channel/Dm's is opened (that the user intend to tag people in) and user is engaging with tag box, their membership in the channel/Dm has been confirmed.
- Success End condition: User can tag a member, using the tag box, in a message.
- Failed End condition: Length of sequence of letters is more than maximum.
- Primary Actor: User
- Trigger: User types '@' in the message box.

Main success Scenario:

- Step1. Frontend waiting for an action (for Tag Box).
- Step2. User types a sequence of letters.
- Step3. Frontend takes new sequence of letters typed.
- Step4. Frontend asks backend for a list of users with 'string handle' that match the letters.
- Step5. Backend searches for handle strings that matches the sequence of letters.

- Step6. Backend returns a list of 'handle string' that match the sequence of letters to the Frontend.
- Step7. Frontend shows the list of 'handle string' that match the sequence of letters.
- Step8. Frontend waiting for an action (for Tag Box).
- Step10. User presses 'Enter'.
- Step11. Frontend puts the 'handle string' inside the message box with '@' as the first element of the string.
- Step12. Moves the cursor to the message box.

Target User 2

User Story 1

As a user, I want to be able to remove an existing Dm so that I have less things to look at.

Acceptance Criteria 1

- Select an already existing Dm.
- Remove Dm by clicking on the 'Remove Dm' button
- The Dm shouldn't exist.

User Story 2

As a user, I want to know the status (sent, received and seen) of the message/Dm typed by me in the message box so that I know if the message/Dm has been sent, received or seen by the recipients, and act accordingly.

Acceptance Criteria 2

- Typing a message (inside message box) in a channel or Dm.
- Hit enter on the keyboard/click on send button.
- Mark the message as seen, delivered or received (with names) appropriately.
- The status should be available on the screen to get information quickly.

Use Case 1

Environment:

- Use case: Remove Dm.
- Goal in context: User's need to delete Dm's to make things more organised.
- Scope: Frontend and backend.
- Level: Primary task.
- Preconditions: The user is logged in.
- Success End condition: Dm the user is trying to remove exists.
- Failed End condition: Dm the user is trying to remove doesn't exist.
- Primary Actor: User

- Trigger: User click remove Dm

Main success Scenario:

- Step1. Frontend asks for an action.
- Step2. User clicks on a Dm.
- Step3. Frontend asks backend if the user is member of the Dm.
- Step4. Backend verifies the user's membership.
- Step5. User clicks on remove DM.
- Step6. Frontend asks backend to remove the Dm.
- Step7. Backend removes the Dm.

Use case 2.

Environment:

- Use case: Provide status of message/Dm.
- Goal in context: User's need to know the status of a message.
- Scope: Frontend and Backend.
- Level: Primary task.
- Preconditions: User is logged in, looking at Dm or channel and their membership has been confirmed.
- Success End condition: User has the permission to view the channel/Dm.
- Failed End condition: User does not have the permission to view the channel/Dm.
- Primary Actor: User
- Trigger: Looking at the Dm or channel which contains the message.

Main Success Scenario 1:

- Step1. Frontend asks for the messages in the channel/Dm.
- Step2. Backend provides the messages.
    - The message data type now has two versions; message_channel and message_dm, which include status keys for Sent, Delivered and Seen.
        - For message_dm, the status keys will have a True or False value attached.
        - For message_channel, the status keys will have a True or False value attached to Sent and Delivered, and a list of channel members who have seen the message attached to Seen.
- Step3. Frontend shows the message (containing status with the names of other users) to the user.

Main Success Scenario 2:

- Step1. Frontend asks to send a message in the channel, with initial status key of Sent and Delivered to have value True but Seen will have attached False (Dm) or an empty list (channel).
- Step2. Frontend asks for the messages in the channel/Dm.

- Step3. Backend provides the messages and simultaneously alters the Seen key to True (Dm) or adds the username to the list of channel members, from whose user_id was confirmed upon entry to the channel/Dm.
- Step4. Frontend shows the message (containing status with the names of other users) to the user.

[Requirements] Validation

Roberto: The problem before was that it wasn't intuitive to tag people because format of the tag wasn't specified. However, the added tagging bar underneath the message bar clears up this problem and also supports the issues I experienced with differentiating the functionality of the different parts of the page.

Ankit: Absence of feature that tells the status of a message made it hard to communicate well with people but, now that the feature is added it's easier to talk to people.

[Design] Interface Design

New Data Types	
Name	Type
message_channel	Dictionary
message_dm	Dictionary
status_sent	Boolean
status_delivered	Boolean
status_seen_dm	Boolean
status_users_seen	List
tag_query_users	List
user_query_str	String

Proposed HTTP Routes				
Name and Description	HTTP Method	Data Types	Exceptions	Description
message/searchuser/v1	GET	Parameters: (token, channel_id, user_query_str) Return type: {tag_query_users}	InputError: -The user_query_str doesn't match any of the users in the channel. AccessError: -The token is not valid or doesn't belong in the channel.	Given a query string and a channel/Dm id, this route will call on the searchuser function. This will search through all the users in that channel/Dm and return the names of the users that match the query string.
message/messagechannelseen/v1	GET	Parameters:	AccessError:	Upon clicking the 'Seen by' option on a

		(token, channel_id, message_id) Return type: {users_seen}	-The token is not valid or doesn't belong in the channel.	message, the backend will post a GET request of the status_users_seen list in the message which matches the message_id of the message in question.
message/messagesent/v1	PUT	Parameters: (token, message_id, channel_id) Return: None	AccessError: -The token is not valid or doesn't belong in the channel.	When the user sends a message, the channel will send a PUT request with the message to assign the value True to the Sent status in the message data type.
message/messagedelivered/v1	PUT	Parameters: (token, message_id, channel_id) Return: None	AccessError: -The token is not valid or doesn't belong in the channel.	When a user's channel receives a message, the channel will send a PUT request with all the messages in the the channel to assign the value True to the Delivered status in the message_channel/dm data type.
message/messagedmseen/v1	PUT	Parameters: (token, channel_id) Return: None	AccessError: -The token is not valid or doesn't belong in the channel.	When the user opens a DM, the channel will send a PUT request with all the messages in the the DM to assign the value True to the Seen status in the message_dm data type.
message/messagechanneladdseenuser/v1	PUT	Parameters: (token, channel_id) Return: None	AccessError: -The token is not valid or doesn't belong in the channel.	When the user opens a channel, a PUT request will be sent to add that user's id to a list, in the message_channel

				data type, of channel members that have seen the message. This will happen for every message in the channel.

[Design] Conceptual Modelling (State)