# Requirements & Design for Future Work

COMP1531

WED13C DORITO

# 1. Elicitation

We interviewed 3 people about their problems with current group communication tools so we could build future requirements. Their responses are shown below:

User A: Winnie Ye (email: winnieye705@gmail.com)

User B: Brittany He (email: brittanyhe2000@gmail.com)

User C: Sarah (z5260324@ad.unsw.edu.au)

## Interview Responses

1. Have you ever used a communication tool such as MS Teams? What is your most preferred collaboration tool for teamwork?

| | |
|---|---|
| **User A** | I haven't used MS Teams before. Have used Zoom, Skype, FaceTime, Messenger Video Chat. Favourite is messenger call but for group work it has been Zoom. |
| **User B** | No, I have never used MS Teams. I use Zoom, Google Docs and Messenger (Facebook) for groupwork. |
| **User C** | Yes, messenger and google doc |

2. When you are a team leader, what challenges do you face in using your current collaboration tool in terms of achieving effective communication, in delegating tasks or managing group meetings?

| | |
|---|---|
| **User A** | There is no tool within zoom which allows me to view another team members timetable and schedule in meetings, which means planning needs to be done manually within the meeting itself. |
| **User B** | There is no tool in zoom that allows us to delegate tasks, and sometimes it can be laggy. Google docs is great when working on the same document in real time, and you can add comments around the document to discuss things. |
| **User C** | Microsoft teams can be hard to navigate when looking for class material and Moodle doesn't have a call option which makes it hard to use on its own. |

3. As a team member, what challenges do you face in terms of structured communication or completing/viewing shared tasks (e.g. pair programming)?

| | |
|---|---|
| **User A** | Zoom should have an emoji, or sign etc where you can indicate you are AFK (away from keyboard) or are not present. Sometimes you need to leave the meeting just for a few minutes but don't want to interrupt whoever is talking. You also can't have more than one people have access to the whiteboard at the same time, or share screens simultaneously. |
| **User B** | Only one person can share their screen on Zoom at a time. Can't have mutual control over a screen at the same time. Is either one person or the other which makes it difficult at times. |
| **User C** | The main challenge I faced when having to produce a video as a group project, is that mainly one person is able to make the video and edit it. Collaborating on making videos is hard and impossible through online meetings. However having to hold meetings via online was hard to ensure everyone was focused and working unless progress was seen on the google doc. |

4. What issues do you face in documenting, recording or reviewing team meetings?

| | |
|---|---|
| **User A** | Zoom also doesn't have the option to write down minutes either, you would have to use a different app to do this. Would be nice to be able to have this feature integrated within Zoom, and after the meeting anyone can access the minutes. |
| **User B** | Microsoft teams can be difficult to navigate when it comes to going back and looking over messages. The record function has stopped on its own before as well. |
| **User C** | They have to be documented while the meeting is occurring to ensure information is able to be reviewed by everyone after. We did that through google docs so it was accessible to everyone. We didn't record meetings as we held meetings via messenger. |

5. As a user, what problems do you face in tracking team and individual milestones/progression during a project/collaboration?

| | |
|---|---|
| **User A** | You can't message an individual/team member via zoom when you're not in a meeting with them already. This results in us needing to find an alternate social communication platform which we can use to message each other outside of meeting time. |
| **User B** | When in breakout rooms, it would be great to have an accessible way to communicate with everyone instead of needing to leave the breakout rooms and going back to the main room. You also can't contact people outside of Zoom meetings. It would be useful to have an individual DM/private message system on Zoom where you can message people outside of meetings. |
| **User C** | It was hard since we mainly collaborated via online means so tracking progress was hard. We had to have regular and frequent meetings to ensure personal deadlines were met. And during the meetings we had to rely on checking individual progress on google docs to ensure whether they were focused and working or not. |

# Analysis & Specification

## Use Cases

**User A**

| **User Story** | **Use Case** | **Main Success Scenario** |
|---|---|---|
| As a team leader, I want to have a meeting scheduler function so that we can generate agreement on the team to hold a future meeting.<br>▪ "Meetings" button at bottom right of channel page<br>▪ User scheduling meeting must have a valid user token<br>▪ Allows user to choose time blocks from calendar | Use Case: Schedule meetings<br>Goal in Context: Channel users are presented with a range of meeting time options and can select preferred options<br>Scope: database, frontend interface<br>Level: Primary task<br>Preconditions: user must be a member of the channel the meeting is scheduled in<br>Success End Conditions: Successfully displays meeting time options to all channel | Step 1.<br>User A clicks the meeting button.<br>Step2.<br>User A can see all current meetings and clicks the 'schedule' button which opens panel with meeting criteria selection<br>Step 2.<br>User A selects meeting criteria and creates Options 1 and 2 before hitting send.<br>Step 3. |

|  |  |  |
|---|---|---|
|  | members and allows them to choose preferred times<br>Failed End Conditions: Users unable to select time<br>Primary Actor: User<br>Trigger: Click on ‘Meetings’ tab and create a new meeting when the ‘schedule’ button is clicked. This will allow the user to select the meeting criteria including the meeting time options. | All the relevant meeting details as well as a poll for current meeting time appears on meetings page.<br>Step 4.<br>User B selects “Option 1” and clicks “vote”.<br>Step 5.<br>Meeting info is shown, showing that “Option 1” has one vote and “Option 2” has no vote.<br>Step 6.<br>User B selects “Option 2” and clicks “vote”.<br>Step 5.<br>Meeting info is shown, showing that “Option 1” has one vote and “Option 2” has one vote as well. |

**User B**

| User Story | Use Case | Main Success Scenario |
|---|---|---|
| As a team leader, I want a way of assigning tasks to different users, thereby notifying them of a task pending completion.<br>▪ Taskboard button at bottom right of channel page<br>▪ User assigning tasks must have a valid user token<br>▪ User being assigned a task must have a valid user token<br>▪ User assigning tasks must be the channel owner<br>▪ User being assigned a task must be an existing channel member<br>▪ Assigned tasks appear when taskboard >> view tasks is clicked. | Use Case: Assigning tasks to channel members<br>Goal in Context: Owner assigns tasks to channel users and all users are able to view taskboard in channel<br>Scope: database, frontend<br>Level: Primary task<br>Preconditions: Authorised user must be a channel owner and specified user must be a channel member<br>Success End Conditions: Successfully assigned task on channel taskboard with notification sent<br>Failed End Conditions: Unable to add task to taskboard<br>Primary Actor: User<br>Trigger: Create task when authorized user clicks “assign task” button and hits “send” | Step 1.<br>User A clicks the taskboard >> assign task in the channel which opens panel with user selection tool and message box<br>Step 2.<br>User A selects User B from user selection box (which presents a list of channel members)<br>Step 3.<br>User A enters task description and hits “send”.<br>Step 4.<br>Task appears on taskboard >> view tasks page.<br>Step 5.<br>User B receives notification alerting them of a new task.<br>Step 6.<br>User A or B remove task when complete. |

**User C**

| User Story | Use Case | Main Success Scenario |
|---|---|---|
| As a team leader, I want to be able to share files so that group members can check and download them.<br>▪ File button at left of search bar at top<br>▪ User uploading files must be owners of channels<br>▪ User downloading files must be members or owners of channels | Use Case: Uploading and downloading files<br>Goal in Context: Owners uploading files that user in channels can see<br>Scope: database, frontend<br>Level: Primary task<br>Preconditions: Authorised user must be a channel owner<br>Success End Conditions: Successfully upload files that other channel members can access<br>Failed End Conditions: Failed to upload files<br>Primary Actor: User<br>Trigger: Users want to upload files when clicking “Upload files” button. | Step 1.<br>User A clicks “Files” and views all the files in the channel<br>Step 2.<br>User A clicks “Upload flies” to upload files from local machine.<br>Step 3.<br>User A upload files successfully.<br>Step 4.<br>User A receives confirmation that their file has been uploaded.<br>Step 5.<br>User B downloads files on their device. |

## Validation

After completing the use cases, we reached out to above 3 interviewees and asked them each: “If we implemented our use case solutions, would you be satisfied? If so, why and is there anything you think should be improved?”

### User A

With the ability to schedule meetings accordingly and have the option for all team members to vote on an agreed time, it significantly helps in the ease of organizing a group efficiently. A function to view a shared calendar will be extremely useful in coordinating larger groups particularly and would reduce time spent in manually organizing.

### User B

The feature to assign tasks will be a great practical addition to help delegating work and keep everyone's progress in check. This will be a very useful tool to help ease the effort of having to reach out individually to communicate and check up.

### User C

This is definitely a very helpful feature that would simplify the difficulty of trying to gather resources. By having shared access to all documents required for a task will make using teamwork/collaboration software a lot more enticing and easy! However, I would like the option of also removing files that I upload because they are sometimes the wrong file or there is a mistake in them.

User C (Sarah) pointed out that while we implemented functionality that would allow team leaders (channel owners) to successfully upload files letting other members in the channel successfully downloading the file, we did not implement any feature that would allow users to delete their files. Therefore an updated use case for User C is as follows:

| User Story | Use Case | Main Success Scenario |
|---|---|---|
| As a team leader, I want to be able to share files so that group members can check and download them. **I would also like the ability to remove these files.**<br>▪ File button at left of search bar at top<br>▪ User uploading files must be owners of channels<br>▪ User downloading files must be members or owners of channels<br>▪ **User removing files must be owner of channels**<br>▪ **Users cannot access and download deleted files.** | Use Case: Removing files that has already been uploaded<br>Goal in Context: Owners removing files that users can no longer access<br>Scope: database, frontend<br>Level: Primary task<br>Preconditions: Authorised user must be a channel owner<br>Success End Conditions: Successfully remove files that other channel members can not access<br>Failed End Conditions: Failed to remove files<br>Primary Actor: User<br>Trigger: Owners can click ‘remove’ on the file that they wish to delete. | Step 1.<br>User A clicks “Files” and views all the files in the channel<br>Step 2.<br>User A selects the file they want to remove.<br>Step 3.<br>User A clicks ‘remove’ and file should be deleted.<br>Step 4.<br>User A returns to ‘Files’ screen and no longer sees the file on the channel.<br>Step 5.<br>User B cannot download the removed file |

# Design

## Interface Design

**Table 1.** Interface Design for Meeting Scheduler

| Function Name & Description | HTTP Method | Parameters | Return Type | Exceptions |
|---|---|---|---|---|
| meeting/details<br>Given a Channel with ID channel_id that the authorised user is part of, provide basic details about all the meetings that have been scheduled by the channel owner. | GET | {token, channel_id} | {meeting_id, meeting_name, poll_option} | **InputError** when :<br>▪ channel_id is not valid<br>**AccessError** when:<br>▪ The user is not member or owner of the |

| | | | | |
|---|---|---|---|---|
| | | | | channel with channel_id |
| meeting/schedule<br>Given the meeting’s description, duration and time options, schedule a new meeting. Return a meeting name and poll_options that other uses can vote on. Note: Each meeting_id should have it's own unique ID. I.E. No meeting poll should share an ID with another meeting poll, even if that other meeting poll is in a different channel. | POST | {token, channel_id, meeting_desc, meeting_duration, meeting_time_options} | {meeting_id, meeting_name, poll_options} | **InputError** when any of:<br>▪ Meeting description is more than 1000 characters<br>▪ Meeting duration exceeds 24 hours<br>▪ More than 30 options selected<br>▪ No options selected<br>**AccessError** when any of:<br>▪ the authorised user has not joined the channel they are trying to schedule meeting in |
| meeting/vote<br>Authorised users in a channel can vote for their preferred meeting time option. This vote should be reflected on the meeting details board. | POST | {token, meeting_id, poll_option} | {user_vote} | **InputError** when any of:<br>▪ Meeting poll (based on ID) is invalid<br>▪ option_id is invalid<br>**AccessError** when any of:<br>the authorised user has not joined the channel they are trying to vote in |

**Table 2.** Interface Design for Taskboard

| Function Name & Description | HTTP Method | Parameters | Return Type | Exceptions |
|---|---|---|---|---|
| task/assign<br><br>Send a task from authorised_user to the channel specified by channel_id. Note: Each task should have it's own unique ID. I.E. No tasks should share an ID with another task, even if that other message is in a different channel. Task appears on taskboard with specified user's name and user is notified. | POST | {token, channel_id, u_id, task} | {task_id} | **InputError** when any of:<br>▪ Task description is more than 1000 characters<br>▪ u_id does not refer to a valid user<br><br>**AccessError** when any of:<br>▪ the authorised user is not an owner of the channel the task is sent to |
| tasks/view<br><br>Authorised users in channels have access to a taskboard which lets them view current tasks and who they've been assigned to. | GET | {token, channel_id} | {tasks} | **InputError** when any of:<br>▪ Channel_id is not a valid channel<br><br>**AccessError** when any of:<br>▪ the authorised user is not a member of channel with channel_id |
| task/remove<br><br>Remove an existing task assigned to an member. This can only be done by the channel owner. | DELETE | {token, task_id} | {} | **InputError** when any of:<br>▪ Task (based on ID) no longer exists<br><br>**AccessError** when any of:<br>▪ the authorised user is not an owner of the channel the task is sent to<br>▪ the authorized user is not the person the task was assigned to |

**Table 3.** Interface Design for File Sharing

| **Function Name & Description** | **HTTP Method** | **Parameters** | **Return Type** | **Exceptions** |
|---|---|---|---|---|
| file/details<br>Given a Channel with ID channel_id that the authorised user is part of, provide basic details about all the files in in the channel that the channel owner has uploaded. | GET | {token, channel_id} | {file_id, file_name, file} | **InputError** when any of:<br>▪ channel_id is not valid<br>**AccessError** when any of:<br>▪ The user is not member or owner of the channel with channel_id |
| file/upload<br>An authorised user of a channel can upload a file. The file can be a maximum of 5mb. Users should receive confirmation that their upload has been successful. | POST | {token, file, channel_id} | {is_upload_success} | **InputError** when any of:<br>▪ The file is more than 5 MB<br>▪ channel_id is invalid.<br>**AccessError** when any of:<br>▪ The authorised user is not an owner of given channel |
| file/download<br>An authorised user of a channel can download a file. Users should receive confirmation that their download has been successful. | GET | {token, channel_id, file_id} | {file, is_download_success} | **InputError** when any of:<br>▪ The file does not exist.<br>**AccessError** when any of:<br>▪ The authorised user is not a member from a valid channel |
| file/remove | DELETE | {token, file} | {} | **InputError** when any of: |

| | | | | |
|---|---|---|---|---|
| An authorised user of a channel can delete the file from their channel. This means this file can no longer be accessed or downloaded. No user confirmation is required. | | | | ▪ The file does not exist<br>**AccessError** when any of:<br>▪ The authorised user is not an owner the channel whose member can access to the file |

## Conceptual Modelling (State)

The following four diagrams are how we would expect our solutions to be implemented into Dreams.

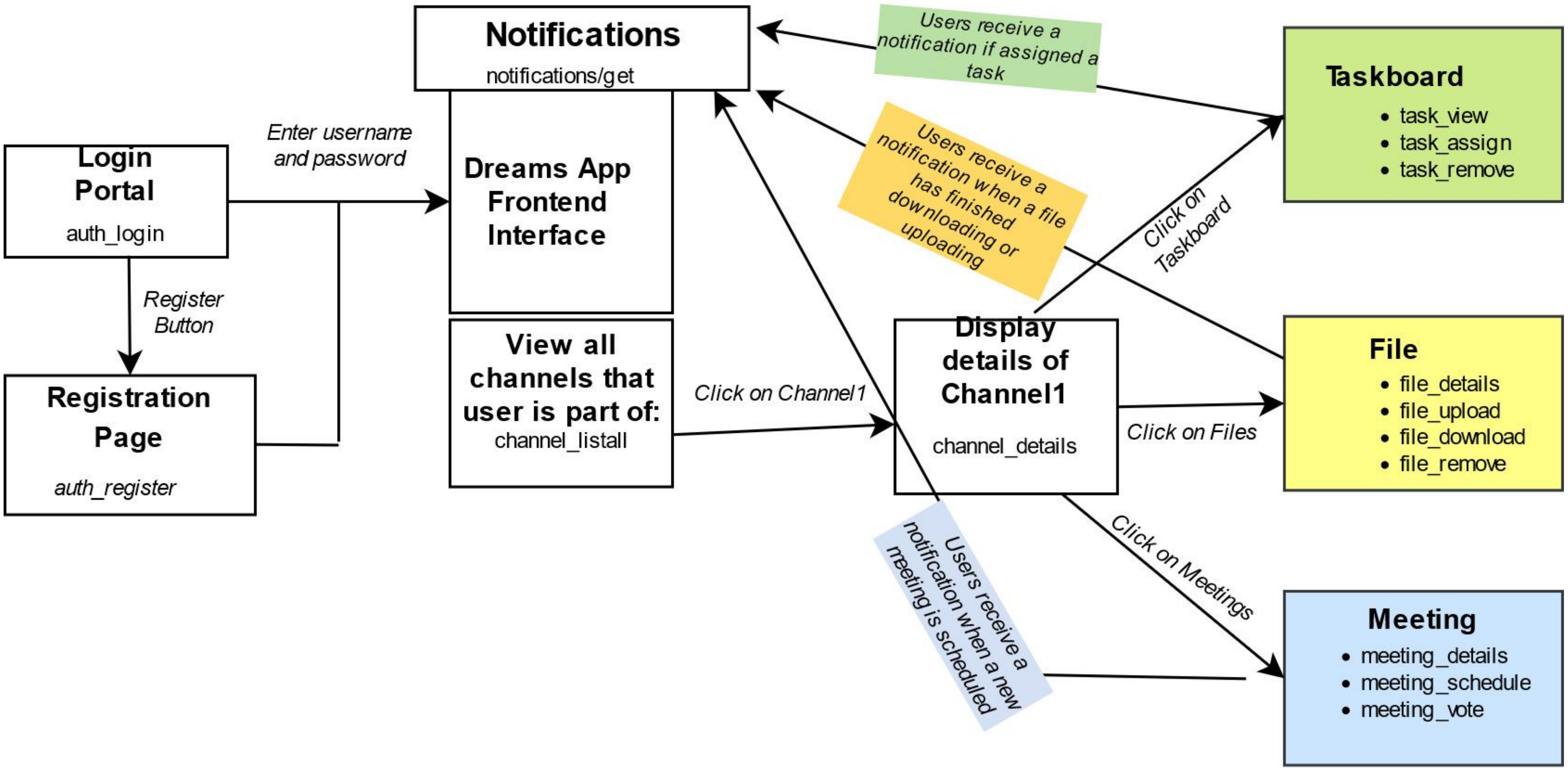
The first diagram shows how the state of our front-end would change. The arrows indicate how an average Dream's user already in a channel will access these features.  All of our new features are channel-specific, primarily focused on improving team communication and efficiency. A developer would see that each individual channel has their own taskboard, file and meeting functionality, and would have to ensure only authorised users can access each channel's aforementioned features. The developer designing the notifications functionality will also have to expand their dictionary to include the new taskboard, file and meeting features.

The following 3 diagrams are more in-depth look at our new features that will help backend developers understand what each function does, how it modifies the data and which authorised users can perform a certain feature. Each of our 3 features has a details/view function that would serve as the 'homepage' for that feature. The arrows between the function's highlights the data that is being stored and/or modified. Frontend designers will also benefit from the visual cues to understand how the Dreams should function.

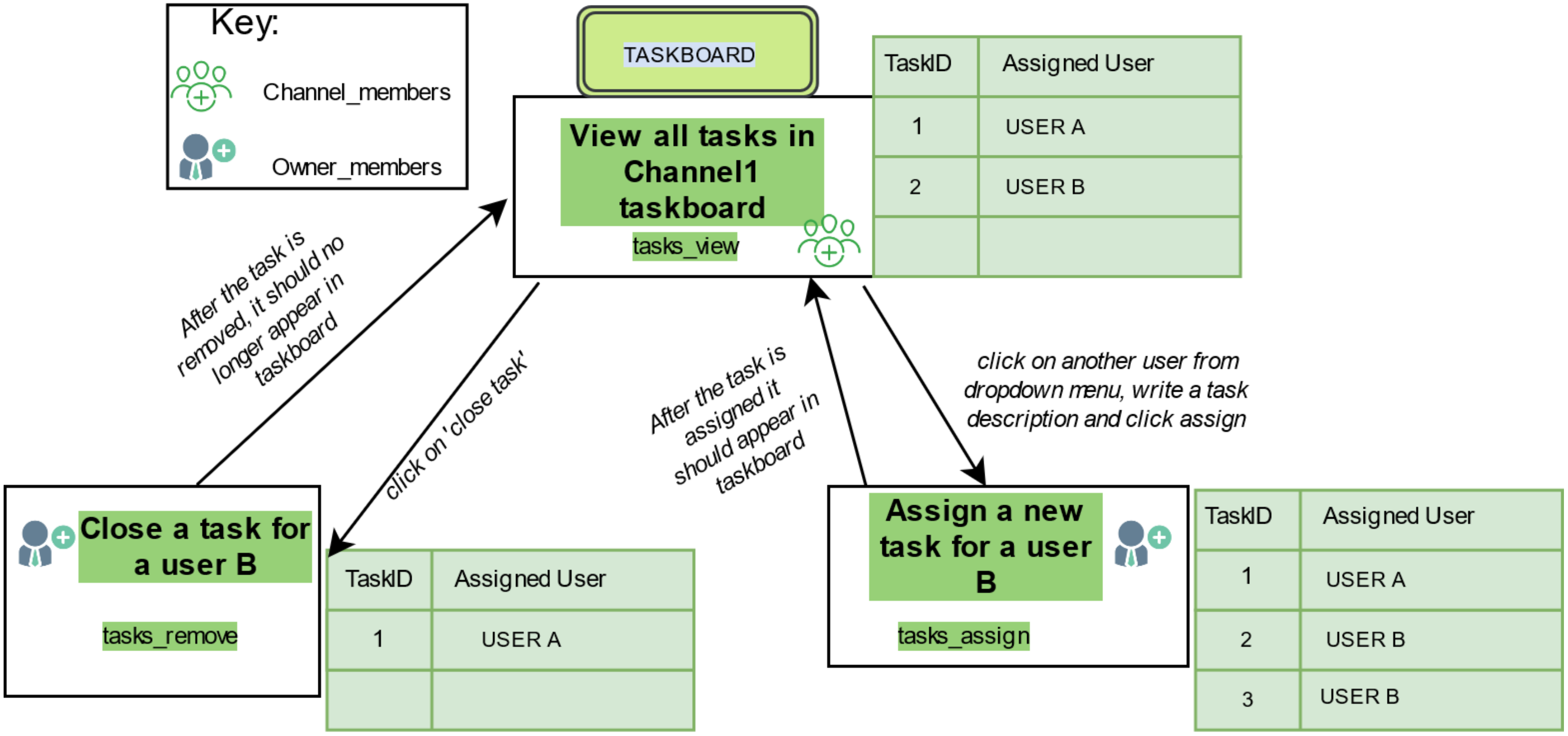
Development of conceptual models is a critical component of the software engineering design principles. In a robust and logical way, conceptual models helped classify the new features of the Dreams in relation to how the system works and underlying drivers and causal relationships among significant components.

They are helpful in clarifying the vision and management goals of stakeholders for Dreams which includes backend developers, frontend developers and users. It allows us to develop objectives for that can assess the effectiveness of the new feature implementation. It is also a much preferred communication model amongst software engineers who like visual cue over long paragraphs of text (sorry Hayden!).

## Conceptual Modelling (State)- Front End



## Conceptual Modelling (State)- Taskboard



Key:

Channel_members

Owner_members

TASKBOARD

**View all tasks in Channel1 taskboard**

tasks_view

| TaskID | Assigned User |
|---|---|
| 1 | USER A |
| 2 | USER B |
| | |

*After the task is removed, it should no longer appear in taskboard*

*click on 'close task'*

*After the task is assigned it should appear in taskboard*

*click on another user from dropdown menu, write a task description and click assign*

**Close a task for a user B**

tasks_remove

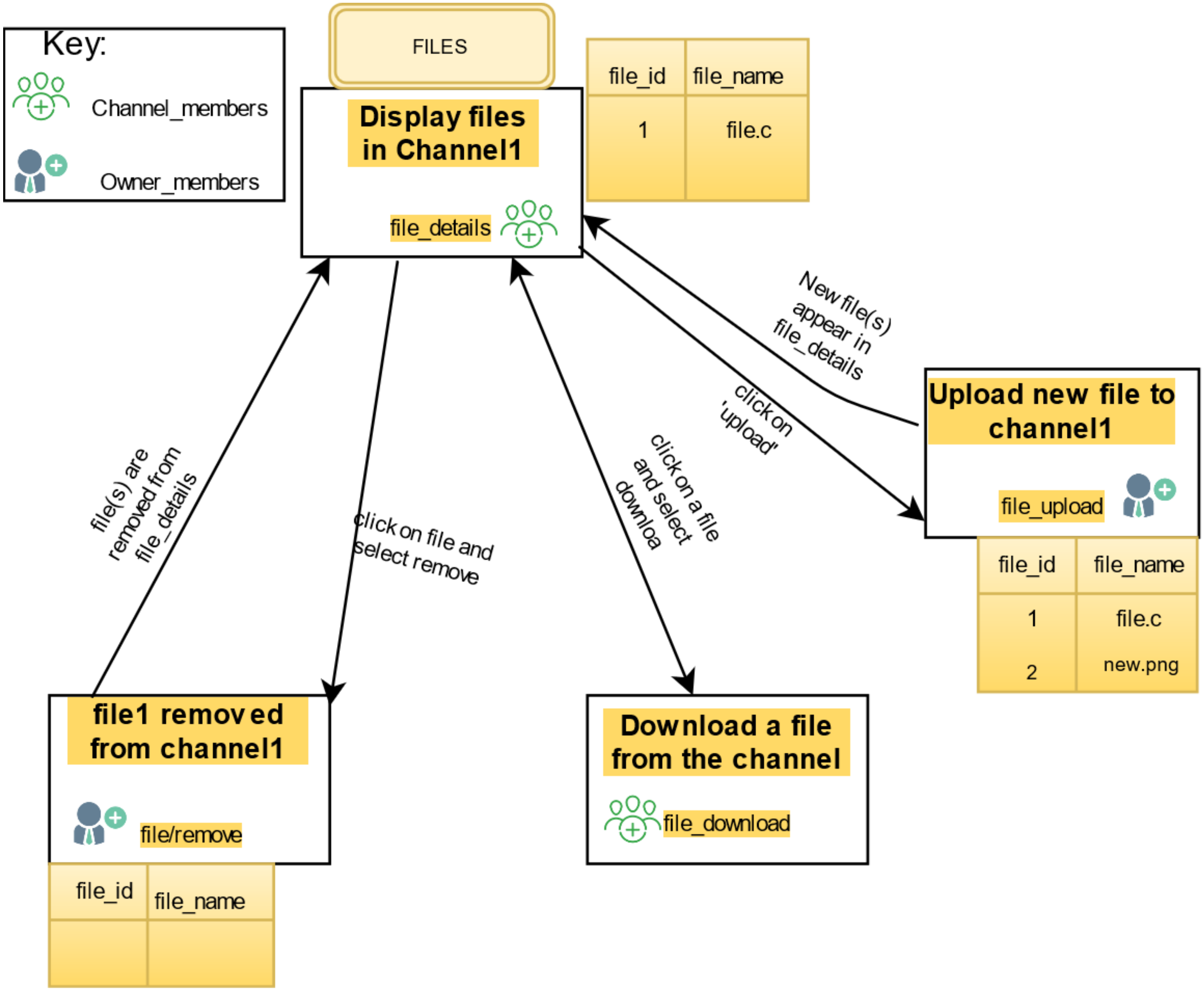
| TaskID | Assigned User |
|---|---|
| 1 | USER A |
| | |

**Assign a new task for a user B**

tasks_assign

| TaskID | Assigned User |
|---|---|
| 1 | USER A |
| 2 | USER B |
| 3 | USER B |

Conceptual Modelling (State)- Files



Key:

Channel_members

Owner_members

FILES

**Display files in Channel1**

file_details

| file_id | file_name |
|---|---|
| 1 | file.c |

New file(s) appear in file_details

click on 'upload'

**Upload new file to channel1**

file_upload

| file_id | file_name |
|---|---|
| 1 | file.c |
| 2 | new.png |

click on a file and select downloa

**Download a file from the channel**

file_download

click on file and select remove

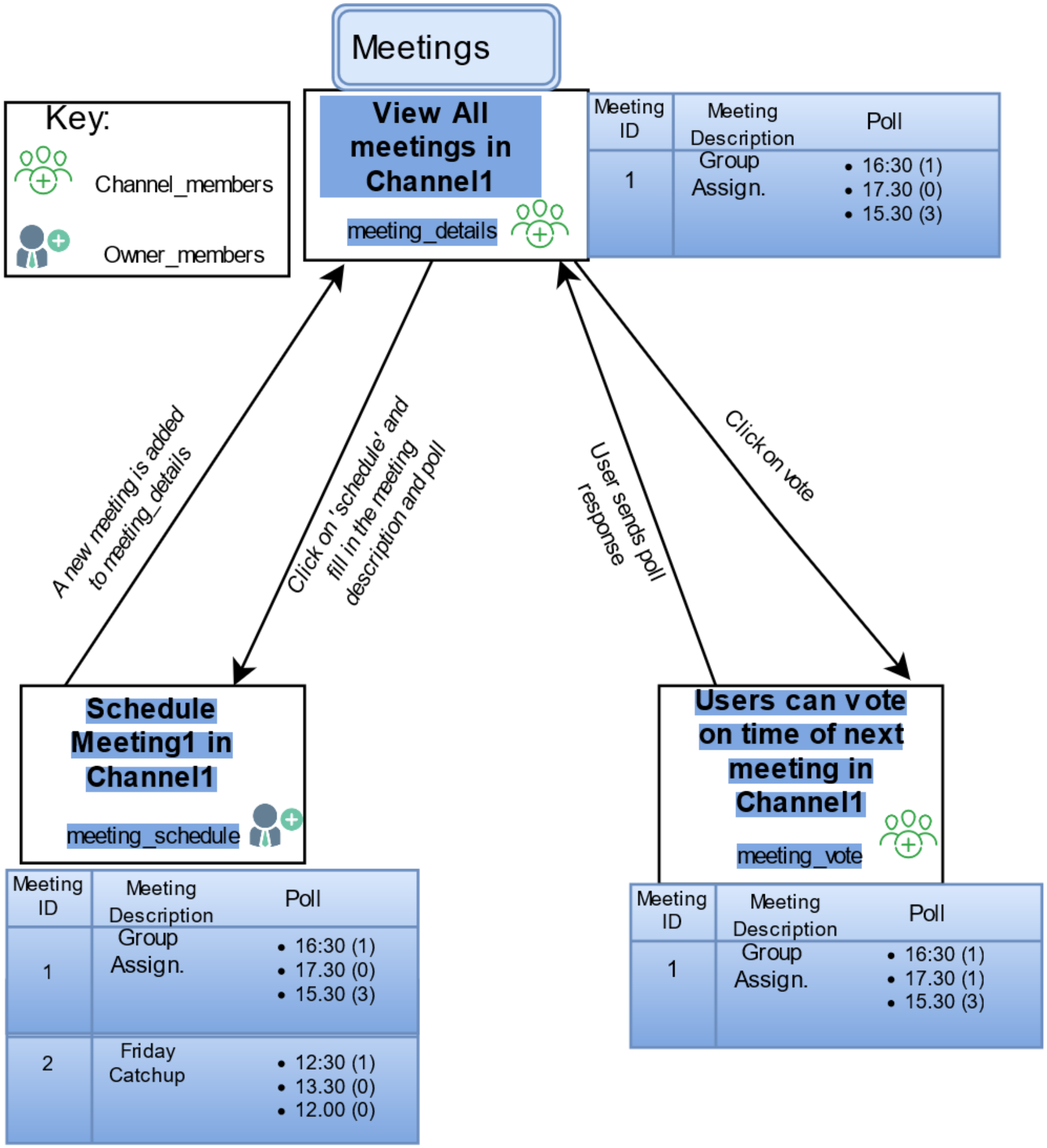
file(s) are removed from file_details

**file1 removed from channel1**

file/remove

| file_id | file_name |
|---|---|
| | |

Conceptual Modelling (State)- Meetings



Meetings

Key:

Channel_members

Owner_members

**View All meetings in Channel1**

meeting_details

| Meeting ID | Meeting Description | Poll |
|---|---|---|
| 1 | Group Assign. | • 16:30 (1)<br>• 17.30 (0)<br>• 15.30 (3) |

A new meeting is added to meeting_details

Click on 'schedule' and fill in the meeting description and poll

User sends poll response

Click on vote

**Schedule Meeting1 in Channel1**

meeting_schedule

| Meeting ID | Meeting Description | Poll |
|---|---|---|
| 1 | Group Assign. | • 16:30 (1)<br>• 17.30 (0)<br>• 15.30 (3) |
| 2 | Friday Catchup | • 12:30 (1)<br>• 13.30 (0)<br>• 12.00 (0) |

**Users can vote on time of next meeting in Channel1**

meeting_vote

| Meeting ID | Meeting Description | Poll |
|---|---|---|
| 1 | Group Assign. | • 16:30 (1)<br>• 17.30 (1)<br>• 15.30 (3) |