Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Piruthusan Arulnesan
Philip Dahlstrøm

Flate: exploring a collaborative platform in mathematics

Master's thesis in Informatics
Supervisor: Trond Aalberg
June 2021

NTNU
Norwegian University of
Science and Technology

Piruthusan Arulnesan
Philip Dahlstrøm

Flate: exploring a collaborative platform in mathematics

Master's thesis in Informatics
Supervisor: Trond Aalberg
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU

Kunnskap for en bedre verden

Abstract

More and more schools in Norway are opting to provide pupils with personal digital devices, such as computer tablets. The increased use of tablets in education provides opportunities to adapt concepts from groupware in learning, like shared text editors, to other contexts like collaboration in mathematics. This project aims to explore how to facilitate collaboration in a software platform using a shared workspace.

The main contribution of this work is a fully functioning prototype supporting collaboration in mathematics. It was developed by identifying common features and technologies in groupware and determining which of these features were necessary to facilitate collaboration in mathematics.

An evaluation of the platform was performed to identify collaborative patterns that occurred when pupils used the prototype in a test setting. A test on three pairs of pupils was executed to generate data, with observations and interviews. Observations were performed by using a monitoring system built for the testing and taking field notes of the conversation between the pupils. The system was developed to visualize interactions in the platform as user actions in a scatter plot. Semi-structured interviews with the pupils and the teacher overseeing the test on-site were used to elicit opinions regarding the collaboration. The data from the observations were qualitatively analyzed to identify collaborative patterns, and the data from the interviews were used to enhance the understanding of how the pupils worked together.

Results from the test show that the pupils collaborated in multiple ways, and the main patterns exhibited were categorized as *parallel*, *ping-pong*, and *singular action*. In the first collaboration pattern, the pupils interacted in parallel; in the second, their interactions were alternating; and in the third, only one pupil interacted with the shared workspace. In addition to these results, this thesis found that collaborative platforms often focus on transparency; all users are kept aware and updated regarding the state of the shared workspace. It also identified simultaneity when drawing lines, allowing users to interact with objects, and making users aware of each other's actions as the most significant features to facilitate collaboration in mathematics.

Sammendrag

Flere og flere skoler i Norge velger å tilby elever en personlig digital enhet, som ofte er nettbrett. Den økende bruken av nettbrett i utdanning gir mulighet for å tilpasse konsepter fra gruppevare, f.eks. fra delte tekstredigeringsverktøy, til andre kontekster som samarbeid innen matematikk. Denne oppgaven hadde som mål å tilrettelegge for samarbeid i en programvareplattform med en delt flate hvor elever kan løse matematiske oppgaver sammen.

Hovedbidraget fra dette arbeidet er en fullt fungerende prototype som er lagd for å støtte samarbeid i matematikk. Prototypen ble utviklet ved å først identifisere vanlig funksjonalitet og teknologi innen gruppevare, og så vurdere hva slags funksjonalitet som var nødvendig for å tilrettelegge for samarbeid innen matematikk.

En evaluering av plattformen ble gjort for å identifisere samarbeidsmønstre som dukket opp under bruk av elever i en test-setting. Testen ble gjennomført av tre par med elever. Observasjoner og intervjuer ble gjort for å generere datamateriale. Observasjonene ble gjort med et egenutviklet monitoreringssystem som visualiserte interaksjoner i plattformen. I tillegg ble det tatt feltnotater av samtalen mellom elevene. Semistrukturerte intervjuer med elevene, og læreren deres, ble brukt for å frembringe meninger om samarbeidet. Datamaterialet fra observasjonene ble kvalitativt analysert for å identifisere samarbeidsmønstre, og datamaterialet fra intervjuene ble brukt for å forbedre forståelsen av hvordan elevene jobbet sammen.

Resultatene fra testen viser at elevene samarbeidet på flere måter, og hovedmønstrene ble kategorisert som: *parallel*, *ping-pong*, og *singular action*. I det første samarbeidsmønsteret interagerte elevene parallelt; i det andre var deres interaksjoner alternerende; i det tredje var det bare én elev som interagerte med den delte flaten. I tillegg til disse resultatene, ble det funnet at samarbeidsplattformer ofte fokuserer på transparens, som kan bety at alle brukere blir gjort bevisste og oppdaterte om tilstanden til den delte flaten. Funksjoner som samtidighet når man tegner linjer, å tillate brukere å interagere med objekter, og å oppdatere brukere om hverandres handlinger ble identifisert som de viktigste funksjonene for samarbeid innen matematikk.

Preface

This thesis is written as a part of the authors' master's degrees in the Informatics study program and presents an implementation-focused software project conducted in autumn 2020 and spring 2021. This project was completed under the supervision of associate professor Trond Aalberg at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU).

The authors would like to thank Fredrik Sørum Andersen and Ida Dahl at *Neddy* for the initial help shaping the platform's idea and their continued support and interest in the project. Beate Horg at *Mattematikksenteret* has been an invaluable actor for this project by providing the mathematical tasks and giving advice on structuring the platform to facilitate learning. In addition, Beate assisted in recruiting Peder Vevelstad, the teacher who was willing to perform the final testing in this project with help from his pupils. Without the help from Peder, this project would not have been possible to complete in the planned way. Lastly, the authors would like to thank *Excited, senter for fremragende IT-utdanning* for providing tablets used for the development in this project.

Table of Contents

List of Figures

List of Tables

1 Introduction

"Fremme læring, ikke teste den."

This thesis presents a project where a collaborative software platform was designed and implemented to facilitate a test where pupils solved mathematical tasks together. To visualize interactions in the platform, a monitoring system was developed. Together with field notes and interviews, this monitoring system was used to identify and analyze the collaborative patterns that occurred during the test.

1.1 Problem description

In 2020, the Norwegian primary school curriculum was updated to focus more on digital competence. As a result, more and more schools are actively using computers such as tablets during classes [1][2]. The usage of tablets in primary schools allows pupils to learn, cooperate, collaborate, and explore in new ways using educational software and the internet. A pupil's physical location is potentially rendered insignificant. Exploring how teachers and pupils can collaborate in new ways is an interesting subject.

Mathematics has traditionally been a subject where pupils explore concepts and logical thinking while supervised. Working with tasks and solving problems have conventionally been done individually. This way of working is possibly due to how mathematics varies a lot in format and is often performed with pen and paper. Some tasks can be solved with text only, while others may require a combination of text, graphs, diagrams, drawings, and formulas. Other subjects, such as languages, history, and social studies, are more suited for utilizing collaborative text processing platforms such as Google Docs (Section 2.4.1). The varied format found in mathematics makes collaborative work challenging to support within a digital space.

With today's tablet usage in primary schools, it is interesting to explore how a collaborative software platform built to focus on mathematics would behave in an educational setting. Due to the varied input types different mathematical tasks and exercises often demand, such a platform would ideally be based on point- and line inputs in a shared workspace rather than text. One of the significant advantages tablets have compared to other personal computers is their integration with styluses, which helps make drawing feel intuitive and natural for pupils. This thesis will discuss the requirements, design, implementation, and validation of such a platform.

1.2 Motivation

The motivation for creating a collaborative platform for solving mathematical tasks was to encourage and support collaborative learning amongst pupils regardless of physical location. After informal conversations with Neddy AS[1] and Mattematikksenteret[2], it became apparent that a platform aimed to encourage learning was sought after within the educational environment. They claimed many of the existing and widely used software platforms for mathematics in schools today were mainly focused on validating and testing pupils' skills within different topics. Neddy expressed that an application with a flexible shared workspace and inter-user communication, where pupils were encouraged to explore, could be a valuable addition to today's learning methods.

[1] https://www.neddy.no/
[2] https://www.matematikksenteret.no/

1.3 Research questions

The project's overall goal was to implement a collaborative software platform with a shared workspace, where pupils could solve mathematical tasks together. This thesis explores the design, implementation, and evaluation of this platform, with the following questions:

RQ1 What are common features and technologies in collaborative software platforms?

RQ2 Which features are necessary to facilitate collaboration in mathematics?

RQ3 What patterns of collaboration occur when using the platform?

1.4 Research method

Figure 1 shows an overview of the research method for this project. The conceptual framework was the result of the initial literature review, and acted as the foundation for the project and the research questions. Additionally, the design and implementation of the platform include features and attributes from it. Awareness is central within collaborative software, and enabling pupils to be aware of each other and their actions would presumably allow them to collaborate in different ways. Identifying collaborative patterns in usage was therefore deemed interesting.

The *design and creation* strategy [3, p. 108] was chosen for this project, as it would allow for the developed platform to act as a vehicle to identify collaborative patterns. A test scenario using this platform was then designed to gather data. The test was performed with pairs of pupils solving multiple tasks under supervision from their teacher. To *observe* [3, p. 202] the collaboration, a monitoring system was developed for the test and used in conjunction with field notes. The system visualized interactions in the platform. In addition to the observations, *interviews* [3, p. 186] were held with each group of pupils. An interview with the teacher was held after all the groups had finished. All of the interviews conducted were *semi-structured* [3, p. 188].

The data was analyzed *qualitatively* using an inductive approach [3, p. 269]. The primary data were the visualizations from the monitoring system and the field notes from the observations. In addition, the data from the interviews were used to assist these visualizations where applicable. This approach led to descriptions and analysis of the different patterns that occurred and the different technologies within the framework that enabled them.

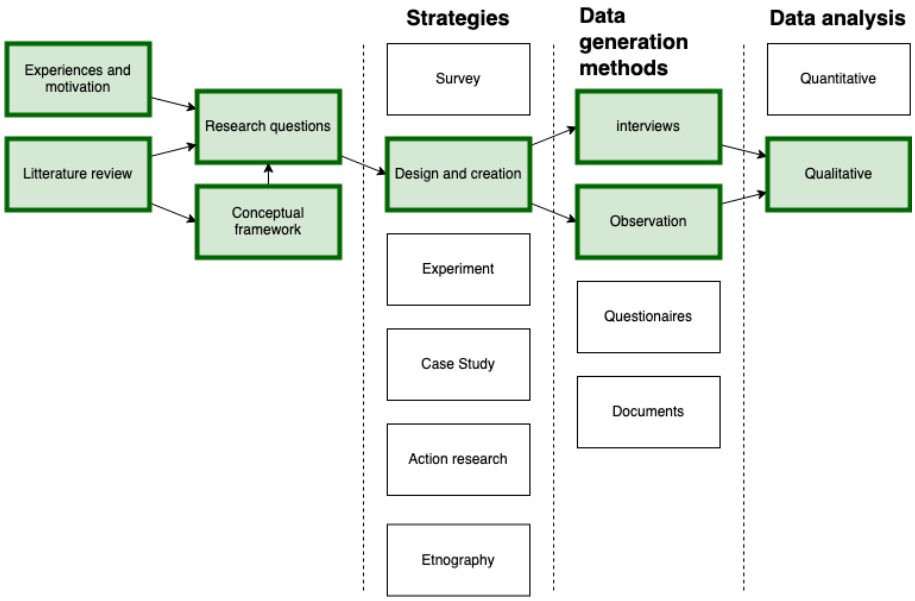


Figure 1: Research method overview

1.5 Platform description

"Flate," the Norwegian word for surface, is the working name of the platform made throughout this project. Flate is a collaborative software platform where users can explore and solve mathematical tasks. The application's core is a shared workspace: an infinitely expandable virtual whiteboard, where users can draw and write together using touch input in real-time. Users create lines that are streamed to other users as they are drawn. After a user completes their lines, the platform converts the lines into objects which all users in the same session can manipulate. The manipulations users can perform include: moving, deleting, commenting, grouping several objects, and more.

When opening the app, the user is presented with a list of mathematical tasks they can solve. After selecting a task, the user enters a lobby. A user can create a room from this lobby and invite other users, which creates a session. Upon joining a session, an audio link between the connected users is established. Through this open audio link, all users can communicate directly with each other. When users are ready, they can start the session, which sends them to the whiteboard. Some tasks may contain images, and these are added to the whiteboard as accessible pre-rendered objects. These image-based objects can then be interacted with in the whiteboard in the same way as line-based objects. The shared whiteboard is unbounded in size, allowing users to move throughout the shared workspace as they please. Having an expandable whiteboard allows users to structure their work however they please. Other users' positions are displayed within the interface using avatars, enabling all users to be aware of their collaborators' positions. Both written and verbal communication is supported in the platform.

The goal of this platform is to create a stand-alone experience where simultaneous collaboration feels effortless and intuitive.

1.6 Thesis structure

A description of the sections in this thesis is provided here:

2. **Background**
   This section presents central theories in the field of Computer-Supported Cooperative Work (CSCW) as a framework, focusing on groupware and awareness. Two relevant groupware platforms for this project are also presented and reviewed in this section, followed by a discussion on Computer-Supported Collaborative Learning (CSCL).

3. **Designing a real-time groupware platform**
   The design of the platform is presented in this section. This design is based on the framework presented in the Background section, where some design decisions are made to fit the project's context.

4. **Implementation**
   The section presents the development process and how the core functionality of the platform is implemented. The features within the platform are based on the design from the previous section. Both the front-end and the back-end are presented, followed by the local processes supporting drawing and interaction. The last subsection presents how actions are transmitted and replicated between devices.

5. **Method**
   This section presents the research strategy, data generation and evaluation, and the chosen approach for analyzing the data. The test case and how the data was generated using observations is described within the data generation subsection.

6. **Results**
   In this section, the results from the data generation are presented. It includes test summaries and interviews for each group, and the interview with the teacher. The section also presents and analyses the collaborative patterns observed using the monitoring system and discusses how the different features in the platform, which are based on the framework from the Background, helped facilitate these patterns.

7. **Discussion**
   This section discusses the findings for each research question and evaluates the project. It also presents potential future work for the platform, and considerations and contributions of the project.

8. **Conclusion**
   In this section, the findings and contributions presented in this thesis are summarized and concluded, followed by recommendations for future work on the research topic.

2 Background

This section presents common attributes and characteristics of groupware software and includes an elaboration on the subgroup of groupware that supports simultaneous activity called real-time groupware. It also defines and discusses awareness within this type of software. Section 2.4 presents two groupware platforms and discuss how the attributes and characteristics of groupware are present within these platforms. Section 2.5 presents Computer-Supported Collaborative Learning (CSCL), and discusses its relevance to the project.

2.1 Groupware

Computer-Supported Cooperative Work (CSCW) is a term for technologies of computer hardware, software, services, and techniques that support people working together in groups of different sizes [4]. A term that is frequently used synonymously with CSCW is *groupware*. Ellis et al. [5] suggest that groupware can be defined as the application class, both for smaller groups and larger organizations, that arises from merging computers, large information bases, and communication technology. Specifically, they define groupware as:

> *Computer-based systems that support groups of people engaged in a common task (goal) and that provide an interface to a shared environment.*

This definition does not specify that users have to be active simultaneously. Groupware that specifically support simultaneous activity is called *real-time groupware* (see Section 2.2).

2.1.1 Groupware spectrum

As software systems support common tasks and shared environments to varying degrees, one can think of a groupware spectrum with different systems at different points on the spectrum. In Figure 2, this spectrum is illustrated in two dimensions. These dimensions derive from the definition of groupware and aim to visualize to which degree a *common task* and *shared environment* are present within a platform.

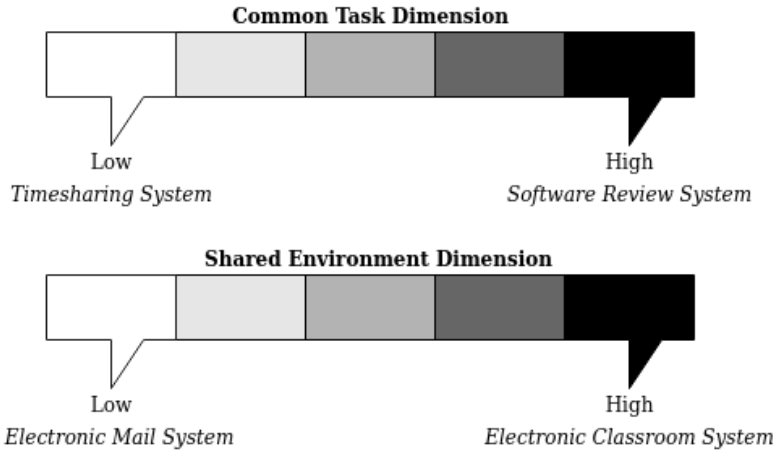


Figure 2: The groupware spectrum

In the groupware spectrum, a system on the far left scores low in the groupware spectrum, and conversely, a system on the far right scores high in the spectrum. Examples of systems are placed on the spectrum in each dimension.

A conventional timesharing system where many users perform separate and independent tasks concurrently is usually low (left) on the groupware spectrum in the common task dimension. A software

review system that electronically allows a group of users to evaluate an entity with real-time interaction would be placed high (right) on the groupware spectrum. E-mail would be placed low on the groupware spectrum in the shared environment dimension as it gives few environmental cues, meaning the environment does not give cues to the user about, for instance, what the other users are doing. An electronic classroom system that emulates a traditional classroom and allows instructors to present online lectures to students at remote computers would be placed high on the groupware spectrum. This is because such a system, depending on the design, can give cues to users about what is happening around them.

2.1.2 Groupware time space matrix

Groupware can be designed to support cooperation within a face-to-face group or a group working together remotely. It can also be designed to facilitate communication and collaboration in real-time interaction and non-real-time interaction. A two-by-two matrix can represent the geographic dispersion (distributed actors vs. non-distributed actors) and time dispersion (synchronous vs. asynchronous communication). The work of Johansen [6] inspires this taxonomy and often categorizes CSCW technologies [7, p. 13].

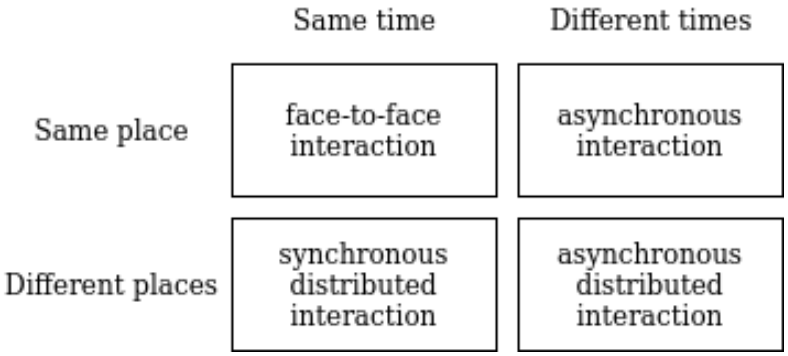


Figure 3: Groupware Time Space Matrix

In Figure 3, the time dispersion is represented by the columns *same time* and *different times*, and the geographic dispersion by the rows *same place* and *different places*.

Video conferencing solutions[3] would be placed in the lower-left cell. An interactive whiteboard, like the SMART Board[4], would be placed in the upper-left cell. A physical bulletin board would be placed in the upper-right cell. Systems do not have to belong to only one cell, and an online text editor like Google Docs, described in Section 2.4.1, could be placed in the lower-right and the lower-left cell.

2.1.3 Alternative groupware time space matrix

Carstensen and Schmidt [7, p. 17] proposed an alternative to Johansen's taxonomy as: "*... it is not very useful for describing most of today's existing CSCW systems since these have different facilities falling in different boxes.*" They suggested categorizing systems in a two-by-two matrix where the rows separate tightly and loosely coupled interaction among collaborators. The columns separate between seeing the computer as a medium or as a regulator of interaction. This taxonomy is illustrated in Figure 4.

[3]Examples of this include, but are not limited to, Zoom, Google Meet, and Microsoft Teams.
[4]https://www.smarttech.com/en/products/education-displays/smart-board-800

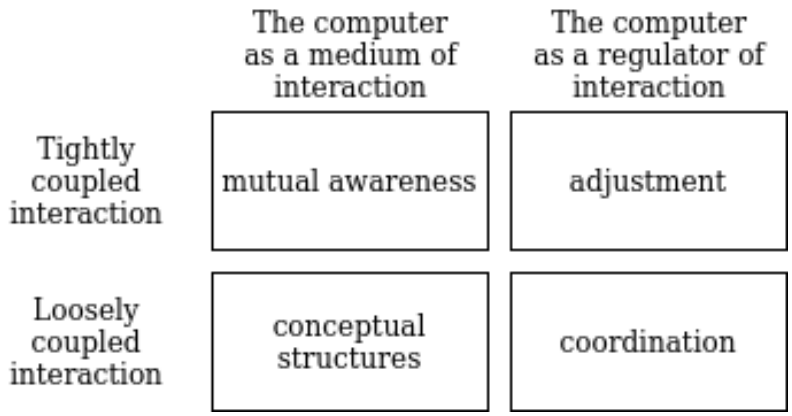


Figure 4: An alternative to the Groupware Time Space Matrix

Imagine moving a dining table set with friends, which consists of a table and some chairs. Moving the chairs can be done individually and requires little coordination. This interaction would be *loosely coupled*. However, moving the table is a joint effort, which requires additional coordination (e.g., agreeing on the side of the table and where to move it). This interaction would therefore be *tightly coupled*.

When using the computer as a medium of interaction, tightly coupled interaction requires that the actors are mutually aware. For example, when working with others in real-time in the same document, Google Docs would be placed in this cell, as users are aware of each other's activity. Loosely coupled interaction when using the computer as a medium requires that the different actors agree on the concepts in the system. For instance, in Wiki software systems, users have to agree on the concept related to the software. The editing of pages is usually done individually, and this would therefore place this type of software in the lower-left cell.

When using the computer as a regulator of interaction, an example of a system that facilitates tightly coupled interaction would be a Kanban board[5] like Trello.This is because Trello is used to regulate activities outside of the system itself, and the actors use the system to adjust these activities. For loosely coupled interaction, an example could be a shopping list application where a list is shared between two or more people.

It is important to note that the two dimensions in the matrix serve as continuums and that one application can serve multiple roles that cut across multiple areas in the matrix space.

[5]https://www.atlassian.com/agile/kanban/boards

2.2 Real-time groupware

Groupware that supports simultaneous activity is called *real-time groupware*. Real-time *distributed* groupware allows geographically distributed users to interact with each other. Many real-time groupware systems provide a bounded space where people can see and manipulate artifacts related to their activities [8, p. 414].

The invocation of a groupware system is usually called a *session*. The group of users in a session is called *participants*. Each participant in a session is provided an interface to a *shared context*. The time the system needs for the actions of a user to be reflected upon within their own interface is called *response time*. The time needed to propagate and replicate one user's action onto the other users' interfaces is called *notification time*.

Ellis and Gibbs [9, p. 399] use the following attributes to characterize real-time groupware systems:

- *Highly interactive*: The response times must be short.
- *Real-time*: The notification time must be short and comparable to the response time.
- *Distributed*: Assumes that participants are not all connected to the same machine.
- *Volatile*: The participants are free to enter and leave a session.
- *Ad hoc*: The information accessed by the participants can not be told *a priori*.
- *Focused*: The high degree of access conflict as participants work on and modify the same data.
- *External channel*: The participants are often connected by an external channel such as an audio or video link.

One example of a real-time groupware system is the online collaborative whiteboard platform called Miro. This system is discussed further in Section 2.4.2.

Methods that aim to ensure correct results of concurrent actions in multi-user softwares are collectively called concurrency control. Ellis and Gibbs' paper [9, p. 400] emphasizes that concurrency control is needed within groupware systems to help resolve conflicting actions (e.g., two users moving and deleting the same object) between participants and to enable users to perform tightly coupled activities. Below are some of the issues the paper found that are related to concurrency control:

- *WYSIWIS*: Ellis and Gibbs note that having a What You See Is What I See (WYSIWIS) interface is necessary to maintaining group focus. The cohesiveness is quickly lost if each user sees a slightly different or outdated version. The response- and notification times have to be as short as possible to ensure the interface always reflects the latest actions.
- *Replication*: The short response times puts high demands on groupware systems, and because of this, the data state is *replicated* for each participant locally. Many expensive operations can this way be done locally. If an object is not replicated, even simple operations require communication between the users, resulting in degrading the response time for each user.

One approach to solving concurrency control is *locking*. This means that the data is locked before it is modified, preventing other users from performing actions on the same data. Ellis and Gibbs note that there are several techniques to help decrease the probability that a user will request a lock on data that is already locked [9, p. 401]. One of the techniques mentioned in the paper is to provide the participants with *visual indicators* of locked resources. The papers note three main problems with locking:

1. *Overhead*: A degradation in response time because of the overhead achieved when requesting and obtaining the lock on an object.
2. *Granularity*: The level of granularity is not always clear when implementing locking. Ellis and Gibbs' paper illustrates this with an example using a text editor. Is it the enclosing paragraph, the sentence, the word, or the character that should be locked when a user inserts a character in the middle of a sentence?

3. *Request and release*: This problem is related to determining when locks on objects should be requested and released. Expanding on the text editor example, when should the lock be requested? Is it when the cursor is moved to the sentence or when the character key is pressed?

2.2.1 Concurrency

Real-time groupware systems always have an element of concurrency challenges. Replicating the data state locally for each participant has been identified as appropriate for real-time groupware systems with shared workspaces [9, p. 400][10, p. 207]. Whenever a user performs an action, their local data state is immediately updated, meaning the system has a short response time as presented earlier in Section 2.2. This introduces an inconsistency with the states of the other replicas [11, p. 195]. The states will be consistent again when the user action, sent asynchronously, has been received and handled by the other replicas. So, one can think of user actions going through several stages. For instance, Greenberg and Marwood [10, p. 208] present these stages: creation, local execution, transmission, reception, and remote execution. When multiple users interact with the system, actions interleave and get executed out of order at different sites. This can lead to interference and inconsistencies in the data states of the replicas.

Concurrency control is the activity of coordinating potentially interfering user actions that occur in parallel [10, p. 208]. The usability of a method for concurrency management relies both on its ability to maintain consistency among the replicas of a shared workspace and on the production of meaningful results that meet users' expectations [12, p. 288]. When discussing methods, its level of "optimism" refers to how strict the method is with the execution order of user actions. It differs slightly from method to method, but the basic idea is:

- Non-optimistic methods prevent a user action from being executed if the previous action has not yet been executed
- Optimistic methods allow user actions to be executed out of order and detect and repair inconsistencies that occur because of this

Examples of classical approaches for non-interactive computer systems, like distributed databases, are *serialization* and *locking* methods. Serialization algorithms work by either synchronizing user actions so that they are executed serially across the entire system or by repairing effects of out-of-order user action to give the illusion that they have been executed serially [10, p. 208]. Locking is a method that works by users gaining privileged access to some object for a length of time [10, p. 209]. Typically, a user will request a lock on an object, and if no one else has it, the user gains access. If someone else has the lock, the user requesting access will be denied. When a user who has the lock on an object no longer needs it, the object is released. More information about these two methods, including the specificity surrounding their optimism levels, can be found in Greenberg and Marwood's paper [10].

The classical approaches to concurrency control assume that computers can tolerate the delays associated with non-optimistic serialization and locking or that they can accept the local inconsistencies that could occur with their optimistic counterparts [10, p. 210]. Distributed systems can use concurrency control methods that do not consider that people are viewing a shared workspace, which is something real-time groupware can not. People can be more or less tolerant of problems related to concurrency in real-time groupware, and the effect of how these problems are presented in the shared workspace are individual.

Google Docs seems to use algorithms based on an optimistic replication technique called Operational Transform (OT) [13, p. 1] [14]. Each user operates on their own replica of the shared workspace, and any change done by a user is immediately propagated to other users. The basic idea of the approach is to transform an operation done by a user in accordance with a previous operation so that the shared workspace can achieve the correct data state. An alternative optimistic replication technique is Conflict Free Data Types (CRDT) [15, p. 150]. It is a class of data types that allow replicas to be modified and can guarantee convergence to the correct data state after updates without coordination [15, p. 150]. CRDTs are either operation-based, where updates are propagated as operations and

executed on every replica, or state-based, where updates propagate as full local states and merge on every replica. Operation-based CRDT and OT are similar in how they solve concurrency. The difference, however, is that the operations in CRDTs are assumed to be commutative, while in OT, the incoming operation is transformed according to a previous operation.

2.3 Awareness

In CSCW, *awareness* is an *understanding of the activities of others*, which provides a *context for your activity* [16, p. 107]. It is a fundamental concept and refers to knowing who is in the proximity and what activities are occurring [17]. The term has been a challenge for the CSCW community to define. The most important thing to note is that it describes a user's internal knowing and understanding of a situation, including other users and the environment, and that it is gained through subtle practices and interpreting information [18, p. 432]. This information partly exists in the system the user is operating in, and it is also partly provided by awareness technology in the system.

Information about awareness can be passively collected and distributed in the same shared workspace as the object of collaboration [16]. The study conducted by Dourish and Belloti [16] suggests that awareness information provided and exploited passively through the shared workspace allows users to move smoothly between close and loose collaboration. This also allows them to assign and coordinate work dynamically. In addition, it suggests that CSCW systems supporting the aforementioned method provide effective support for collaboration by using an approach called *shared feedback*.

The shared feedback approach is when the interface presents feedback on all individual users' activities within the shared workspace. This approach provides low overheads for both the providers and consumers of awareness information and the availability of information is as-and-when needed as a context for individual activities. Dourish and Belloti [16] performed a case study on a group of three designers using the ShrEdit[6] editor and had them linked via video and audio. Their paper noted the informal channel of verbal communication as important for supporting a system that required flexibility. From the case study, the paper suggests multiple benefits of using the shared feedback approach:

- Individuals have the opportunity to monitor each other's activities peripherally and comment on them. They can both communicate their activities and provide others with the opportunity to communicate on the activity or observe consequences for their own actions.
- Individuals can explicitly tailor their contributions knowing that others can see them, conveying information and soliciting responses via the shared workspace or other communication channels.
- The group can amongst themselves assign and reassign roles through fluid negotiation.

2.3.1 Workspace awareness

One subgroup of awareness in shared spaces is *workspace awareness*. Gutwin et al. define workspace awareness as the up-to-the-moment knowledge about other's interaction within the environment in which a task is performed that is afforded by person-to-person interaction in a shared workplace [19]. The paper Gutwin and Greenberg published in 2002 [8] provides a conceptual framework of three parts. It examines the workspace awareness concept and provides designers with help for understanding the concept when building awareness support in groupware. Even though the conceptual framework uses, among other things, observations from physical shared workspaces, it can be used to identify necessary features for groupware systems supporting shared workspaces.

The first part of the framework contains the different types of information that constitute workspace awareness. This awareness is made up of many kinds of knowledge, and the framework divides the concept into components. It presents that people usually keep a certain set of information to track in all kinds of collaborative work, either consciously or unconsciously. When working in a shared space, the set of information people usually keep consists of the elements that answer:

[6] A synchronous, multi-user text editor that runs on a network of Apple Macintoshes

- *Who*: Who they are working with
- *What*: What the other people are doing
- *Where*: Where different events are happening
- *When*: When the different events are happening
- *How*: How the different events occur

Given these categories, Gutwin and Greenberg identify the core elements that constitute workspace awareness. These elements are further divided into elements related to the present and elements related to the past. The two types of elements are shown in Figure 5 and Figure 6, respectively.

Category	Element	Specific questions
Who	Presence	Is anyone in the workspace?
	Identity	Who is participating? Who is that?
	Authorship	Who is doing that?
What	Action	What are they doing?
	Intention	What goal is that action part of?
	Artifact	What object are they working on?
Where	Location	Where are they working?
	Gaze	Where are they looking?
	View	Where can they see?
	Reach	Where can they reach?

Figure 5: Elements of workspace awareness related to the present

The elements in Figure 5 and Figure 6 are commonsense phenomenons that deal with interactions that happen between the person and the environment. For instance, in Figure 5, the elements *awareness*, *intention*, and *artifact* refer to the understanding of what another participant is doing on what object, at a high level of granularity or a general level.

Category	Element	Specific questions
How	Action history	How did that operation happen?
	Artifact history	How did this artifact come to be in this state?
When	Event history	When did that event happen?
Who (past)	Presence history	Who was here, and when?
Where (past)	Location history	Where has a person been?
What (past)	Action history	What has a person been doing?

Figure 6: Elements of workspace awareness related to the past

In the elements related to the past, *action history* and *artifact history* refer to the details of events that have occurred, while *event history* refers to when events happened. The remaining elements deal with the historical aspects of the elements related to the present.

Workspace knowledge will consist of these elements in some combination. However, when designing groupware systems, one should not aim to include support for all of these elements equally. Two factors are paramount in deciding how to include each element:

1. The degree of interaction between the participants indicates the level of granularity the information should include in the activity

2. How often the information changes indicates how often the interface needs to be updated

Gutwin and Greenberg illustrate that some situations do not require explicit support as certain elements never change. For instance, there is little need for the system to gather and distribute location information if the participants are constantly assigned to specific areas of the shared workspace.

The elements presented in the first part of the framework provide a high-level organization of workspace awareness, and Gutwin and Greenberg remark that they should act as a starting point for thinking about the awareness requirements of particular task situations.

The second part of the framework describes the mechanisms people use to gather workspace awareness information from the workspace environment. This means that the mechanisms described are how people find the answers to the *who*, *what*, *where*, *when*, and *how* listed in Figure 5 and Figure 6. These are described to make it easier to design groupware that presents awareness information to make the continuation of the workspace awareness easy and simple to understand. Gutwin and Greenberg describe three mechanisms to gather workspace awareness information [8, p. 422]: *consequential communication*, *feedthrough* and *intentional communication*.

- *Consequential communication*: This is transferred information and emerges as a consequence of a person's activity within an environment. The producer of this information does not intentionally undertake actions to inform other people. The people that perceive the information pick up the available information.
- *Feedthrough*: Manipulated artifacts in the environment give off information, and the feedback given to the person performing the action can also be observed by others to gather information.
- *Intentional communication*: Use verbal communication, the usual form of communication, and the exchanges to pick up awareness information. This can be done in three ways:
    1. Information is gathered by simply stating where they are working and what they are working with.
    2. Information is gathered by people overhearing others' conversations.
    3. Information is gathered by listening to the running commentary people produce while performing their actions (spoken to no one in particular).

The third part of the framework describes how workspace awareness is used in collaboration. It provides a basic set of collaborative activities that can be used to analyze work situations. The five activities the framework provides are [8, p. 425]: *management of coupling*, *simplification of communication*, *coordination of action*, *anticipation*, and *assistance* . Below, a summary of the benefits of these activities in workspace awareness is presented [8, p. 432] :

- *Management of coupling*: People keep track of others' activities when they engage in loosely couple collaboration to determine the appropriate time to initiate tightly coupled collaboration. Workspace awareness assists people in noticing and managing transitions between individual and shared work.
- *Simplification of communication*: Workspace awareness allows people to use the shared workspace and artifacts as conversational props that let people mix verbal and visual communication. It includes using mechanisms of deictic references[7], demonstrations, manifesting actions as a replacement for verbal communication, and visual evidence of understandings.
- *Coordination of action*: Coordination can be accomplished by either explicit communication or less explicit, brought about by the shared objects in the workspace. The latter is enabled by workspace awareness. Workspace awareness can also be useful in the division of labor and planning and replanning the activity.

[7] The practice of pointing or gesturing to indicate a noun, such as "this", "that", "here", or "there"

- *Anticipation*: Workspace awareness allows people to predict others' actions and activity at several time scales. This allows them to replan movements based on the anticipated action. Without the up-to-the-moment knowledge of workspace awareness, this would be difficult to accommodate.
- *Assistance*: Workspace awareness assists people in understanding the contexts where help is to be provided. In order to assist someone, one needs to know what they are doing, what the goal of the action is, and the context in which they are doing it. By being aware of these things, one can assess the situation and provide the appropriate assistance.

2.4 Relevant collaborative software platforms

This section presents two collaborative software platforms that handle simultaneous cooperation in different contexts and settings. It also presents a review of the software using the relevant models and frameworks presented in Section 2.1 and Section 2.3.

2.4.1 Google Docs

Google Docs [20] is an online word processing software that allows users to create, share, and collaborate on writing documents in real-time. Google Docs has a similar user interface and feature-set as Microsoft Word, which has been the most commonly used word processing software for Computer Assisted Writing (CAW), with additional features which handle real-time inter-user interaction and document sharing in the web-browser [21]. A user can create a new document and give other users access to it. Once a user has been granted access to a document, they can read and write in the document together with the original creator, depending on the type of access given. Other users currently viewing the document are represented as colored avatars in the top-right of the software, and their cursor is shown in the editor with the same color as the avatar. When other users are writing, their names are displayed above their cursor. One can be transported to other users' cursor placement by clicking on their avatar.

In addition to this, the software gives users the ability to mark an area in the text editor and create a comment thread. Users can tag other users in a comment in this thread, and a comment thread can be resolved when done. Users can also suggest edits in the editor, which can be approved or rejected.

2.4.2 Miro

Miro [22] is an online collaborative whiteboard software where users can collaborate in a shared surface. It provides users with the possibility to choose from pre-built templates or the option to create their own. The software has a board with an infinite canvas and different tools like sticky notes, a freeform pen, shapes, and arrows. A user can invite other users to their board, which enables collaboration. Like Google Docs, users currently viewing the canvas are shown in the top-right of the software as avatars. One can also be transported to another user's view by clicking on their avatar. In addition to this, a user has the possibility to bring everyone to their view by clicking on their own avatar. A user can also see the other users' cursor in the canvas and has the possibility to get feedback, reviews, and approvals on work done on the board. Like Google Docs, users can start a comment thread on objects, tag collaborators in the comment, and resolve comment threads.

When clicking on an object, the user is presented with several operations that can manipulate it. The operations are related to the type of object and will therefore vary somewhat depending on the object. For instance, when clicking on a sticky note, the user is presented with the option to change the font, but when clicking on an arrow object, the user is presented with the option to change the direction of the arrow.

2.4.3 Review of the software platforms

Although both Google Docs and Miro can be used outside of group work, the thesis will consider the softwares within the context of enabling collaborative work.

Both Google Docs and Miro fit the description of groupware as they *engage people in a common task* and *provide an interface to a shared environment*. The common task that the people are engaged in motivates the choice of software, i.e., writing an article in Google Docs or planning a sprint in Miro.

Google Docs
The interface provided in Google Docs is a shared online text editor. It provides a shared workspace to users working on the same document. The interface gives environmental cues such as displaying each other's cursor and each other's activity, for instance, seeing the words typed by other users and seeing who typed it. The software can therefore be placed high in the *shared environment dimension* on the groupware spectrum. Depending on the task, users can perform separate and individual tasks concurrently and perform activities with real-time interaction. This makes it hard to place it on the *common task dimension* since the placement on the spectrum depends on the task.

The software supports simultaneous activity, but whether or not users are active simultaneously depends on the task they are performing. For instance, a group could outline an article that they will write together in the same document and delegate each part of the article to team members in the group. The team members could then, at their own convenience complete the delegated tasks. This means that the software supports activities that both happen at the *same time* and at *different times*. Since the software is designed to support *distributed actors*, meaning users working on their own devices, the software can be characterized as supporting both *synchronous*- and *asynchronous distributed interactions*.

Supporting synchronous distributed interactions makes the software fit the category of *real-time groupware*. The attributes specified in Section 2.2 are present in the platform, meaning it is *highly interactive*, *volatile*, *ad hoc*, and so on. The interface in Google Docs is a *WYSIWIS* (what you see is what I see) interface, meaning actions in the text editor (i.e., a character) is immediately propagated to all users in the same *session*. The response- and notification times are kept short to maintain group focus.

Visual indicators in the software indicate where users are working in the document (by showing other users' cursor) and highlighting sections in the text that other users have highlighted. This can decrease the probability that other users will operate on that section in the editor. Although highlighting the section can decrease the probability, conflicting actions could occur in Google Docs. However, it does not implement locking as a way to solve concurrency control as deciding the level of granularity is not an easy task in text editors (like mentioned in Section 2.2). Instead, a user can operate in the same place in the editor as another user. Conflicting actions, for instance, a user deleting a character and another user changing the color of the same character, seems to be solved by the technique called Operational Transform (see Section 2.2.1. This can lead to some confusion between the users, which can be solved by the users utilizing an audio link like an external channel or removing the write access of other users.

The software implements the *shared feedback* approach, where the feedback on users' activities is presented in the shared workspace. Through the software itself, users can monitor each other's activities by observing their cursors and actions. The cursor shown is the typical cursor used in a text editor, and it only shows the last place clicked by another user in the editor itself. They can also use the commenting functions to solicit responses to their text in the editor.

The ability to view users' activities and monitor each others cursors and actions are examples of how users can use the *feedthrough* and *consequential communication* mechanisms in the second part of the workspace awareness framework to update their awareness information. For instance, when a user moves their cursor, another user updates their workspace awareness by simply noticing that the cursor has been moved. In this example, the workspace awareness has been updated as a result of the consequential communication mechanism. The software does not directly support the *intentional communication* mechanism, but this can be achieved using third party software (an external channel in Ellis and Gibbs description of real-time groupware, described in Section 2.2).

Miro
The interface provided in Miro is a shared online whiteboard. It provides a shared workspace to users in the form of a canvas. Like Google Docs, it gives environmental cues such as displaying users' cursors and their activity. Unlike Google Docs, the activity consists of creating and moving objects such as

different shapes, drawing, adding sticky notes and texts. Miro could be placed similarly to Google Docs on both the *shared environment dimension* and the *common task dimension* for the same reasons as mentioned above.

Miro is similar to Google Docs in many ways. It supports simultaneous activity, but how the users use the software is dependent on the task. Miro seems to be designed to be used by users on their own devices, and the software is built to enable distributed and remote teams to collaborate without the constraints of a physical location[8]. Like Google Docs, Miro can be categorized as supporting both *synchronous-* and *asynchronous distributed interactions*. Miro categorizes as *real-time groupware* because it has the attributes specified in Section 2.2. It also implements the *WYSIWIS* interface.

Visual indicators are also used in Miro to decrease the probability of other users operating on the same object. When a user moves an object, it is highlighted and marked with the user's username in the bottom-right of the object. Miro differs from Google Docs in how it solves concurrency control when users operate on the same object. When a user moves an object, it prevents other users from performing operations on the same object. This shows that Miro implements *locking* as a concurrency control when a user moves an object. But, it does not lock an object when a user merely has it pressed, and other users are allowed to operate on the same object. It could not be found how Miro solves conflicting actions, but it seems to implement Operational Technique (OT) like Google Docs.

The software implements the *shared feedback* approach, like Google Docs. The users' activities are presented in the shared workspace, and users can observe each other's cursors and actions. The cursor shown in Miro is different from the cursor shown in Google Docs. Miro uses a mouse pointer cursor, and it displays the hovering/movement of the cursor, as opposed to just the last place clicked like in Google Docs. Presenting this information in the shared workspace makes more sense in Miro than in Google Docs because moving around in the canvas requires users to click-and-drag to get around. Because of this, the position of the mouse provides information about where the user is currently looking. In Google Docs, users do not have to click and drag in the editor to move around, and presenting the last place clicked instead of the mouse's current position is a better indicator of where the user is currently working.

2.5 Computer-supported collaborative learning

Computer-Supported Collaborative Learning (CSCL) describes a field within the learning sciences, which explores how collaborative learning can occur with the help of computers [23]. Similiar to Computer-Supported Cooperative Work (CSCW), CSCL focuses on supporting group work, and providing shared workspaces to users [24]. Additionally, CSCL is focused on the learning outcomes of the computer-supported collaboration. According to Dillenbourg et al. [25, p.4], CSCL emerged as a researching field in the early 1990s and has since gained a scientific community. In the later years, CSCL has to an extent disappeared as a *distinct* pedagogical approach. Dillenbourg et al. argue this disappearance is due to collaborative activities in education being integrated into more complex environments. These environments also often include non-collaborative activities and bridge the gap between physical and digital spaces using several different tools.

There has been a lot of research into the effectiveness and quality of learning using CSCL methods. Dillenbourg et al. present an overview of different ideas, myths, and results in the field of CSCL [25, p. 5], based on research performed by multiple researchers over the last few decades. For instance, it is argued that collaboration itself does not produce learning outcomes. Therefore, the goal of designing a CSCL environment should not *just* be to enable collaboration but to also "*create conditions in which effective group interactions are expected to occur*" [25, p. 6]. It is also emphasized that CSCL environments should strive to give users a shared understanding of the environment, using shared graphical representations and visual indications of individuals' contributions. This focus is similar to *awareness* in CSCW (see Section 2.3). In the overview, Dillenbourg et al. argue that initially CSCL was tackling the problem of *compensating* for actors not being in a face-to-face situation. This focus has then gradually shifted towards how technology can *enable* collaboration in ways that are not available in a face-to-face situation. There are several other interesting points brought up in this overview, which Dillenbourg et al. sums up as:

[8] https://miro.com/about/

> "*In summary, a CSCL environment is not simply a tool to support communication among remote students but a tool used in both co-presence and distance settings for shaping verbal interactions in several ways[...] and for capturing, analyzing and mirroring these interactions in real time.*"

Within CSCL, there are several explored paradigms such as Mobile Computer-Supported Collaborative Learning (MCSCL), Computer-Supported Cooperative Writing (CSCWR), Computer-Supported Cooperative Reading (CSCR), and Computer-Supported Collaborative Drawing (CSCD). Within these paradigms, the research presented in this thesis relates to both MCSCL and CSCD, as it explores using a tablet platform, which includes drawing features, to facilitate collaboration between pupils solving mathematical tasks. While this research relates to the paradigms in CSCL, the focus is not on evaluating the effectiveness or quality of learning occurring within the platform.

In primary education worldwide, Carapina and Boticki estimate that between 2009 and 2014, tablets were used in 44% of cases where mobile devices supported collaborative activity [26]. Additionally, Carapina and Boticki conclude that a 1:1 distribution between users and devices is the most common case in which tablets are used. Research regarding drawing applications within MCSCL is not as widespread as other types of software, such as reading and writing platforms. Notable entries on this topic include research by Bollen et al. [27], and Ferraris and Martel [28]. Bollen et al. explores advantageous conditions for collaborative drawing activities, and conclude that learning results may benefit from *awareness* information, amongst other findings [27, p. 14]. Ferraris and Martel explore introducing a teacher-centered regulation role into a drawing platform. This role allows a teacher both to manage groups and to coordinate the pupils' interactions. In this thesis, adding a similar role to the platform is discussed in Paragraph *Teachers* in Section 7.5.

3 Designing a real-time groupware platform

This section presents the design of the platform developed during this project. The design is based on the theories and software platforms presented in Section 2, while some design decisions are made as a result of the context of the project.

Section 3.1 describes the concept of the platform. Section 3.2 presents design decisions that were made for the platform to be used in the test setting. Section 3.3 places the platform within the groupware frameworks from Section 2.1. Then, Section 3.4 presents how the platform is designed to attain the real-time groupware attributes. Section 3.5 discusses awareness design and shows the design of the features assumed as most important. The presented features are dependent on the implicit awareness technology provided by fulfilling a subset of the real-time groupware attributes presented in the previous section.

3.1 Concept

RQ2 relates to identifying the necessary features in a software platform to facilitate collaboration in mathematics. In the project presented in this thesis, a line-based **real-time groupware platform** is developed, allowing pupils to solve mathematical tasks collaboratively. This platform is then tested with a selection of pupils in a controlled setting. The test is described in Section 5.2. Features needed for this are identified using the knowledge gained about groupware and awareness from Section 2, adapted to the project's needs. RQ3 is answered by the test-setting and aims to examine collaborative patterns occurring when using the platform.

The designed platform includes several features and functionalities presented in Section 2. A central part of the design is to allow for **synchronous distributed interaction** between users similar to Google Docs and Miro, only in the context of drawing lines in an **expandable whiteboard**. This whiteboard is represented by a shared **What You See Is What I See (WYSIWIS)** interface, in which **replication** and **locking** are used for concurrency control.

Awareness is assumed to be an important contributor to enable collaboration and facilitate different types of it. Therefore, approaches such as **shared feedback** are included in the design of the platform. A feature supporting this approach is Cursor activity. The design of this feature within the platform is inspired by the implementation in Miro and aims to represent a user's location within the **shared workspace**. Another feature inspired by both Miro and Google Docs is Highlighting, which acts as an indicator for objects selected by other users.

Workspace awareness is supported within the platform with several different features. Features supporting the workspace awareness category *who* are **user avatars**; supporting the category *what* is **streaming partial lines**[9], having a **low notification time** for replicating object actions, **highlighting** objects selected by other users, and supporting both verbal and written **communication**; finally, supporting the category *where* is primarily **"cursor" activity**, where an avatar represents other users' location within the shared workspace.

Some additional features were added to the platforms inspired by the two software platforms presented in Section 2.4. Examples of this are: commenting and clicking on another user's avatar to jump to where they are in the shared workspace; both inspired by Google Docs. Included features inspired by Miro are: implementing a whiteboard to act as the shared workspace between users and presenting users with a side menu when selecting objects. This side menu contains actions that manipulate objects. The implementation of this side menu is shown in Figure 31.

3.2 Platform design decisions

The platform is designed to facilitate collaborative work within mathematics and is centered around users drawing and interacting with lines and objects within a shared workspace. Since mathematical

[9] see Paragraph *Streaming of partial lines* in Section 4.5.2

tasks often require many different input formats, such as numbers, equations and plots, basing the collaborative platform on line-based objects seemed apt. Having users draw lines allows their contributions to be structured in several different ways, in contrast to text editors that are often more rigid. Also, tasks in mathematics are usually solved by hand on paper in primary schools; therefore, emulating this workflow within the platform allows the users to solve them in a familiar way. As discussed in Section 3.1, the platform is designed to be used in a test setting. To enable this test, the following design decisions were made:

- **Platform flow:** How should the platform be structured in terms of screens and navigation?
- **Tasks:** What type of tasks should the platform include?
- **Core functionality:** What functionality needs to be present in the platform for the test?

As a platform, Flate is primarily meant to function as a tool used in the final tests. Some basic functionality, such as login and registration, can therefore be omitted. This allows the pupils to be ready to start testing the platform quickly. Instead of having user accounts, the pupils are given a randomized set of credentials representing them within a session. This is done by generating a random id for each pupil and generating a random word-pair (adjective + noun) to act as their username.

With no login screen, the pupils are immediately presented with a list of tasks to choose from. The platform could immediately give the pupils a predetermined set of tasks to perform for the test. However, for simplicity and flexibility, the pupils are instead instructed on which tasks to select. This way, the number of tasks per group of pupils can easily be adjusted, and the platform does not need to include methods for ordering tasks for each group. Selecting a task sends the pupils to another screen to connect with other pupils to work with. If the platform included user accounts, a list of friends or classmates that the pupils could invite could be shown within this interface. Instead, the platform lists all other pupils that have selected the same task to work on. As a pupil invites others to their session, a connection is established between all participants. After the connection is established, users can start the session and are sent to the screen where the task can be collaboratively solved. Figure 7 shows an overview of the flow between the different parts of the platform.

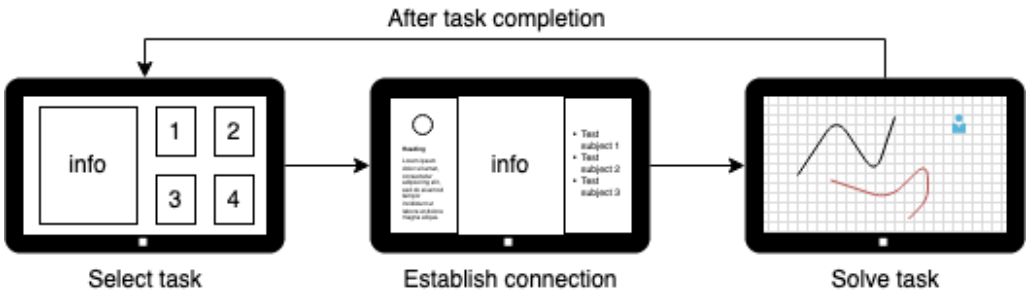


Figure 7: Flow between the different parts of the platform

To distinguish collaborative patterns that occur during testing, the tasks in the platform are chosen to make the pupils collaborate in different ways. The chosen tasks are presented in Paragraph *Description of tasks* in Section 5.2.1. The emphasis of these tasks is on *how* users solve them rather than the solution itself. Some tasks allow the pupils to delegate sub-tasks and work somewhat independently, while other tasks force and encourage the pupils to collaborate more tightly. A varied set of tasks allows for potentially more patterns to occur, as each task could influence how the collaboration would ensue.

The shared workspace is designed to provide the pupils with an expandable surface enabling structuring contributions as they see fit. Designing a flexible work surface within the platform allows different tasks to be solved using the same user interface. Another option would be to tailor the interface depending on each task, but this would somewhat limit the pupils' freedom to explore and solve tasks in their own way. As presented in Section 2.2, this form of *ad hoc* behavior is an attribute of real-time groupware and should therefore be facilitated within this platform. How the platform is designed as *real-time groupware* is described further in Paragraph *Real-time* in Section 3.4.1.

The set of features present in the platform are designed to allow the pupils to solve the different tasks during testing collaboratively. Most central are the different functions that allow the pupils to draw lines within the shared workspace and have these lines appear on other pupils' devices in real-time. As a pupil is drawing, the drawn line is chunked into smaller parts and streamed to other devices where they are replicated. Pre-rendered assets related to the task and lines are made into accessible objects that the pupils can interact with. The types of interactions include moving, deleting, commenting, grouping, and more (see Paragraph *Canvas object toolbox* in Section 4.4.3).

Streaming of lines allows the pupils to be aware of each other's actions within the shared workspace. Awareness between pupils is assumed to be important for collaboration. Other features designed to support awareness within the shared workspace include visualizing the pupils' locations within the interface and object highlighting. The design of these features is discussed further in Section 3.5.

3.3 Groupware framework placement

When designing a groupware platform, certain attributes and characteristics need to be considered. Figure 8 shows the ideal placement of the platform described above. For the platform to be placed on this right-most side of the Groupware Spectrum, the platform needs to emphasize common tasks and create a shared environment. This means that the platform would need to focus on the users of a session, letting them collaboratively work together in the same environment while giving them a common goal to achieve.

The common task in the platform is the mathematical task the users are to solve together. Therefore, giving all users an equal opportunity and responsibility in solving these tasks is essential. The platform should have no specialized roles for different users in a session to achieve this equality, letting them interact equally in the shared environment. The shared environment of the platform is the shared workspace. The state of the shared workspace, the virtual whiteboard, has to be updated for all users as actions happen. This way, all users are made aware of what is happening in the platform. If the users are constantly updated with changes in the shared workspace, they will have the same prerequisites for contributing towards the common task. Implementing the platform emphasizing these features would make the platform score high in the groupware spectrum in both dimensions. This high-scoring placement is outlined in Figure 8.

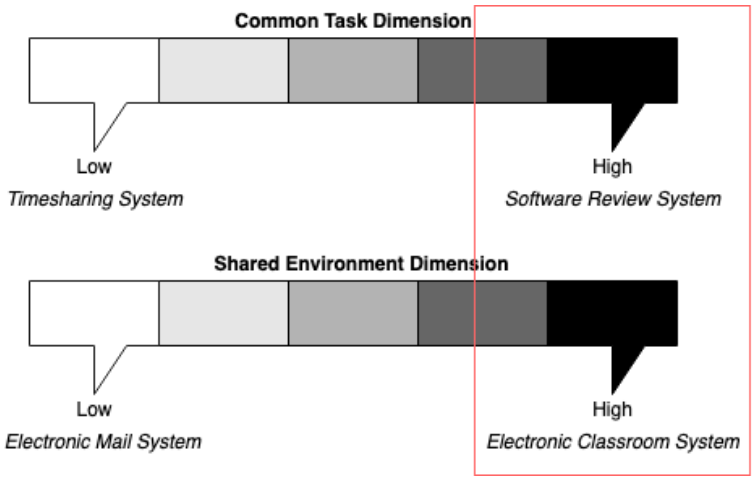


Figure 8: Ideal placement in the Groupware Spectrum

Figure 9 shows desired placement (in green) for Flate in the Groupware Time Space Matrix. This placement requires the platform to support geographically distributed users and enabling synchronous interaction within the platform. Therefore, the platform should send messages containing user actions between users' devices with minimal latency to support distributed users. How the platform should support distributed users is discussed further in Section 3.4.1.

Synchronous interaction means that users work together simultaneously to solve a common task. A platform supporting synchronous interaction may also support asynchronous interaction, as is the case with Google Docs (Section 2.4.3). Synchronous interaction requires users to collaborate in real-time, and the core requirements for real-time interaction are discussed further in Paragraph *Real-time* in Section 3.4.1.

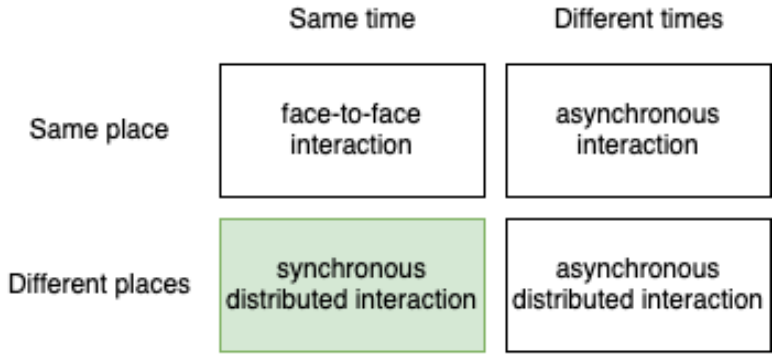


Figure 9: Placement in the Groupware Time Space Matrix

The alternative Groupware Time Space Matrix focuses on the role of the computer in collaborative work and the nature of the interactions between users. In this matrix, Flate would be placed in the upper-left, and this is shown in Figure 10. The computer, or tablet, is the medium the users use to interact with each other. The platform should allow users to interact with the same objects in the shared environment directly to facilitate tightly coupled interaction. This type of interaction requires that the platform enables users to be mutually aware of each other, as explained in Section 2.1. In general, the platform should give visual clues and feedback to reflect other users' actions within the shared environment. How the design of the platform enables these features is explained in Section 3.5.

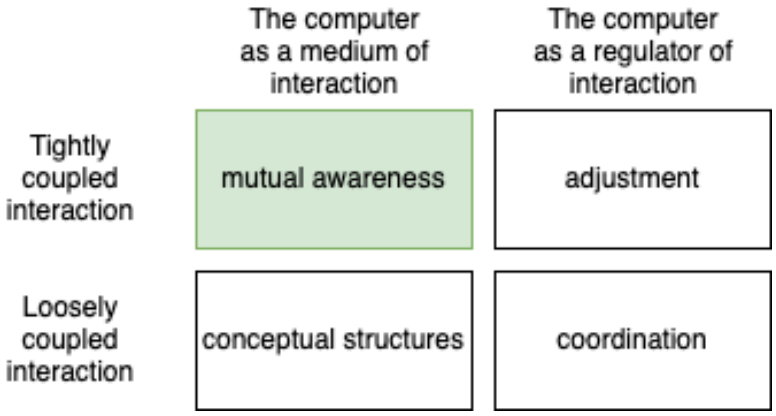


Figure 10: Placement in the alternative Groupware Time Space Matrix

3.4 Real-time groupware features

This section presents Flate as a *real-time* groupware platform and explains how the platform is designed to attain the attributes and requirements associated with this type of groupware.

3.4.1 Attributes

This section presents a selection of the attributes described by Ellis and Gibbs [9, p. 399] (Section 2.2). The attributes discussed below were chosen as they required certain design decisions to be made to

fulfill the requirements of the attributes and were relevant for the context of this project. The *ad hoc* attribute does not describe a requirement, as it simply characterizes a type of user behavior that is common within real-time groupware. It is therefore omitted from this discussion. The *focused* attribute is discussed in Section 3.4.2 as it relates to concurrency control.

Highly interactive

A highly interactive platform means that a user's actions should be quickly reflected within the user interface. To achieve this, the platform is designed to have a short response time. As the platform's core functionalities are drawing lines and interacting with objects, high interactivity is critical. Having an object not immediately respond to actions when interacting with it might make actions feel clumsy and make the platform feel less interactive. For instance, when drawing a line on paper, users naturally expect lines to be drawn as the pen moves on paper. Mirroring this behavior in a virtual context is essential to ensuring that drawing feels natural. When users perform actions such as moving drawn lines or other objects, the interface gives instant feedback. This way, the user can feel confident that objects are moved precisely as intended.

Real-time

The real-time attribute is essential in groupware platforms and is directly dependent on the platform's notification time. The notification time is the time it takes for a user's action to be handled and reproduced on another user's interface. For real-time platforms, the notification time should be short and comparable to the response time. If a user draws a line in the platform, the points in the line are streamed immediately to other users in the session. This allows them to observe the line being drawn in real-time. If lines only were to appear after a user has drawn complete lines, other users would be less aware of this action while it is happening.

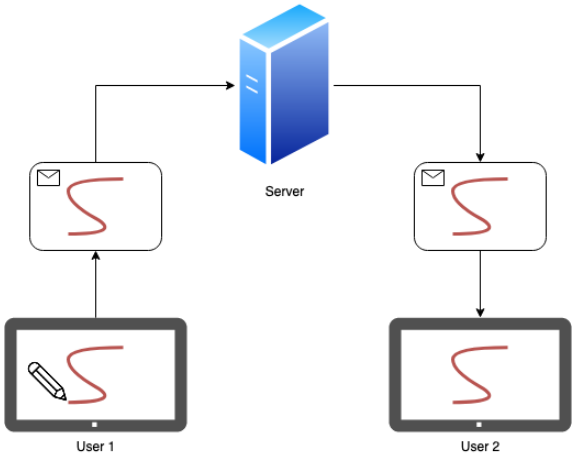


Figure 11: Line creation action message sent from User 1 to User 2 via the server

Figure 11 shows an example where User 1 draws a line. Upon creation, this line is immediately transmitted via a server as a message with information about this action. The message is then used to display the action in the other user's interface. To further enhance the real-time aspect of the platform, actions that transpire over an extended amount of time, such as drawing lines, are split up into smaller, more frequent messages. Sending smaller chunks of a line while a user is drawing allows the other user to be aware of the action in real-time. Displaying actions while they are happening enables *awareness* within the platform, which is important for collaboration in shared workspaces. Awareness within Flate is discussed further in Section 3.5.

Distributed

Distribution in real-time groupware means that all users are connected using separate devices in different

geographical locations. Flate supports distributed users and includes methods to handle connections and interactions between users. To establish connections between users, the platform presents users with an interface to create sessions and invite other online users.

A server is needed to handle connections between users, and the connections have to be kept active for all connected users. User actions that manipulate objects in the shared workspace are transmitted to all connected users via the server. This way, user actions are mirrored on all users' interfaces and lets all users have an up-to-date and identical view of the shared workspace.

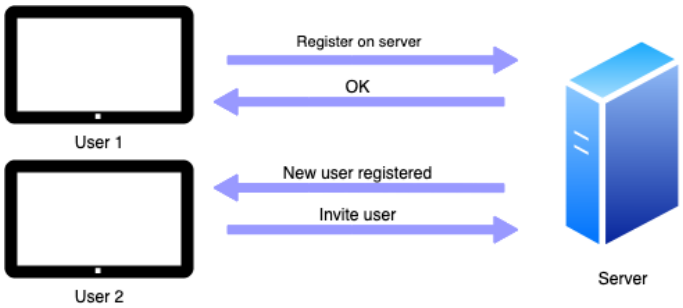


Figure 12: Socket communication example.

The server is designed with full-duplex communication. Full-duplex communication means that the server can both send and receive messages. Without this, clients would need to poll the server to get messages, whereas, with full-duplex communication, the server can push messages directly to connected clients. An event where a new client registers on the server is shown in Figure 12. Here, another client is already registered on the server and is notified by the server of the newly connected user. This client is then able to invite the new client to their session. As a user performs an action within the session, this action is sent to the server as a message. The server then pushes this message to all connected users in the session.

External channel

When users work together using real-time groupware, they are often connected externally through audio or video link. Today, platforms such as Zoom[10] and Microsoft Teams[11] are often used for this. Having an external channel is often beneficial as users can communicate directly to plan and structure their work.

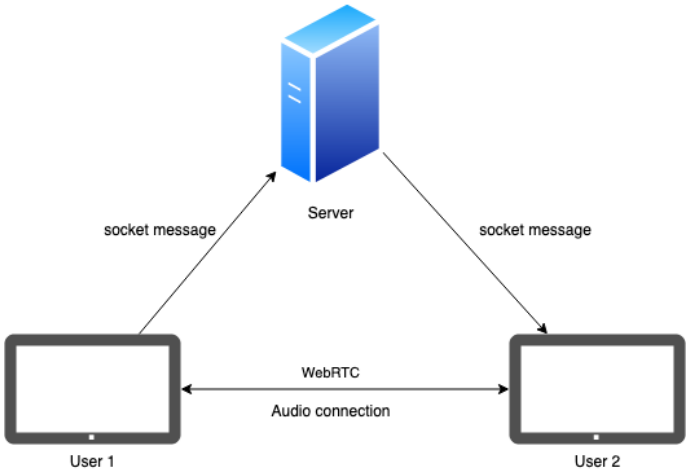


Figure 13: WebRTC connection triangle

[10]https://zoom.us/
[11]https://www.microsoft.com/nb-no/microsoft-teams/group-chat-software

As the platform's target device type is tablet computers, most users have access to a built-in microphone. Because of these built-in microphones, it is possible to include an audio link between users in the platform. Users can then communicate verbally without relying on an external connection using third-party software. The platform enables this using WebRTC[12] to stream audio between users. WebRTC enables real-time communication using peer-to-peer connections. Figure 13 shows how a WebRTC connection is established. There needs to be a server between users to establish a connection initially, but once users are connected, the audio is transmitted directly between the users' devices with minimal latency. Using WebRTC for the audio link is therefore beneficial as a media server for streaming audio is unnecessary. The initial connection between users is established through the socket server described earlier.

3.4.2 Concurrency control

As explained in Section 2.2, concurrency control is important in real-time groupware, especially when interacting with shared objects in tightly coupled activities. Within Flate, concurrency control is needed to handle potentially conflicting actions on objects in the shared workspace. All users need an equal representation of the state of the shared workspace to ensure effective collaboration. If the state is represented differently in the users' interfaces, it could lead to confusion and problems.

What you see is what I see

To achieve a WYSIWIS interface in the platform, there are several considerations to be made. All actions that modify objects in the shared workspace would need to be quickly replicated in other users' interfaces. For this to be possible, the notification time needs to be as short as possible to decrease instances where users' interfaces show different states. If the notification time in the platform is slow, for instance, a few seconds, and a user moves an object, there would be a few seconds time-frame where users are viewing different states. Figure 14 shows User 1 moving an object, while User 2's interface shows the object in the original position due to slow notification time. Objects being displayed differently may confuse users and decrease their effectiveness in collaboration, as they could be making decisions based on different circumstances.

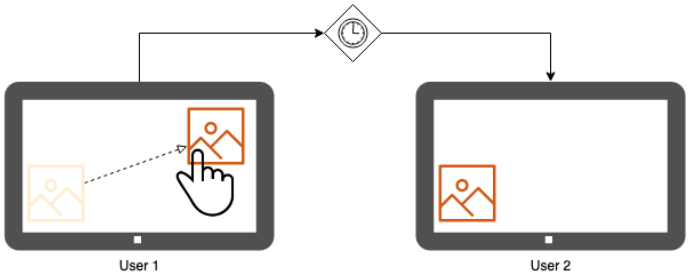


Figure 14: Two interfaces displaying different states due to slow notification time

Replication

In addition to keeping notification times short, the platform needs a robust system to ensure all users' interfaces display a uniform representation of the state. One solution would be to keep track of the state on the server and mirror this state to connected users. While it sometimes may be preferable to have a shared "truth" of the session state stored in one central place, it would require additional processing on the server. For each message, the server would then need to access objects in the stored session state and update it before broadcasting this new state to all the other users in the session, as shown in Figure 15.

[12]https://webrtc.org/

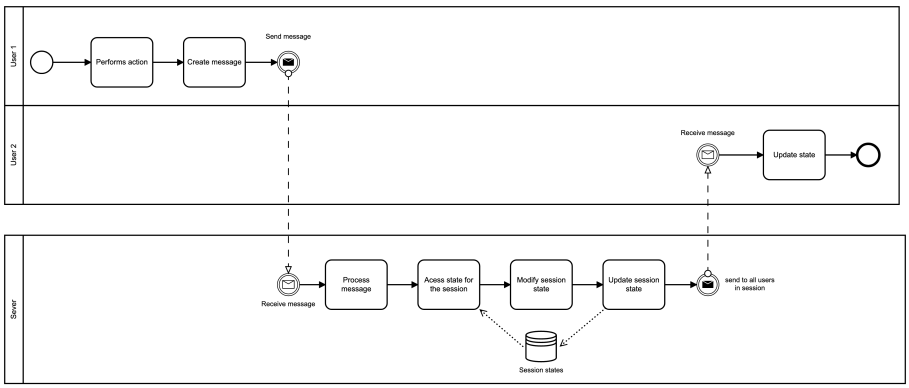


Figure 15: Processing messages on the server

Another solution, and the one implemented in the platform, is that the server only forwards messages without processing them. Receiving users would then *replicate* the actions locally. This reduces the amount of processing needed on the server, which helps keep notification times between users shorter. Reducing the amount of processing needed for each message would increase the server's performance.

Having the server only forward messages provides the benefit of faster transmission between users. Replicating the actions performed by other users requires the messages to be transmitted in a standardized format. Following is the standard set of fields these messages contain:

- Action author id
- Session id
- The type of action
- The id of the target object
- Parameter values for the action

Figure 16 shows an example where User 1 performs an action locally. A message in the format presented above is then created and sent to the server. The server forwards this message to other connected users in the session. Receiving users' devices then replicates this action by accessing the correct object and performing the action according to the parameter values. In this example, all connected users of a session end up with an identical representation of the shared workspace through their interfaces within a short time frame. Replicating actions locally reduces the notification time of the platform, which enables tightly coupled collaboration.

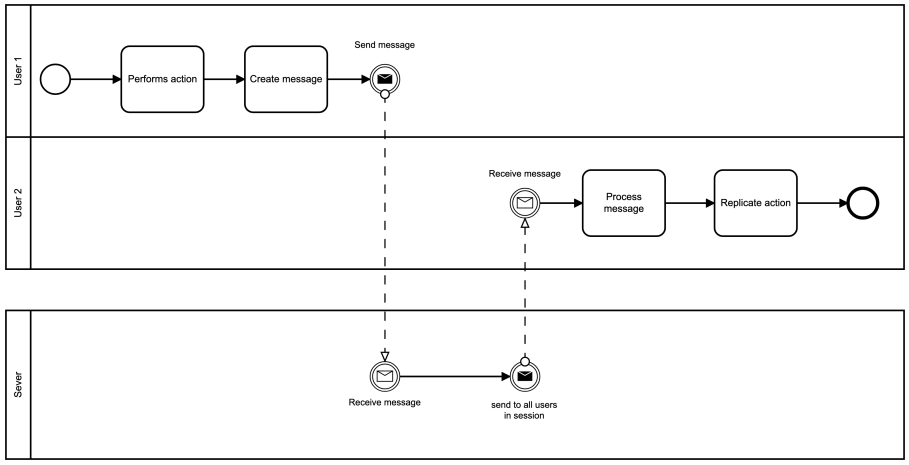


Figure 16: Replication of actions on users devices

This designed method, i.e., replicating any action in other replicas (workspaces) by sending a message to update the states, is similar to the *optimistic replication* approach operation-based CRDT, discussed in Section 2.2.1. It is only inspired by the operation-based CRDT framework, and the design does not contain it in its entirety. This because the complexities of how to resolve conflicting actions using CRDT were not needed for this project.

Locking
The platform includes concurrency control methods to prevent conflicting actions modifying the same objects happening in the shared workspace. Locking is used in the platform to handle concurrency control. Following is a description of the locking implementation used in the platform:

An object in the shared workspace is only locked when a user selects it, and upon de-selecting an object, the lock is undone. An object consists of one or multiple drawn lines or assets related to the task. A message is sent to all users in the session when an object is locked/unlocked. Information on which objects are currently locked is kept on each replica, not on the server. Locked objects are highlighted (see Paragraph *Highlighting* in Section 3.5.1), and other users are unable to interact with locked objects until the lock is resolved.

This method of locking is *non-optimistic*, as it prevents others from manipulating a locked object. By preventing others from accessing a locked resource, conflicting actions are easily avoided. Any potential conflicts that could arise because of this, like, for instance, two users trying to operate on the same object, can be solved by the pupils communicating with each other. Solving concurrency control on the same object using this method seems appropriate for this project, and the implementation costs related to this method are assessed as ideal.

As explained in Section 2.2, locking presents three problems that need to be addressed: *overhead*, *granularity*, and *request and release*. Table 1 shows how this locking implementation handles the three problems.

Problem	**Solution**
Overhead	Not having to poll the server by storing locking information locally reduces overhead.
Granularity	Locking is performed on an object level. For instance, a point in a line object can not be locked.
Request and release	The lock should only be applied as a user selects an object and should only be released as the user deselects it.

Table 1: The problems of locking addressed by the designed locking-implementation

3.5 Awareness

The section presents how the platform is designed to provide awareness technology.

3.5.1 Shared feedback

Effective support for collaboration can be provided with the shared feedback approach (for more read Section 2.3). In this approach, individual users' activities are presented within the shared workspace. From the review in Section 2.4.3, some visual indicators of users' activities in the platforms Google Docs and Miro were identified. The visual indicators deemed most notable were:

- Cursor activity: Both platforms use either a cursor in the text editor or a mouse pointer in the whiteboard canvas to present the individual user movement in the shared workspace.
- Highlighting: Google Docs presents when a user has highlighted a section in the text editor. Miro presents when another user moves an object by highlighting the object.

Since this project's scope is limited to creating a tablet platform, the design adapts some of the different techniques used to present user activities in the aforementioned platforms to this context.

Cursor activity
The cursor activity can be adapted to present the position of each user relative to what part of the expandable whiteboard is present in their viewport. This would be similar to how Miro presents cursor activity, as it shows users' hover activity in the shared workspace. However, it differs in that presenting the cursor in Miro requires the point on the canvas where the mouse currently resides, and this point is not relative to the browser's viewport. This is illustrated in the simple example below, where the mouse pointer of User 1 is presented in the shared workspace for User 2:

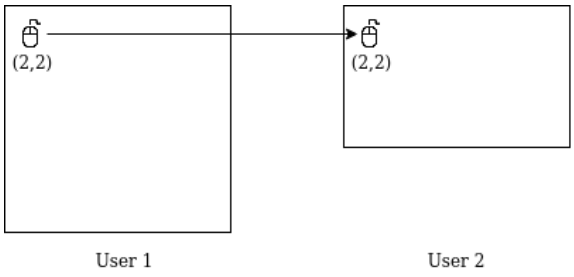


Figure 17: A simple example showing the cursor pointer presentation in two different browser sizes

The visual indicator of each user position in the project's platform would need to consider the different sizes of tablet viewports. The different sizes means that even though two users are both at the origin point, a user with a bigger screen can view a larger section of the whiteboard than a user with a smaller screen. This is illustrated in Figure 18:

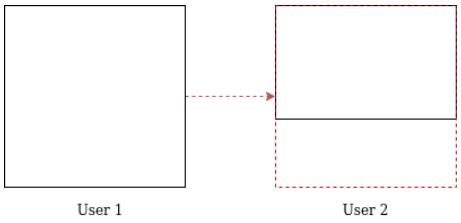


Figure 18: Illustration of the expanded view a larger screen might have compared to a smaller screen at same zoom level

This problem occurs when defining if users are currently viewing the same section of the shared workspace, and there are multiple ways of determining this. One way could be to determine a certain threshold of overlap to conclude that users are viewing the same thing. However, this solution would require each user to keep track of the viewports of all the users in a session to calculate whether or not the threshold is met. The chosen threshold can also seem arbitrary and confusing for users. At anything less than full overlap, there is the possibility of a false positive.

Another way to implement the visual indicator of a user position is to consider a fixed point for each user. The fixed point is used to represent the current view of each user and moves according to the panning of the user. If the fixed point of another user is not currently in a user's viewport, one could conclude that the other user is not viewing the same thing. This fixed point should coincide with the anchor point (meaning: the origin point of the coordinate system) in the infinite whiteboard. For instance, imagine that the anchor point of the platform is in the top-left of the canvas, but the fixed point for two users in a session is in the middle. If User 2 stands still while User 1 moves to the right, a false negative would occur at some point. This is illustrated below with the blue cross representing the anchor point, the black cross representing the fixed point from User 1, and the red cross representing the position of User 1 in the view of User 2:

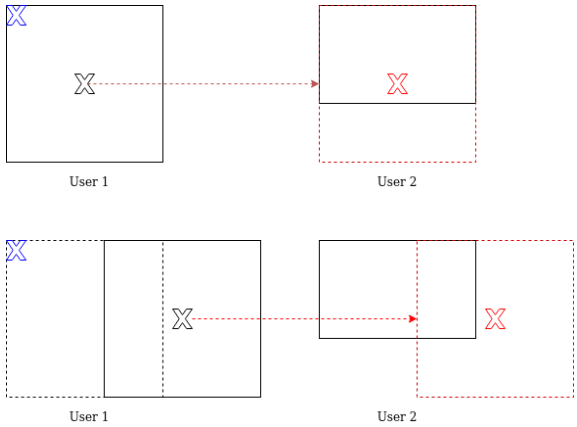


Figure 19: Illustration of how a false negative could occur

In Figure 19, User 1 and User 2 still have overlapping views after User 1 scrolls to the right. However, User 1 would be presented as outside of the current view of User 2 since the chosen fixed point represents the middle of the view of User 1. If the fixed point in this crude example had been top-left, the same as the anchor point, this false negative would not have occurred.

The fixed-point implementation has parallels to the cursor pointer presentation in Miro because it only considers a single point and presents this in the whiteboard. It is evaluated as a favorable way of designing "cursor activity" in the platform.

Highlighting

Both highlighting techniques from Google Docs and Miro can be adapted to highlight which objects users currently are operating on. In Miro, when a user moves an object, the object is highlighted for other users. Other users are then presented with a border surrounding the object, and they can not operate on the object. An example of this is depicted in Figure 20:

Figure 20: Depiction of how Miro highlights moving objects

Miro only highlights the object when another user moves it, meaning it is not highlighted when another user only has it pressed. It makes sense that Miro wants to restrict the possibility of two users moving the same object simultaneously. However, other conflicting actions like a user changing the color of an object and another user deleting it have to be solved by an intermediary. The intermediary can be a server that determines which actions have precedence and applies the actions in order. If each user has a replicated state, synchronizations with the server is needed to render the correct state of the canvas.

A user simply marking a section in Google Docs is highlighted for other users. This is adapted to the project to highlight when a user presses an object. By also implementing locking when an object is pressed to avoid conflicting actions, the platform does not have to determine the order of precedence. For more information about locking in the platform, read Paragraph *Locking* in Section 3.4.2. Figure 21 presents an illustration of how it could look like when User 1 presses an object in the platform:

Figure 21: Object highlights for User 2 when User 1 clicks on object

3.5.2 Workspace awareness

Using the conceptual framework by Gutwin and Greenberg [8], one can identify necessary features in groupware systems. The framework focuses on workspace awareness, an awareness type related to awareness within shared workspaces (see Section 2.3.1).

The first part of the framework identifies the core elements that constitute workspace awareness and presents them in Figure 5 and Figure 6. Elements of workspace awareness related to the past can be omitted due to this project's scope, which is to facilitate a test case where two users collaborate in real-time to solve a mathematical task. Thus, this thesis will only discuss elements related to the present, as shown in Figure 5. The elements are divided into the categories *who*, *what*, and *where*.

Category: Who
The elements in this category are *presence*, *identity*, and *authorship*. They relate to knowing if anyone is in the workspace, who they are, and who is doing what action.

From the descriptions of Google Docs and Miro (Section 2.4.1 and Section 2.4.2), one can notice that rounded avatars are used in both platforms to indicate the presence of others and their identities. Their avatars are a circle with either an abbreviation of a user's first and last name (Google) or just the first letter in the first name (Miro). In addition to this, each avatar has a color that is unique to the user. The designated color seems to be locally delegated in Google Docs, meaning the user's avatar will not have the same color in every users' platform. However, in Miro, the color of the user's avatar will be the same for everyone.

Using a circle avatar with letters to indicate both *presence* and *identity* in the platform seems to be an effective and easy-to-implement solution. To create the avatars, the interface of each user needs the profiles of other users. This profile needs to include the name of the users and their colors. Like Google Docs is seemingly doing, assigning the color locally requires less synchronization with the server. It is also preferred that the server only stores the necessary information to act as an intermediary. To adhere to standards set by other groupware systems, the platform is designed to have the avatars placed in the top-right.

Due to this project's scope, the *authorship* element is implicitly included in that only two users will collaborate in the test. However, the platform should include this element when used in more general contexts, for instance, in use-cases involving more than two users.

Category: What
The elements in this category are *action*, *intention*, and *artifact*. They relate to understanding what another user is doing, what the intention of the action is, and what object they are working on.

To understand what another user is doing, the *action* performed by one user is designed to be reflected in other users' replicas with a low notification time. For instance, when a user moves an object, the entire chain of movement has to be reflected in other users' whiteboards to properly convey that a displacement of the object has occurred. This also applies when a user draws a line; other users should also get immediate feedback in their whiteboards when a point in the line has been drawn. Reflecting user actions is discussed in Paragraph *Real-time* in Section 3.4.1, and in Paragraph *What you see is what I see* in Section 3.4.2.

The Paragraph *Highlighting* in Section 3.5.1, describes highlighting an object as a visual indicator to indicate what object users are currently operating on. This is a good technique to provide users with the *artifact* element.

Reflecting the *intention* of an action in the canvas seems difficult, and neither Google Docs nor Miro asks users to declare what the goal of an action is. In the presence of a communication channel, it also seems somewhat unnecessary as a user could just ask the other users what the intention of an action is. Therefore, rather than trying to imbue each action with a goal, providing users with the ability to communicate with each other will implicitly provide users with support for this element.

Category: Where
The elements in this category are *location*, *gaze*, *view*, and *reach*. They relate to understanding where another user is working, where they are looking, where they can see, and where they can reach.

The fixed-point implementation discussed in Paragraph *Cursor activity* in Section 3.5.1 can support this category. By implementing this solution the platform explicitly supports the *location* element, as it would represent the current workspace of a user's whiteboard as a position in other users' whiteboards. It would implicitly support the *gaze* element, assuming that the workspace is where a user would be looking. There exist implementations of gaze technology to correctly represent where a user is currently looking, like Gazture [29], but designing a platform that includes this is outside of this project's scope.

Using the fixed-point implementation to represent user position could also implicitly support understanding where other users can *view* and *reach*. This requires the user to understand that the representation of the position reflects the current workspace. They have to use this position as an interpretation of other users' views and reach. However, having a communication channel allows users to communicate to avoid issues surrounding this.

The second part of the framework describes the mechanisms users use to gather awareness information from the environment. The mechanisms are called *consequential communication*, *feedthrough*, and *intentional communication*, and these are further described in Section 2.3.1.

Many of the design decisions mentioned here prompt users with a combination of *consequential communication* and the *feedthrough* mechanism. For instance, a user drawing a line in the whiteboard provides information to other users using the consequential communication mechanism to gather awareness information. The user producing the line might not have created the line to inform other users in the shared workspace of the action, but the action itself produces a line in other users' whiteboard. However, if one considers the whiteboard to be an artifact and the creation of a line to be a manipulation of it, the feedback given to the user performing the action can also be observed by other users to gather information. This also applies to other actions, like moving an object.

By providing a communication channel within the platform, all three methods of *intentional communication* can be used by users to access awareness information. To make it easy for users to gather this information, the communication channel allows users to produce work in the platform while talking. An open-microphone solution enables that, as the audio link between users would always be open. In a push-to-talk solution, the users would have to push a button to open the audio link, and only after opening it could they converse with the others. A push-to-talk solution could therefore cause disruptions

and is evaluated as less optimal than an open-microphone solution.

4 Implementation

This section gives an overview of how the platform was developed throughout this project and presents the core functionalities, methods, and models. The core functionality of the platform coincides with features identified in Section 3. The implementation of the platform relates to RQ3, as the implementation was used to observe collaborative patterns that occurred. Additionally, from the usage of the platform the design and choices made regarding functionality can be validated.

Section 4.1 presents the different phases of the development process and the features added to the platform following two rounds of usability testing. Section 4.2 gives a tour of the various screens and interfaces presented to the users. Section 4.3 presents and explains the different parts of the back-end. Section 4.4 presents local processes happening within the front-end, such as drawing and handling user interaction with objects within the whiteboard. Section 4.5.1 presents structures and methods that enable communication between clients. Section 4.5.2 presents how replication of the user's actions and state between devices is handled.

4.1 Development

The development of the platform was structured based on the *Design Thinking* process. Dam and Siang [30] describe Design Thinking as an iterative process where developing an understanding of the target user is a central concept. Design Thinking is also a well-suited process for tackling problems with unknown or loosely defined solutions, which is fitting for this project. As presented by Dam and Siang, the Design Thinking process has five distinct phases:

- **Empathise** with the target users
- **Define** the users requirements
- **Ideate**, create solutions to the requirements of target users
- **Prototype**, implement the solutions into a prototype
- **Test** the prototype on users

These phases are not necessarily executed in sequential order; several phases can be parallel or repeated. During the platform's development, a complete cycle of the Design Thinking phases was completed, followed by a supplementary partial cycle for additional user testing.

Early stages - Empathise, Define, Ideate
At the beginning of the project, the focus was on collecting information and defining requirements for the platform. It was largely done by reaching out and having informal conversations with experts. Experts, in this case, were teachers and other individuals with experience in the field of educational technology. In addition to these conversations, there was a focus on finding scientific papers related to how technology and software can facilitate collaborative work between individuals. From this initial phase, core functionality and essential attributes needed in the platform were identified.

Prototyping
Prototyping began as a result of the findings from the information-gathering phase. The emphasis during this prototyping phase was to implement the local processes (Section 4.4) within the shared workspace, called the *Whiteboard* screen. Additionally, enabling communication and replication of user actions between devices (Section 4.5) was implemented. After a few months of development, early versions of both the front- and back-end were ready for initial user testing.

First round of user testing
This round of user testing aimed to measure the platform's usability and identify new features that could be valuable additions. This round of testing was done using IT students as test subjects. Test

subjects were matched with one of the authors and were given a set of tasks to complete. These tasks were designed to ensure that the test subject would use all the different features and functions present within the platform. Appendix D shows a table containing these tasks.

Findings from this round of testing were made by observing the test subjects and taking notes during tests. After each test was completed, test subjects were asked to evaluate how they experienced using the platform for solving collaborative mathematical tasks. Following are the most critical findings from the first round of testing:

- Features that support awareness within the whiteboard are essential and help facilitate collaboration
- The platform should provide users with information about the different features in the workspace
- The platform should include a guide to help users understand and utilize more functionality within the platform
- Several minor user interface-tweaks were necessary to improve usability, such as changing icons and labels on buttons, and more

Prototyping and final round of user testing
After evaluating and implementing new features based on the findings from the first round, a second round of testing was conducted to ensure that the platform was ready for the test with the pupils described in Section 5.2. A newly implemented feature was an onboarding modal when users entered the *Whiteboard* screen (Appendix C.1). Also, as test subjects selected different tools and features, a box containing a written explanation of how this tool or feature worked was shown in the top right corner of the workspace (Appendix C.2).

The second round of testing was performed using the same tasks from Appendix D with a new set of IT students. Findings from this round were minor, as the changes made from the previous round of testing seemingly improved the platform's usability. Findings consisted primarily of specific user interface changes that needed to be made to enhance the platform's usability further. A test subject also felt that *zoom* functionality could be helpful, as it would give a better overview of the whiteboard.

Finalizing the platform
After the final round of user testing, zoom functionality and minor changes to the user interface was added. This final round of prototyping was focused on covering edge cases that could cause errors when during the data generation for this project (Section 5.2). In addition, the monitoring system used during the test with the pupils, described in Paragraph *Monitoring system* in Section 5.2.1, was developed during this phase.

4.2 Front end

The front end of the platform is built for both iOS and Android, and was written using Flutter[13], Google's open-source UI toolkit for creating multiplatform mobile applications using a single code base. Flutter code is written using the object-oriented programming language Dart[14].

4.2.1 Screens

The platform includes three main screens: Session picker, Lobby, and Whiteboard. The session picker screen is where users select which mathematical tasks to solve. The lobby screen is where a connection between the users is established, and the whiteboard screen is the shared workspace where users work together to solve the common task.

Session picker

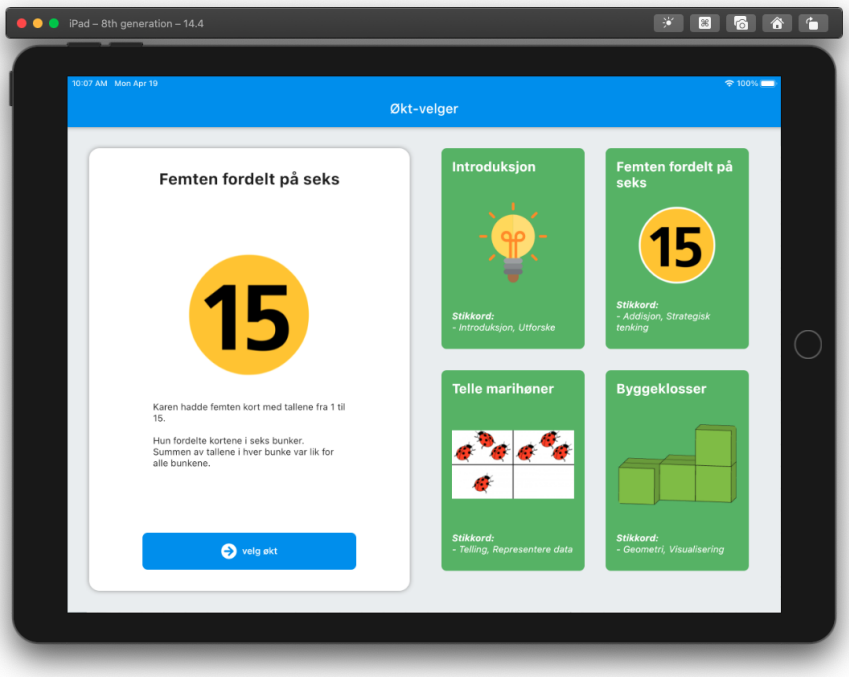


Figure 22: The session picker screen

Figure 22 shows a screenshot of the platform running on an iOS emulator. The session picker screen is the first screen presented to users when opening the platform. From here, users can see the different math tasks present in the platform. There are five tasks in the platform, including an introduction task to guide users through the different tools and functions within the whiteboard. The remaining four tasks are based on mathematical problems, detailed in Paragraph *Description of tasks* in Section 5.2.1. Tasks are shown to the right as cards in a scrollable area. After selecting a card, the box on the left side of the screen displays information about the task. The blue button on the bottom of this box lets the user move on to the *Lobby* screen.

[13] https://flutter.dev/
[14] https://dart.dev/

Lobby

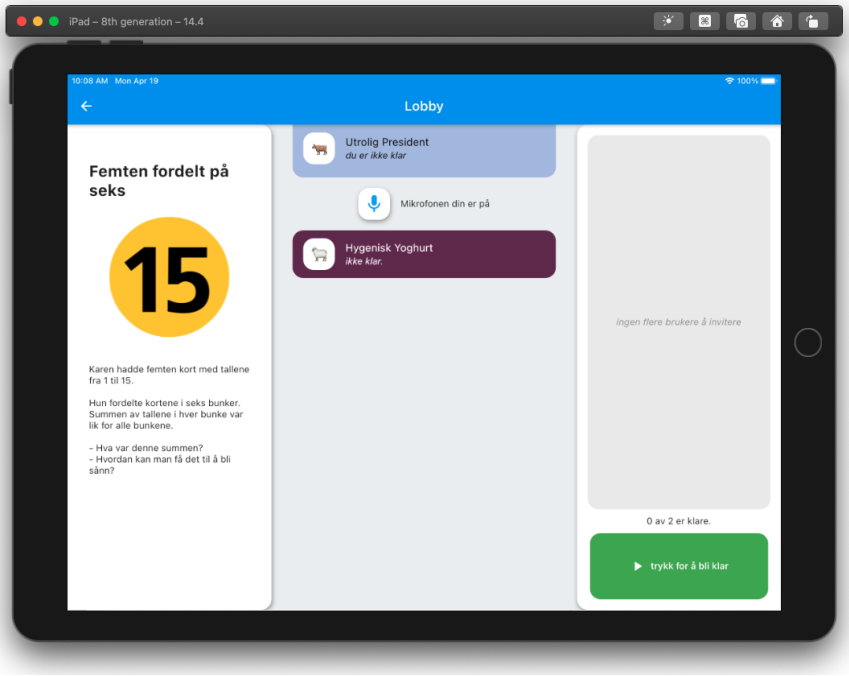


Figure 23: The lobby screen

This screen is where the connection between users is established. As shown in Figure 23, the lobby is centered around mathematical tasks chosen by the user. Information about this task can be seen on the left side of the screen. The box on the right side contains an area where other users with the same task selected are shown. From here, users can invite other online users to their session. In the figure, this area is empty as there are no other users to invite, and the user *Hygenisk Yoghurt* has already joined the room. When a user joins a session, an audio link is established between all users. The middle part of the screen contains information about the current session and allows users to toggle their microphones. Invitations are also shown in the middle part of the screen, as shown in Appendix C.3. The green button in the lower right allows users to mark themselves as *ready*. When all users in a session are ready, users are sent to the *Whiteboard* screen together.

Whiteboard

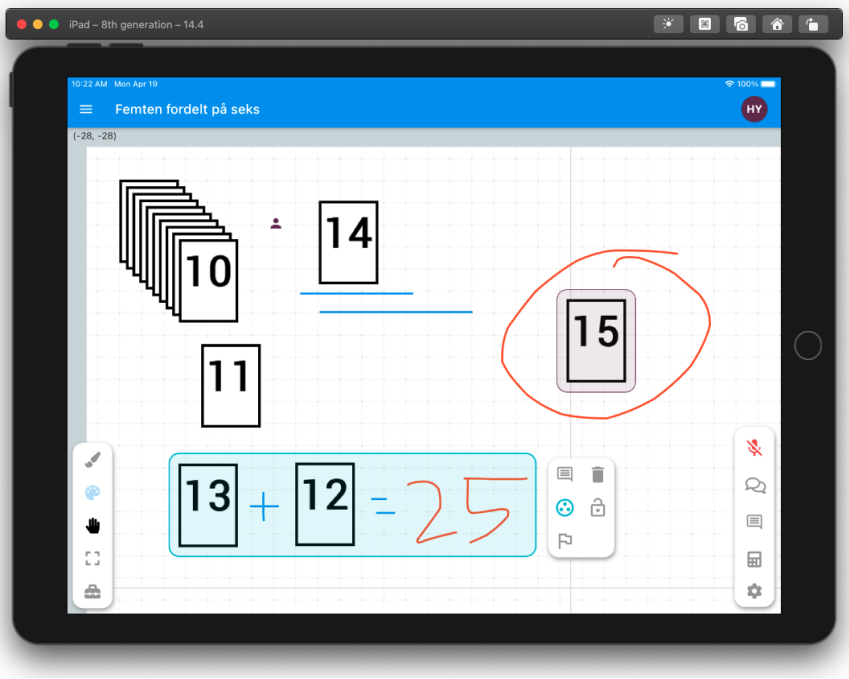


Figure 24: The whiteboard screen

Figure 24 shows a screenshot of the whiteboard screen. This screen provides users with an interface to the shared workspace for solving tasks collaboratively. The central part of this screen is the collaborative workspace in which users can draw and interact with objects. The floating container on the lower left contains tools (see Section 4.4.3) that allow the user to interact with and manipulate objects within the workspace. The container of the lower right contains additional features, such as a microphone toggle, text chat (Appendix C.4), comments (Appendix C.5), calculator (Appendix C.6), and options (Appendix C.7). On the top right corner, avatars with other users' initials are displayed. When another user's avatar is pressed, a user is transported to their location. Other users' positions are displayed as colored person-icons on the whiteboard itself. In this case, the purple user, *HY*, is currently positioned between the *10* and *14* cards. In the top left corner, pressing the three horizontal lines opens a side menu (Appendix C.8). In this menu, users can read the task's description (Appendix C.9) and see all users in the session (Appendix C.10). After the task has been completed, users can leave the session from this menu.

4.3 Back end

To enable synchronous activity, two different users using the platform have to communicate with each other. This communication is realized in the platform by utilizing the client-server model. The users' tablets are the clients, while a virtual machine (VM) administered by NTNU is the server.

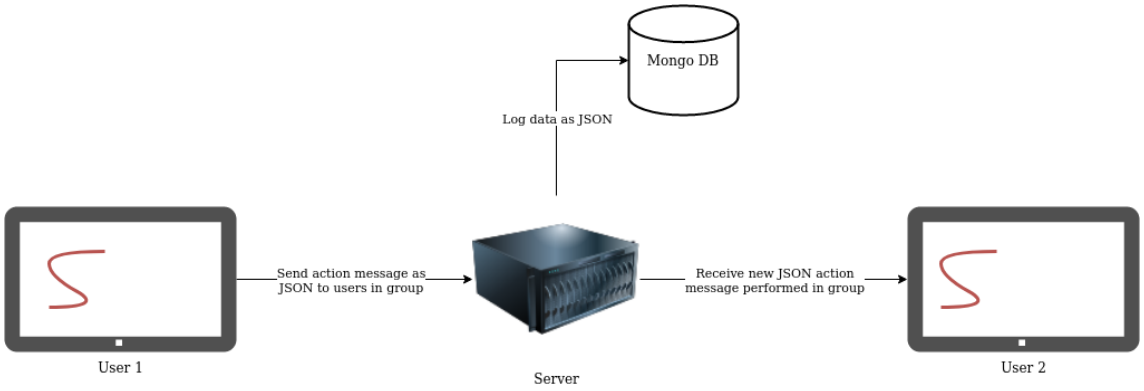


Figure 25: A simplified view of the communication between the server and the users when a user performs an action

In Figure 25, a simplified model of the communication can be viewed. In this model, an action is performed by User 1 and sent to the server. The server relays this action message to User 2. In addition to this, the server logs the action performed in a database.

This section describes the implementation of the server and the database. Section 4.3.1 discusses the technologies used to implement the server and gives an overview of the processes. Section 4.3.2 elaborates on the database used to log data, detailing the technologies used and provides an example.

4.3.1 Server

The server communicates with the clients using the WebSocket protocol, which provides full-duplex communication. Full-duplex communication allows messages in both directions, and the communication can coincide. This communication is different from the HTTP protocol, where the communication is half-duplex, meaning the messages are only sent if requested. In the WebSocket protocol, once the connection has been established, the overhead per message is minimal. Using this protocol, a client can send an action to the server, and the server can push the updates to all other clients without the client polling for updates.

This protocol was implemented in Python, version 3.6.9, using the APIs from the *websockets*[15] package. The document guide[16] for the package recommends using the *asyncio*[17] library to enable asynchronous programming, and was implemented according to this guide.

When a user enters the lobby of a session in the platform, it gets registered on the server. In this process, the client sends the auto-generated id of the user to the server. The registration on the server consists of saving the WebSocket connection and user information (e.g., username) as the JSON value in a Python dictionary[18] and using the provided user id as the key. Once the user is registered on the server, they can create a group or receive invites to an already established group.

As a user creates a group, an id is created locally for the group, which is sent to the server. The server uses this id as the key in a dictionary to keep track of all groups. This user's id is then appended as the first item of a list of user ids. When this user invites another user to the group, the invited user's id gets appended to the same list if they accept.

[15] https://websockets.readthedocs.io/en/stable/index.html
[16] https://websockets.readthedocs.io/en/stable/intro.html
[17] https://docs.python.org/3/library/asyncio.html
[18] https://docs.python.org/3/tutorial/datastructures.html#dictionaries

Users in the same group are presented with a shared workspace when they move on from the lobby in the platform. The shared workspace is an illusion; it relies on communication and replication of actions to maintain it. These are some of the most important actions to create the illusion in the platform:

- The position of a user
- User object actions
- Streaming of partial lines

When actions are performed by users, e.g., moving objects, a message is sent to the server. This message is then relayed by the server to all users in the group, informing them of the object's displacement. This displacement is then replicated locally by each client, ensuring that each user in the group has the same workspace state. Replication is discussed further in Section 4.5.

4.3.2 Database

In Figure 25, one can see that a database was used for logging actions in the platform as JSON objects. The data stored represent interactions performed by users in the same group within the shared workspace. MongoDB[19] was used for storing this. The data was stored on an NTNU VM and did not contain any identifiable information. In Listing 1, one can find an example of the data stored from one of the pilot test runs.

```
{
        "_id" : ObjectId("604b5bb0fa9434e06d183eb5"),
        "group_id" : "4513yw",
        "user_id" : "XYAeSn",
        "user_name" : "Trøtt Hest",
        "user_action" : "move_object",
        "object_id" : "kort14-s0-i13",
        "object_author" : "session",
        "message" : "none",
        "tool_used" : "hand_tool",
        "other_data" : "session_image",
        "timestamp" : "2021-03-12T13:16:48.159287"
}
```

Listing 1: Example of stored log-data

The data shown in Listing 1 is an action performed by a user. In this case, the user is identified by the name "Trøtt Hest." The action performed was moving an object, which one can see from the field *user_action*. From the *object_author* field, one can see that the object's author was "session," meaning the object moved was one of the pre-generated objects related to the task. From the data, one can also get information about the tool used, the timestamp of the action, and the group- and session-id registered on the server.

Below, a list of the user_actions logged by the server is presented. These were logged because they were assumed to be most important in indicating collaboration:

- *create_object*: Creating an object on the canvas, i.e., drawing a line or a number. Here, additional information about the type of pen tool used was stored.
- *move_object*: Moving an object. Here, additional information about the owner of the object moved was stored.
- *delete_object*: Deleting an object. Here, additional information about the owner of the object deleted was stored.

[19] https://www.mongodb.com/

- *chat_message*: Sending a chat message. Here, additional information containing the message sent was stored but not visualized in the monitoring system.
- *comment*: Commenting an object. Here, additional information about the owner of the object commented on was stored.
- *ping_object*: "Pinging" on an object. Here, additional information about the emoji used was stored.
- *position_update*: A user has updated its position. Here, additional information about the coordinate was stored but not visualized in the monitoring system.
- *misc_action*: Sub-actions like the opening of the menu, opening the calculator, opening the chat, and opening the comments were stored in one user_action. The field *message* was used to differentiate between the sub-actions.

In addition to this, the following actions were also logged: the creation of a group, the connection and disconnection of a group, the pressing, and un-pressing of an object, the locking of an object, the grouping and ungrouping of an object, and the rotation of an object.

Logging interaction data enables analyzing collaboration in different ways. For instance, one could create a histogram plot of all interactions done during a specific session. One could also visualize the interaction data in a monitoring system. The latter was used as a data generation method to make observations, and it is discussed in Section 5.2.

4.4 Local processes

This section presents methods and data models that enable the *local* part of the core functionality within the platform, i.e., the actions performed by users on their devices. The local part of the core functionality includes drawing and representing lines graphically, turning lines and images into user-accessible objects, and letting users interact with objects and the whiteboard itself.

4.4.1 The drawing pipeline

Many of the methods and models that support the drawing pipeline in the platform are based on the GitHub repository *whiteboardkit* made by user *abdulaziz-mohammed*[20]. This repository includes code for a single-user whiteboard with rudimentary support for drawing. The decision to import and modify this code within the platform was made to save development time in the early stages of the project. Classes from this repository were heavily modified and built upon, and made into a collaborative whiteboard supporting multiple users collaborating.

Important classes
`Whiteboard` is the widget[21] responsible for displaying the whiteboard to users and registering user input. The widget contains two canvases, one for rendering lines and one for images. User input is then sent to `DrawingController`, which modifies data according to the input it receives. Within `Whiteboard` there is a class named `ToolBox` that lets the users choose between the different whiteboard tools in the platform (see *Tools*). When a touch is registered, updated, or removed, information is sent to `DrawingController`. This information includes coordinates of the touch, the timestamp, and an id.

`DrawingController` is the class responsible for creating and manipulating data based on user input. There are three main methods for input handling in `DrawingController`:

1. **addTouch:** A new touch is registered.
2. **updateTouch:** A touch is moved, position changed.
3. **removeTouch:** A touch is removed.

[20]https://github.com/abdulaziz-mohammed
[21]https://api.flutter.dev/flutter/widgets/Widget-class.html

As a new touch is registered, it is stored in a dictionary. Storing each touch as an entry in this dictionary makes it possible to track multi-touch gestures. Touch-entries are updated as touches are moved and removed. `DrawingController` also keeps track of which tool the user is currently using and handles input depending on this.

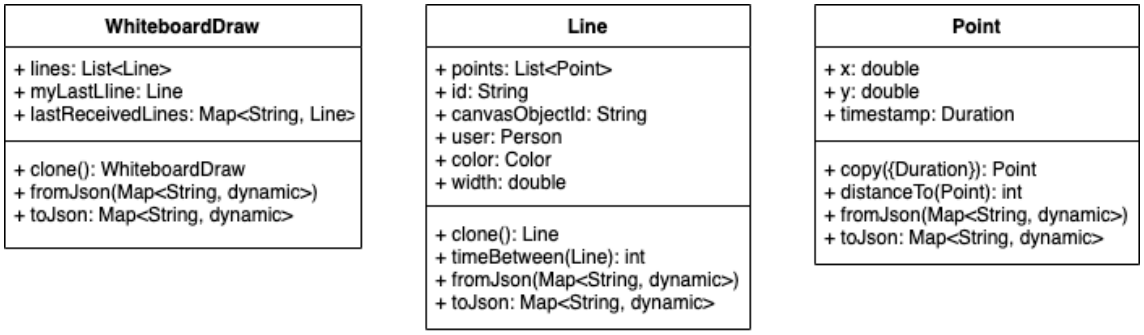


Figure 26: Three model classes in the drawing pipeline

Figure 26 shows three important model classes used by `DrawingController` in the drawing pipeline. The `DrawingController` includes an instance of `WhiteboardDraw` where `Line` objects are stored. After input methods are completed by `DrawingController`, a copy of the `WhiteboardDraw` is sent to the whiteboard for rendering (see *Rendering lines*). `Line` objects contain a list of `Point` objects. Additionally, `Line` objects include several fields that identify and characterize it, such as id, color, width, etc. The `Point` class consists of a timestamp from when the point was created, and two double fields that combinedly represents the position of the point within the whiteboard.

Pipeline

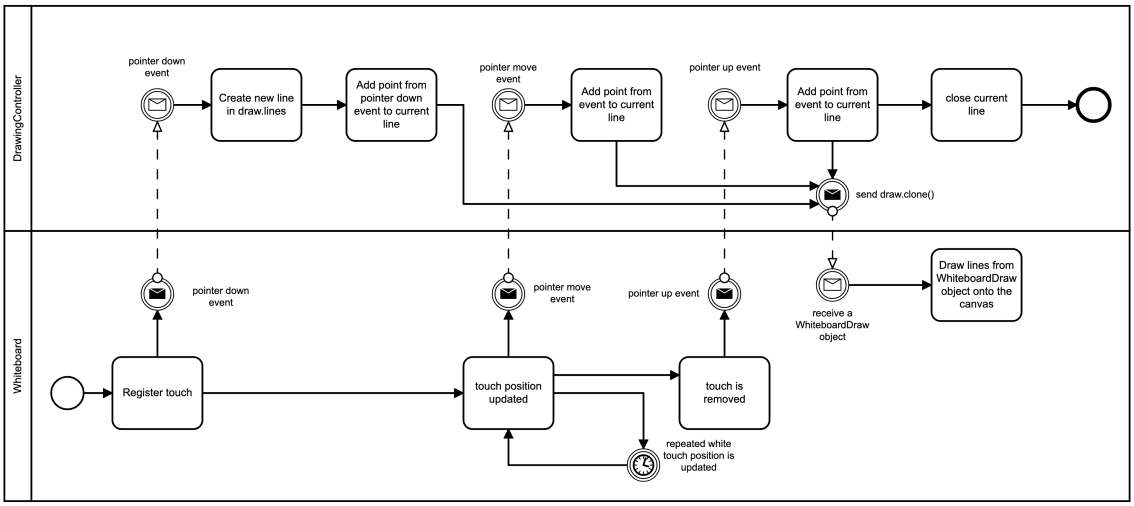


Figure 27: The drawing pipeline simplified

Figure 27 shows a simplified view of the drawing pipeline and includes its most important events. This set of events allow users to draw lines using one of the drawing tools within the platform. When a new touch is registered by `Whiteboard` the `addTouch` method in `DrawingController` is triggered. The `DrawingController` then adds a new `Line` to its `WhiteboardDraw`. A `Point` object is then added to to the line, and a copy of the `WhiteboardDraw` is sent to `Whiteboard` for rendering. As the user updates the position of the current touch, the `updateTouch` method adds a new point to the current line based on the new location of the moving touch. This method is repeated for as long as the user updates the location of the touch without removing it. When the touch is removed, the line is closed.

Rendering lines
The `Whiteboard` class has a subscription that listens to a stream of `WhiteboardDraw` objects from `DrawingController`. When a new `WhiteboardDraw` object is received, the state of `Whiteboard` is updated with the lines from this object using `setState()`[22]. As the `Whiteboard` widget is re-drawn, so are the canvases in its render tree. The canvas responsible for rendering lines in the `Whiteboard` state then draws lines between all `Point` object in each `Line` object with its correct color and line width.

Line chunking
`DrawChunker` is a class used when transmitting partial lines to be shown on other users' devices. Without this class, lines would only appear on other users' devices after a line has been completely drawn. While the user is drawing, `DrawingController` sends copies of the `WhiteboardDraw` to the `DrawChunker` every 100th millisecond. As mentioned above, the `WhiteboardDraw` contains all lines in the whiteboard, and a field called `myLastLine`. This field references the last line drawn by the user, and while drawing, it refers to the current line being drawn. The `DrawChunker` then converts the current line into JSON-serializable chunks, which are then transmitted for replication (see Section 4.5).

4.4.2 Canvas objects

Users can interact with drawn lines on the whiteboard, as lines are made accessible by encapsulating them in a type of `CanvasObject` upon creation. Images related to the mathematical tasks, elsewhere referred to as pre-rendered objects, are also encapsulated by a different extension of this same class. `CanvasObject` is an abstract class that surrounds its child or children with a hitbox. This hitbox is a rectangle (Flutter Rect[23] class) based on the child's size, and includes a `contains(Offset)` method that returns `true` if the provided offset is within the bounds of the rectangle.

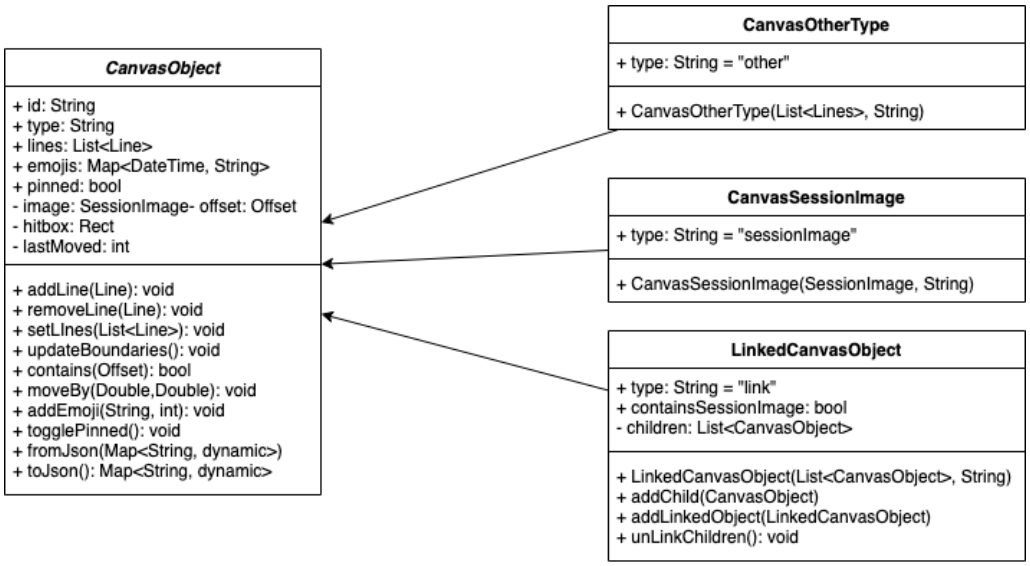


Figure 28: Canvas Objects

Figure 28 shows the abstract `CanvasObject` class, and three subclasses that extend it. `CanvasOtherType` is used for line based objects, `CanvasSessionImage` is used to make images interactive, and `LinkedCanvasObject` is made by grouping several `CanvasObject` instances together. The main benefits of this inheritance is that all the different subclasses of `CanvasObject` include a similar set of methods, and that all subclasses can be handled as instances of `CanvasObject` within the platform. While all subclasses includes the methods from `CanvasObject`, the classes override them in different ways as shown in Figure 29.

[22]https://api.flutter.dev/flutter/widgets/State/setState.html
[23]https://api.flutter.dev/flutter/dart-ui/Rect-class.html

Some subclasses, such as `LinkedCanvasObject` include additional methods to handle actions that are not applicable to the other subclasses.

Line-based object creation

The last event in the drawing pipeline from Figure 27 is *close current line*. This event happens when a line is completely drawn, i.e., the user's touch is removed. A `CanvasObject` that encapsulates the line is then created. Lines drawn within a certain threshold of time are joined together within the same object. Joining is performed based on the assumption that lines drawn one right after the other are often are related, e.g., the number "5" is often drawn using two lines and should therefore be joined. This threshold has a default value within the platform, which can be changed within the user interface to allow users to control the granularity of joined lines, even allowing complete statements to act as one object.

The subclass used for line-based objects is the `CanvasOtherType` subclass. When the `CanvasObject` system was originally designed, the platform used TensorFlow[24]-based machine learning methods to classify whether or not the user had drawn a number or a mathematical operator. The idea was to create subclasses for numbers and operators and only classify an object as a `CanvasOtherType` if it was neither. The classification was dropped later on in the development process as the need for it dissipated, resulting in only `CanvasOtherType` being used for line-based objects.

CanvasOtherType

```
void moveBy(double dx, double dy) {
  for (var line in _lines) {
    for (var point in line.points) {
      point.setX(point.x + dx);
      point.setY(point.y + dy);
    }
  }
  updateBoundaries();
  _lastMoved = DateTime.now().millisecondsSinceEpoch;
}
```

CanvasSessionImage

```
@override
moveBy(double dx, double dy) {
  double x =
      (this._offset.dx + dx) >= 0 ? this._offset.dx + dx : this._offset.dx;
  double y =
      (this._offset.dy + dy) >= 0 ? this._offset.dy + dy : this._offset.dy;
  this._offset = Offset(x, y);
  updateBoundaries();
}
```

LinkedCanvasObject

```
@override
moveBy(double dx, double dy) {
  for (CanvasObject child in _children) {
    child.moveBy(dx, dy);
  }
  updateBoundaries();
}
```

Figure 29: Example of an overridden method across different subclasses

CanvasObjectHandler

`CanvasObjectHandler` is the class keeping track off all objects in the whiteboard, and handles object manipulation. `CanvasObjectHandler` includes a dictionary, called `characters`, that contains all `CanvasObjects` in a session. Upon creation, the `CanvasObjectHandler` generates a random alphanumerical `String` and provides this to the different subclass-constructors as an id. All subclass constructors take in this id, while the first parameter which sets the child/children of the subclass differs between subclasses, as shown in Figure 28. These constructors also set the `type` field according to the type of subclass.

As the user interacts with objects, the `CanvasObjectHandler` applies the modification to the correct object. Listing 2 shows that `CanvasObjectHandler` can move all types of `CanvasObject` as all the subclasses inherit and overrides the same methods.

[24] https://www.tensorflow.org/

```
void moveSelectedObjects(double dx, double dy, List<Line> drawLines) {
    if (allSelectedObjects != null) {
      for (var object in allSelectedObjects) {
        if (object.type != "sessionImage") {
          updateObject(object.id, drawLines);
          _characters[object.id].moveBy(dx, dy);
          _canvasObjectUpdateController.sink.add(CanvasObjectAction.move(
              _characters[object.id], dx, dy, session.user));
        } else {
          _characters[object.id].moveBy(dx, dy);
          _canvasObjectUpdateController.sink.add(
            CanvasObjectAction.moveImage(
                _characters[object.id], dx, dy, session.user),
          );
        }
      }
    }
}
```

Listing 2: moveSelectedObjects method in the `CanvasObjectHandler` class

4.4.3 User interaction

In the platform, the users can interact with objects in two ways: either by using the different *Tools* or by using the features within the *Canvas object toolbox*.

Tools

In the shared workspace, there are multiple tools the user can use to draw and interact with objects in the collaborative surface. These tools are selectable from the bottom left container in the *Whiteboard* screen (see Figure 24). The user can choose which tool to use by pressing the different icons. From the top, the first button lets the user choose between different drawing tools from a pop-up menu, as shown in the middle frame in Figure 30. From left to right, the drawing tools are:

- **Brush:** Standard drawing tool, which lets the user draw lines of any color, length, and width.
- **Timed line:** Draws a semi-transparent line with a fixed width and the user's color. These lines disappear after a set amount of time. Users can change this time threshold in the whiteboard options.
- **Straight line:** Draws straight lines only.
- **Polygon tool:** Allows users to draw polygonal shapes. Adds points on the interface as the user presses down on the whiteboard. Upon pressing the first point, lines are drawn between all points, forming a polygon.

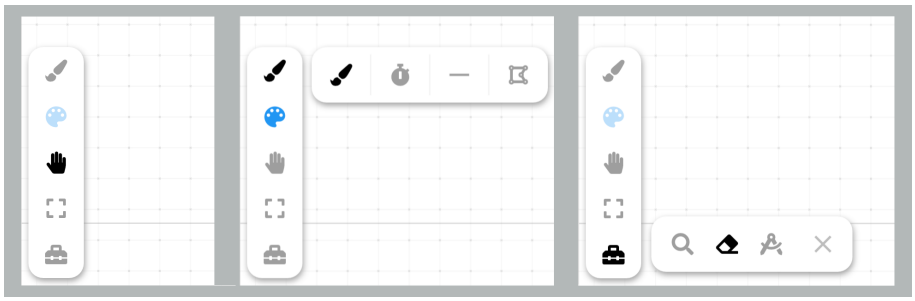


Figure 30: Screenshot snippets showing all whiteboard tools icons

The *color palette* icon opens a color picker when pressed (Appendix C.11). Colors chosen from this palette are applied when using all drawing tools, except for *timed lines*, which always use the user's color. The *hand tool* is the default selected tool when starting the platform. This tool lets the user move their position within the whiteboard by pressing and dragging. Objects can be selected using the hand tool. Selecting an object highlights it, and opens the *Canvas object toolbox*. By dragging in the box surrounding an object, users can move it. The *multi-select* tool can select multiple objects at once by pressing and dragging over objects. This tool is selectable by pressing the square icon with the four corners. The toolbox icon on the bottom reveals a pop-up menu with additional tools, as seen in the right frame in Figure 30. From left to right, the tools in this menu are:

- **Zoom:** Lets the user zoom in or out by pinching (Appendix C.12).
- **Eraser:** Lets the user delete line objects by dragging over objects.
- **Compass:** Allows the user to draw arcs and circles (Appendix C.13).

Canvas object toolbox

The canvas object toolbox consists of a box surrounding selected objects and a side menu appearing adjacent to it. An example where both single and multiple objects are selected is shown in Figure 31. The box is drawn based on the size of the corresponding object(s). When multiple objects are selected the box expands to include all objects while drawing a darker cell surrounding each object. This box contains a gesture listener that responds to user input, allowing users to perform the following actions:

- **Move:** Touching and dragging allows users to move all selected objects.
- **Rotate:** Using two fingers, users can rotate single objects.

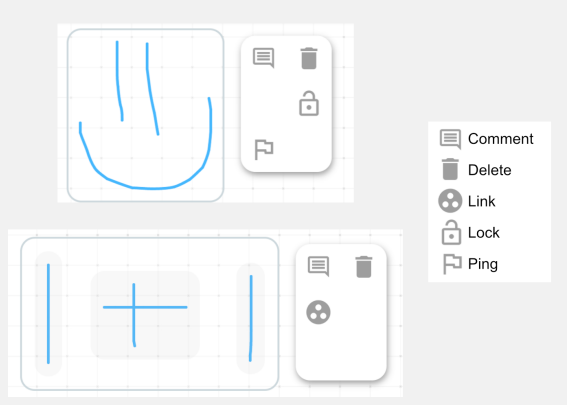


Figure 31: The canvas object toolbox when single and multiple objects are selected.

Depending on the object type currently selected, the side menu shows different icons. These icons relate to the following actions:

- **Comment:** Allows the user to make comments related to the selected object(s). Shown for all types of objects.
- **Delete:** Deletes the selected objects(s). This icon is only shown when one or more line-based objects are selected, as images related to the task can not be deleted.
- **Link:** Allows multiple objects to be grouped, allowing them to be interacted with as a single object. This icon is only shown when multiple or linked objects are selected. Linked objects are highlighted in a turquoise color, as seen in Figure 24.
- **Lock:** Allows objects to be locked in place. Locked objects cannot be moved or deleted. The icon is only shown for single and linked objects. All users in the session can unlock a locked item.

- **Ping:** Allows users to *ping* an object in order to get their attention faced towards this specific object. Pinging an object marks it visibly within the whiteboard by attaching specific emojis to it. In addition to an audible notification, a pop-up is displayed within the receiving users' interfaces. Pressing this pop-up will transport users to the pinged object's position in the shared workspace. This action is only available for single and linked objects.

4.4.4 Canvas object actions

To reproduce a user action on other devices, the actions need to be transmitted. This process is described in Section 4.5.1. Supporting this transmission is the `CanvasObjectAction` class, shown in Figure 32. This class includes information on what action was performed, which objects are related, which user performed the action, and parameters needed for replication.

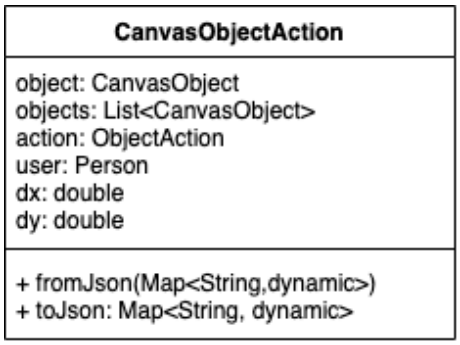


Figure 32: The `CanvasObjectAction` class

A `CanvasObjectAction` object is made after each user action is completed locally. This object is then serialized into a JSON object and transmitted to other users for replication. Replication of actions is presented in Section 4.5.2. The `CanvasObjectAction` class has multiple named constructors relating to each of the actions possible within the platform. The constructors for *move* and *link* actions are shown in Listing 3.

```
CanvasObjectAction.move(CanvasObject object, double dx, double dy, Person user) {
    this.action = ObjectAction.move;
    this.object = object;
    this.dx = dx;
    this.dy = dy;
    this.user = user;
}

CanvasObjectAction.link(List<CanvasObject> children, CanvasObject object, Person user) {
    this.action = ObjectAction.link;
    this.objects = children;
    this.object = object;
    this.user = user;
}
```

Listing 3: Move and link action constructors

4.5 Communication and replication

Communicating the actions a user performs and replicating these locally for each user are essential for the shared workspace. In Section 4.5.1, the messages that are created for the actions listed in

Section 4.3.1 are shown. Section 4.5.2 explains how the platform replicates the actions of other users by using the action messages it receives.

4.5.1 Communication

To be able to communicate with the server, the platform implements the client-side WebSocket protocol with the *web_socket_channel*[25] package. This package provides an API to communicate over a WebSocket easily.

All actions done in the platform are completed by executing Dart methods, and the methods manipulate Dart objects. To communicate these actions with the server, the data must be sent in a format suitable for converting into a stream of bytes. This was achieved by implementing serializers and de-serializers. A serializer is a method to take a data structure and convert it into a series of bytes, and the inverse operation is a deserializer. The serializers convert to the JSON[26] format since it encodes objects into a string.

The position of a user
The position of a user is indicated as an avatar in the shared workspace. A fixed point of a user's view is used to communicate their current position in the shared workspace. Each user has an overview of all other users in the same session saved as a list of `Person` objects. The user position is a property of the `Person` object, and it is used to represent the current position in the workspace.

Each second, the platform evaluates the user's current position up against their previous position. If the position has changed, the platform sends the `Person` object to the server with the updated positions. To serialize the object, the `toJson()` method in Listing 4 is used.

```
Map<String, dynamic> toJson() => {
    "id": this._id,
    "name": this._name,
    "color": Constants.colorToHexString(this._color),
    "emoji": this.emoji.toString(),
    "position_dx": this.position.dx,
    "position_dy": this.position.dy,
    "is_ready": this.isReady,
  };
```

Listing 4: Serializer for the `Person` object

The serialization creates the JSON object shown in Listing 5. The JSON object is included in the message sent to the server in a field called `actionData`. The message will also include the id of the user sending the message, the id of the group, and the action performed as fields. In this case, the action performed is titled `user_position_update`, so the message to the server will include this in its `action` field.

[25] https://pub.dev/packages/web_socket_channel
[26] https://www.json.org/json-en.html

```
{
    "user":
        {
        "id": String,
        "name": String,
        "color": String,
        "emoji": String,
        "position_dx": double,
        "position_dy": double,
        "is_ready": boolean
        }
}
```

Listing 5: User object after serialization

When the server receives a JSON object, it checks what type of action the message contains by checking the `action` field. All actions possible are checked in an *if-elif* clause, and unsupported actions are logged as an error. Upon receiving an `user_position_update` action, the server uses the group id to identify all other users in the group and forwards it to them. However, it changes the action field to `receive_user_position_update`. The server also logs the action as a `position_update` log, as shown in Section 4.3.2.

User object actions

As presented in Section 4.4.3, there are multiple ways in which users can interact with objects using the *Canvas object toolbox*. In addition to these interactions, there are two other actions that are made into `CanvasObjectAction` objects.

- **Create:** Happens when a line object is created, i.e., the user has completed drawing a line using one of the drawing tools.
- **Press:** Happens as objects are selected using the hand tool or the multi-select tool. When another user selects an object, it is highlighted in this user's color, and the object is locked. Locking an object ensures no other users can not perform conflicting actions on it.

Most of these operations are executed as methods in the `CanvasObject` class, described in Section 4.4.2. To communicate the actions performed by a user to the server, the actions are made into `CanvasObjectAction` (see Section 4.4.4) objects. `CanvasObjectAction` objects are then serialized as JSON objects as shown in Listing 6. These JSON objects are then transmitted to all the other users in the session for replication.

```
{
    "object": CanvasObject,
    "objects": [CanvasObject, ...],
    "action": String,
    "dx": double,
    "dy": double,
    "user": Person
}
```

Listing 6: All the fields an action message sent to the server can include

The fields `object`, `objects`, and `user` in Listing 6 do not contain primitive types. These fields have data that themselves are serialized objects. The `user` field has the serialized `Person` object in the same

format as shown in Listing 5. `Object` and `objects` both contain similar data, respectively either a `CanvasObject` or a list of `CanvasObject` instances. These objects are serialized using the `toJson()` method the class contains, as shown in Figure 28. This is similar to the `toJson()` method shown in Listing 4.

The format shown in Listing 6 shows all the possible fields that an action message could have. The three fields used in every message are `object`, `action`, and `user`. These fields are all pretty self-explanatory; they refer to the `action` performed on an `object` by a `user`. For the create, delete, press, pin, flag, rotation, these three fields contain everything needed to replicate the action locally.

If a user has grouped several objects into one object, the `objects` field is included in addition to the three fields. The `objects` field has the grouped objects, and the `object` field refers to the new object created by grouping. For the move action, the `dx` and `dy` fields are included to indicate the new placement of the object.

Streaming of partial lines

When a user is drawing a line, they are essentially adding new points to a list. This list of points is what constitutes a line. When the user has completed the creation of a line, an object with that line is created. The user also has the possibility to include multiple lines into one object when drawing by starting a new line right after drawing one within a set time threshold. After the object has been created, an action message with the create action is sent to the server (mentioned above in *User object actions*).

As described in *Line chunking*, a local process called chunking occurs periodically as users are drawing. Listing 7 shows the format of the JSON object sent when chunking.

```
{
    "id": int,
    "draw":
        {
        "lines":
        [{
            "points":
            [{
                "x": double,
                "y": double,
                "timeStamp": int,
                }, ...],
            "color": String,
            "width": double,
            "duration": int,
            "id": string,
            "canvasObjectId": String,
            "user": Person
            }, ...],
        },
    "createdAt": String
}
```

Listing 7: Example of a line message sent to the server

The field `draw` represents a serialized `WhiteboardDraw` object, which contains the information about the partially drawn lines. In `draw`, a field called `lines` contains a list of lines. If a user is currently drawing multiple lines in the same object, the length of this list will be bigger than 1. As mentioned above, a line is itself a list of points. The `points` field is a list of points, where each element in the list specifies the `x` and `y` of the point and the `timeStamp` of when the point was drawn. Each line also contains information about the `color` and the `width` of the line. A line has an `id`, and it is also

connected to the object it is creating with the `canvasObjectId` field. `Duration` specifies how long the user spent drawing the line. The `user` field contains the serialized `Person` object of the user who drew the line.

4.5.2 Replication

The position of a user
All other users in a group will receive a JSON message with the `action` specified by the server. The platform will then deserialize the JSON object containing the `Person` object, which it will find in the message's `actionData` field. It deserializes these objects by using the method in Listing 8.

```
factory Person.fromJson(Map<String, dynamic> json) {
    return Person(
      json["id"] as String,
      json["name"] as String,
      Constants.colorFromHex(json["color"]),
      Emoji.byName(json["emoji"]),
    )
      ..position = Offset(json["position_dx"], json["position_dy"])
      ..isReady = json["is_ready"].toString().toLowerCase() == "true";
}
```

Listing 8: Deserializer to convert JSON data to the `Person` object

User object actions
Users actions are replicated using `CanvasObjectAction` instances, which are presented in Section 4.4.4. When a JSON-serialized `CanvasObjectAction` object is received, this object is deserialized using the named constructor `fromJson(Map<String, dynamic>)` in the `CanvasObjectAction` class. This action object is then handled by the `receiveObjectAction` method (Appendix B.1) within the `CanvasObjectHandler` class. Based on the type of action performed, denoted by the `action` field in the `CanvasObjectAction` object, this action is replicated using the corresponding methods:

```
void handleMoveAction(CanvasObjectAction action, List<Line> drawLines) {
    String objectId = action.object.id;
    double dx = action.dx;
    double dy = action.dy;
    updateObject(objectId, drawLines);
    moveObject(objectId, dx, dy);
}

void updateObject(String id, List<Line> drawLines) {
    List<Line> newLines =
        drawLines.where((line) => line.canvasObjectId == id).toList();
    _characters[id].setLines(newLines);
}

void moveObject(String id, double dx, double dy) {
    if (_characters.containsKey(id)) {
      _characters[id].moveBy(dx, dy);
    }
}
```

Listing 9: Method for replicating move actions

Listing 9 shows the methods used to replicate received move actions. The method `handleMoveAction`

receives a `CanvasObjectAction`, which represents the action to replicate, as well as a reference to all lines currently in the users' whiteboard. The `updateObject` method is called to ensure that references between the local `CanvasObject` and its related lines are correct. This way, as the `moveObject` method is called, the locally stored object's related lines are moved correctly. Listing 9 also shows the `moveObject` method, which uses the `moveBy` method from Figure 29 to move the lines related to the `CanvasObject`. By using these methods, the receiving user's device can replicate the move action. Other actions are handled similarly with their own methods.

Streaming of partial lines

Listing 7 (Section 4.5.1) shows the structure of messages sent between devices as lines are drawn. These messages are used to replicate another user drawing a line on all devices in a session. Lines sent via these messages are deserialized into `Line` object instances and added to the `WhiteboardDraw` object of the receiving users. As explained in Paragraph *Pipeline* in Section 4.4.1, the `WhiteboardDraw` contains all lines to be drawn in the whiteboard. Due to the frequency of these messages, lines appear to be drawn in real-time on receiving users' devices.

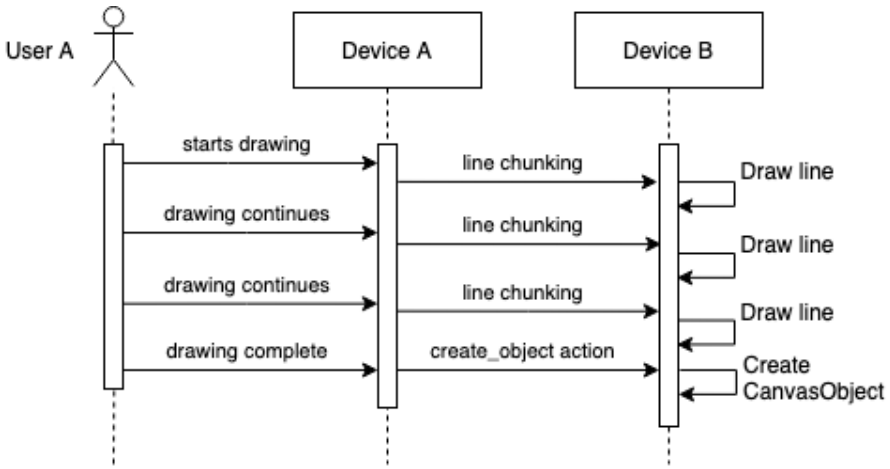


Figure 33: Object creation replication

Figure 33 shows the sequence of events that occur as lines are streamed between devices. User A draws a line, which is streamed in chunks and replicated on Device B. When User A has completed drawing the line, a *create_object* message containing a JSON-serialized `CanvasObjectAction` object is sent from Device A to Device B. As with other `CanvasObjectAction` instances, *create* actions are handled within the `CanvasObjectHandler`. Chunked lines are then deleted from the receiving users `WhiteboardDraw`, and replaced with the lines from the `CanvasObject` within the `CanvasObjectAction`. A `CanvasObject` is then made using the id from the action object, and the correct lines from the users `WhiteboardDraw`. This removes the many, smaller `Line`-object chunks that combinedly represent the drawn line, with the actual continuous line drawn by the user. After this process has been completed, the whiteboards of both Device A and Device B will display identical lines, which are both encapsulated by local `CanvasObject` instances.

5 Method

This section presents the research methodology for this project. It includes a presentation of the research strategy, the chosen data generation methods with a description of the monitoring system developed for the test and the approach for analyzing the collected data.

5.1 Overall research strategy

The research was conducted using the design and creation strategy [3, p. 108]. The strategy allows researchers to analyze, design, and develop an IT product such as a platform. New IT products are called *artifacts* [3, p. 108]. The IT artifact created for this project, presented in Section 4, is an *instantiation* [3, p. 108] of the design presented in Section 3. Oates emphasizes that projects within the design and creation strategy contribute to knowledge by either being the main focus of the research, being a vehicle for something else, or being a tangible end-product of a project where the focus is on the development process [3, p. 109]. The research conducted in this paper contributes to knowledge by creating an application that is a vehicle for something else, namely to discuss the collaborative patterns that occur when using this platform.

5.2 Data generation and evaluation

The design and creation strategy often makes use of different data generation methods to find out how people evaluate the IT artifact created [3, p .117]. A test with groups of pupils was designed to generate data to answer RQ3. Observations were used as the primary data generation method. In addition to observations, interviews were used to support the observations by eliciting feedback from the participants.

The participants of the test were pupils from the 5th and 7th grades. The pupils were divided into three groups, each consisting of two pupils. They were chosen and placed into the groups by their teacher, who was identified through communication with Matematikksenteret. Matematikksenteret recommended choosing pupils in those grade levels, as one could with more certainty guarantee a good level of maturity and technical aptness. The teacher chose pupils based on their aptness, availability, and interest in participating in the test. Groups were constructed so same-grade pupils who already knew each other were grouped. They were grouped like this because it was assumed that knowing the other pupil would ease collaboration. Since the goal of the test was to see what patterns of collaboration would occur and not to see if the platform itself would enable collaboration, this was evaluated as a fair way to construct the groups. The sampling frame, meaning the population of pupils that could have been included, was the pupils from the 5th and 7th grades the teacher taught and deemed apt to participate in the test. The sampling technique executed for the data generation was non-probabilistic convenience sampling [3, p. 98] because the teacher chose based on the availability of the pupils in the sampling frame.

Each group was asked to perform three or four tasks, depending on how much time they spent on each task. The order of the tasks was shuffled and was therefore different for each group. The test could only last for an hour per group due to limitations in their schedule. The test schedule, including the order of the tasks and instructions given to the teacher, can be seen in Appendix E. Groups were expected to complete a minimum of three tasks within the allotted time. A distribution that ensured that all tasks were performed at least twice was chosen. Tasks marked in red in the schedule were dropped if necessary to avoid delays.

The tasks in the platform were adapted from Mattelist[27] and chosen based on their required type of interactivity. For instance, one task would require more drawing in the shared workspace, while another would require moving specific pre-generated objects around. A variety of tasks was preferred to potentially enable different collaboration patterns since tasks themselves were assumed to be of influence. Before the test, the teacher was consulted to make sure the tasks chosen were not too

[27] https://www.mattelist.no/

difficult for the pupils.

5.2.1 Observations

The observations were conducted *overtly* [3, p. 208], meaning the participants knew that they were being observed. The type of observation used was *participant observation* [3, p. 209]. It was chosen over *systematic observation* because defining particular types of events of interest to observe did not apply to observing what type of collaboration Flate would enable. Meaning there was no preconceived notion of the collaboration patterns that would ensue. The type of participant observation used was *complete observer*. Everything that occurred was observed. Other than minimal formalities to make sure the observations would go smoothly, there was no participation in the proceedings from the authors. To ensure *inter-observer reliability* [3, p. 206] between the authors, pilot runs of the test were conducted beforehand.

The test was conducted with one group at a time. During the test, a teacher was present with the pupils to assist them but was asked to be reserved unless they needed help. The pupils were in the same room, but they had their backs to each other to ensure that they would only look at their tablets. Because of this, the implemented open-microphone solution in the platform was not used by the pupils to converse. Due to not being physically on-site, two methods were deployed to enable the observation:

1. A monitoring system was created to visualize interactions in Flate by each pupil in a group
2. An audio link to the pupils was used to take field notes of the conversation between them

Monitoring system
The monitoring system was created in Python (version 3.6.9) using Matplotlib[28] for the visualization. It uses the data stored in the database described in Section 4.3.2.

Each second, the system retrieves and visualizes the hundred latest user actions registered in the database for a specific group. In Figure 34, a frame of the monitoring system from a pilot test run is shown. The red arrow in Listing 1 indicates the example action.

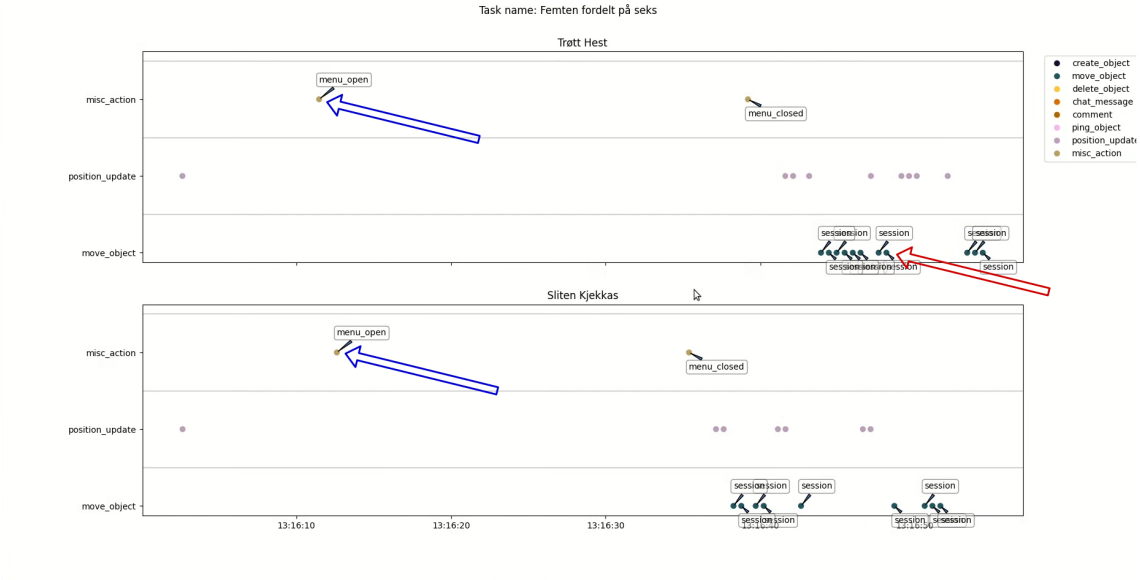


Figure 34: A frame from the monitoring system from the pilot test

In the monitoring system plot, each user_action has a predesignated lane. Visualization of actions occurs as a point in specified lanes, with a designated color corresponding to the legend in the plot.

[28] https://matplotlib.org/

The actions will appear in order with *create_action* as the bottom-most lane and *misc_action* as the top-most lane. The lane of a user_action will only appear if at least one point is plotted in that lane. If an action includes additional information, it will be plotted as an annotation connected to the point. For instance, the action in Listing 1 includes information about the object's author, which is plotted as an annotation.

The visualized sessions were recorded using screen recording, allowing them to be used in further analysis. In addition, the log data from a session was persisted in the database. This log data includes actions that were deemed unnecessary to monitor. Examples of unnecessary actions include rotating, grouping, pressing, and locking objects. These actions, and the actions plotted in the monitoring system, can be used to visualize all the actions done during a session. This could be characterized as the historical view of a session. An example of a historical plot of a session from the monitoring system is shown in Figure 35. The historical plot is similar to the live version of the monitoring system, except that it includes additional user_actions. Also, using functions provided by the plotting function of Matplotlib, one can zoom in and out on the historical plot.

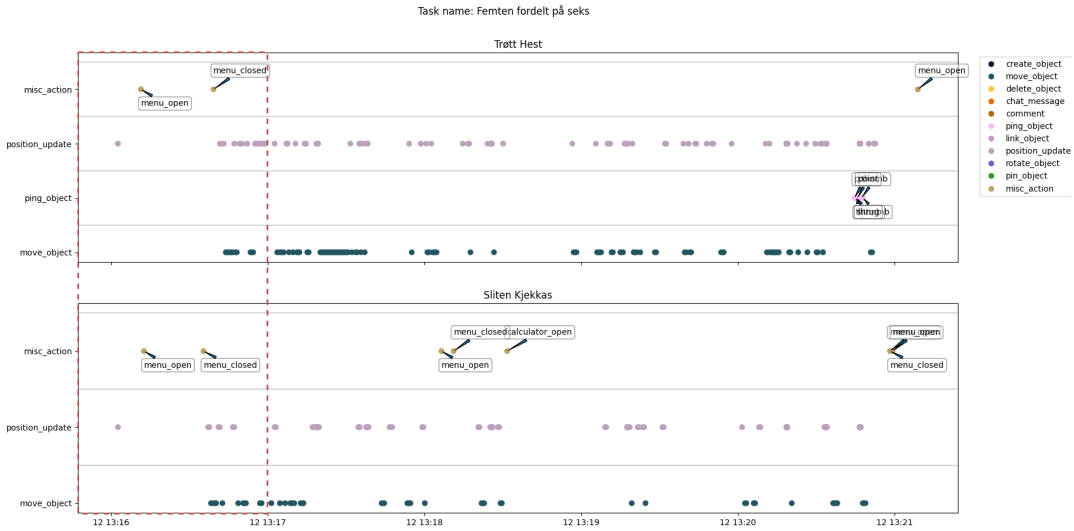


Figure 35: The historic plot of the monitoring system from the pilot test. The section marked in red is the frame shown in Figure 34.

Field notes

Field notes [3, p. 210] of the conversation between the pupils were taken to contextualize the plots created by the monitoring system. The notes were either verbatim quotes or discussed topics and included a timestamp. For quotes, an alias was included to link the quote to the pupil. Below is an example of a field note taken during the same pilot test previously mentioned:

> *13:16:10* Coordinating between themselves that they are going to read the task description

Aliases were not relevant for this note, as it presents a topic the participants discussed. This note explains why both participants opened the menu around 13:16:10 in Figure 34, indicated by the blue arrows.

Description of tasks

As mentioned above, the tasks were adapted from Mattelist and were chosen based on the type of interactivity they required. Table 2 shows an overview of these tasks. The tasks are presented with their Norwegian names, description, pre-rendered objects related to the tasks, and the users' expected behavior in terms of interactivity with the platform.

Task name	Task description	Pre-rendered objects	Expected behaviour
Byggeklosser	In this task you will see multiple three-dimensional figures. How do you think they will look like from another angle? Draw at least two different angles for each figure	Multiple cards with 3-D figures.	Users draw the figures from different angles.
Femten fordelt på seks	Karen had 15 cards with the numbers from 1 to 15. She distributed the cards in 6 piles. The sum of the numbers in each pile was equal in all the piles. What was this sum? How can the cards be distributed?	15 cards with the numbers from 1 to 15.	The users move the pre-rendered objects to create the piles. Users might use in-built calculator to find the sum.
Grublis	How many calculations can you come up with whose final result is 81? Find at least 10 ways to do this using different methods like addition, multiplication, etc. Try not to use the same method multiple times.	One card with the number 81.	Users draw different calculations.
Telle marihøner	Some kids played a game where they collected cards with ladybirds. Create a diagram that shows how many ladybirds each kid collected. What kind of diagram do you think fits best?	Cards for each kid with a number of ladybirds.	The users draw a diagram that represents how many ladybirds each pre-rendered object has.

Table 2: Description of the tasks in the platform

5.2.2 Interviews

In addition to the observation methods presented above, interviews were used to support the findings from the observations. The primary goal of the interviews was to elicit feedback regarding the collaboration from both the pupils and the teacher.

After each session, a *semi-structured* interview was carried out with the pupils in the group. Semi-structured interviews were chosen because of the opportunity to ask additional questions if the pupils brought up something of relevance. The questions were all related to a theme and were designed to prompt open-ended answers, allowing them to speak their minds. The themes that the questions were related to were: *application*, *collaboration*, and *awareness*. These were chosen to inquire the pupils about their subjective notions of the application, how they experienced collaborating using it and if they felt aware of each other during collaboration. A *structured* interview was not applicable as it would require the pupils to answer in a pre-coded answer. An *unstructured interview* was not chosen because the authors would have had less control of the themes raised during the interview.

After having completed the tests with all the groups, a *semi-structured* interview was conducted with the teacher. The goal of having an interview with the teacher was to assess how the teacher perceived the pupils' engagement with Flate and their collaboration. Additionally, questions about if the teacher could imagine continuing using the platform and what worked well with the platform were asked.

The responses from the pupils and the teacher can not be used to reach generalizable conclusions. However, they can be used to give an indication of what types of collaboration the platform facilitated. Table 3 and Table 4 showcase the questions asked to the pupils and the teacher. The questions were asked in Norwegian and have been translated for this thesis.

No.	Theme: Application
Q1	Was it fun?
Q2	Did you enjoy doing the tasks in the application?
Q3	What did you like most in the application?
No.	**Theme: Collaboration**
Q4	Was it fun to collaborate?
Q5	Do you feel like both of you contributed?
Q6	Did you distribute the work evenly?
Q7	Was it a good collaboration even though both of you had your own tablets?
No.	**Theme: Awareness**
Q8	Did you see what the other person was working on?
Q9	Were you ever unsure about what the other person was working on?
Q10	Was it easy to notice when the other person was working in the canvas?

Table 3: Questions to the pupils, separated into themes

Question	Question description
Q1	How do you think the test went?
Q2	What did you think of the collaboration between the pupils
Q3	How did you experience the engagement of the pupils?
Q4	Would you use such a platform in your teaching? Why, or why not?
Q5	What worked well?
Q6	What could have been better?

Table 4: Questions to the teacher

5.3 Analysis

As presented above, data was primarily collected through observations of the tests. The data consisted of non-textual data from the monitoring system and textual data from the field notes. In addition to the data from the observations, textual data from interviews conducted were collected. A description of the data generation method is presented in Section 5.2.

The *inductive approach* [3, p. 269] was chosen for the analysis. In this approach, one goes into the field first and then extracts themes from the data collected. During the observations conducted for the tests, recordings of the monitoring system were taken. Historical plots of the sessions were also available after the tests from the monitoring system. These were used in conjunction with the field notes to analyze and extract themes and patterns. The interviews were used to explain particular behavior or enhance the understanding of how the pupils collaborated.

The themes and patterns were mainly extracted by looking for relationships within the plots from the monitoring system. If interesting patterns were noticed in the plots, field notes were used to indicate the conversations occurring between the pupils in the relevant time frame. Then, either the entire plot of the session or a specific time frame was extracted. Drawings (arrows, boxes, circles) were added to

the plots using *draw.io*[29] when necessary to emphasize the relationships within them.

Field notes were used to try to explain why a certain pattern appeared. Additionally, answers from the interviews with the pupils were used to get a general indication of how they collaborated and how aware they were of each other's actions. The interview with the teacher was used to see if anything was noticed about their collaboration that could help explain the occurrence of a pattern.

An approach to qualitative research is *grounded theory* [3, p.274], where researchers go into the field and then analyze the data to see what theory emerges. The grounded theory approach is also inductive, but it aims to create a theory. Therefore, this approach was not chosen for this thesis. This thesis is not concerned with generating a theory. It aims to give a descriptive account of the data generated from the tests and analyze it to describe the patterns that occurred (as presented with RQ3).

[29] https://app.diagrams.net/

6 Results

This section presents the results from the data generation described in Section 5.2.

Sections 6.1 - 6.3 describe results for groups 1 - 3, respectively. These sections begin by introducing how the test went overall for each group. A description of how it generally went when the group performed a task is also provided. This description includes excerpts of the field notes, which consist of direct quotes and explanations of conversations between pupils.

In Section 6.1.1, Section 6.2.1, and Section 6.3.1, the answers to the semi-structured interviews with each group of pupils are shown. In Section 6.4, the interview conducted with the teacher that supervised during the test is presented. The answers are translated to English from Norwegian for this thesis.

In Section 6.1.2, Section 6.2.2, and Section 6.3.2, the notable patterns that occurred are displayed. Section 6.5 categorizes and discusses the main patterns observed from the notable patterns. This is related to answering RQ3 by explaining the patterns that occur when using the platform.

During the test with the first group, it became evident for the teacher that the pupils did not read the task's description thoroughly. Therefore, the teacher explicitly encouraged group 1 to do so during their second task and continued doing this with every group.

6.1 Group 1

Most of group 1's collaboration was coordinated through non-verbal communication. After the group had completed the test, the teacher noted that they often waited and looked at what the other was doing before embarking on individual sub-tasks.

Below is an ordered list of their tasks, the time they spent on completing them, and the pupils' aliases for each task.

1. **Byggeklosser**
    - Time: circa 8 minutes.
    - Aliases: Legendarisk Kulturmelk and Flau Professor.
2. **Femten fordelt på seks**
    - Time: circa 8 minutes.
    - Aliases: Introvert Smarting and Lei Solnedgang.
3. **Grublis**
    - Time: circa 5 minutes.
    - Aliases: Lang Lemmen and Syrlig Moped.

Byggeklosser
For task 1, the group required some assistance from the teacher in the beginning and during the exercise. Other than Lei Kulturmelk announcing once at 08:59:30 that they had been able to complete drawing a block, *"I was able to draw the purple one"*, there was not much communication between the pupils in the group. The monitoring system showed that they only deleted their *own* lines and that there was a pattern emerging in their collaboration in the create_object messages.

Femten fordelt på seks
In task 2, there was more communication between the pupils. The teacher encouraged that one of the pupils read the task aloud, which Lei Solnedgang then proceeded to do. Lei Solnedgang then suggested how to solve the task, and the pupils discussed the tasks amongst themselves. They found a solution to the task together. The monitoring system showed clear breaks where they discussed and agreed on a tactic. The work seemed to be pretty evenly distributed.

Grublis
During task 3, the communication between the pupils was characterized by confirming the solutions they had concurred with individually. For instance, in the very beginning, after reading the task description:

> *09:15:35*
> Syrlig Moped: "*9 times 9?*"
> Lang Lemmen: "*Yes.*"
> Syrlig Moped: "*Should we write it down?*"

At 10:17:00, they talked to each other about possible solutions, and at 10:18:00, they asked each other about what they had written. At around 10:18:00, one could see from the monitoring system that Legendarisk Kulturmelk moved around in the workspace, marked in Figure 37 with a blue circle. This action could be Legendarisk Kulturmelk moving to where Syrlig Moped was in the workspace to compare their calculations. Other than this, a pattern reminiscent of what was noticed in task 1 was prevalent throughout this session.

6.1.1 Interview

No.	**Theme: Application**	**Answers**
Q1	Was it fun?	Yes.
Q2	Did you enjoy doing the tasks in the application?	Yes.
Q3	What did you like most in the application?	A lot of stuff. Mainly that it was easy to use. However, we wish we could write text using the keyboard.
No.	**Theme: Collaboration**	**Answers**
Q4	Was it fun to collaborate?	Yes.
Q5	Do you feel like both of you contributed?	Yes.
Q6	Did you distribute the work evenly?	Pretty evenly. We did so by talking with each other. We also distributed the work by watching what the other person did.
Q7	Was it a good collaboration even though both of you had your own tablets?	*Not asked*
No.	**Theme: Awareness**	**Answers**
Q8	Did you see what the other person was working on?	Yes.
Q9	Were you ever unsure about what the other person was working on?	If we were, we would just watch what the other person did.
Q10	Was it easy to notice when the other person was working in the canvas?	Yes.

6.1.2 Patterns

The most notable patterns were observed during Byggeklosser (task 1) and Grublis (task 3). These are presented in Figure 36 and Figure 37. Each red circle in the figures contains a cluster of create_object messages from a pupil. These clusters span a length of 30 seconds or less. Each message in a cluster is assumed to be related to each other. Meaning a cluster represents some work produced, and each message in a cluster relates to that work.

From the figures, one can see that the clusters from each pupil form a pattern when related to each other. The prominent characteristic of this pattern is that clusters from a pupil are either completely non-overlapping or almost non-overlapping with clusters from their partner.

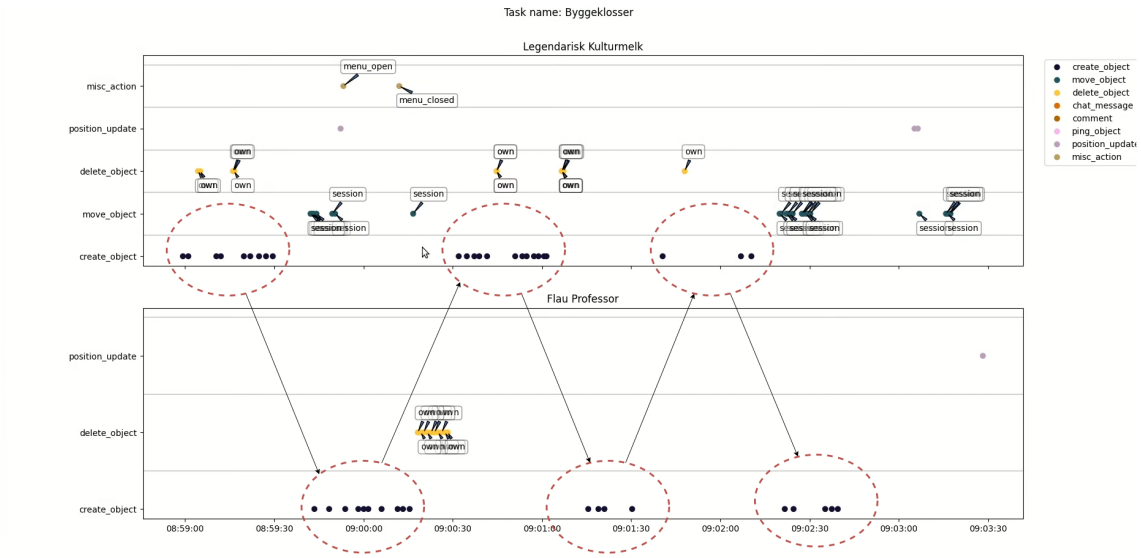


Figure 36: Cluster messages (red circles) forming a pattern for group 1 during task 1: Byggeklosser.

In Figure 36, there seems to be more space between the clusters of the two pupils than in Figure 37. Particularly the second and third clusters of Lang Lemmen and Syrlig Moped overlap more than their first clusters. Even though there are slight differences, the patterns shown in the figures are still similar.

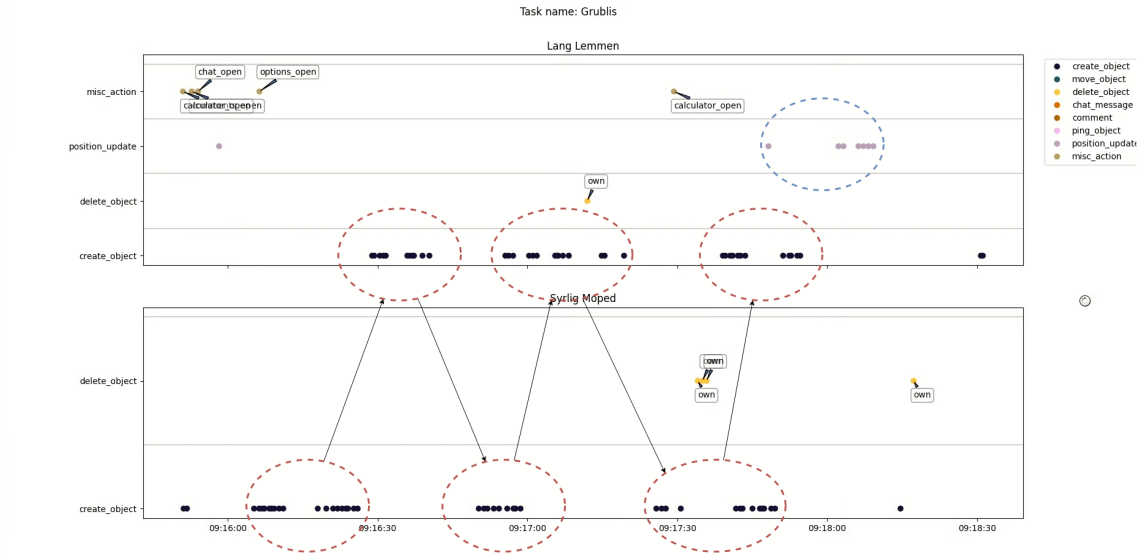


Figure 37: Cluster messages (red circles) forming a pattern for group 1 during task 3: Grublis. The blue circle marks that Legendarisk Kulturmelk moved around in the workspace after both pupils asked each other about what they had written.

6.2 Group 2

The pupils in group 2 communicated more verbally than the previous group. However, like the group before them, each pupil also looked at what the other pupil was doing to figure out what they could do to complete the task.

During the test with this group, a bug within the platform occurred several times. This bug was related to drawing and appeared during all tasks except during the third. From Table 2, one can see that the expected behavior from the pupils for *Femten fordelt på seks* does not involve them drawing in the canvas. This could explain why the drawing bug did not appear during this task.

Below is an ordered list of their tasks, the time they spent on completing them, and the pupils' aliases. For task 2, the group used two attempts. When the bug occurred during the second task, the group tried to restart the platform to see if it helped, thus starting a second attempt. This group was able to complete four tasks during the allocated time.

1. **Grublis**
    - Time: circa 5 minutes.
    - Aliases: Mellomstor Sykkel and Intern Trønder.
2. **Byggeklosser**
    - Time: circa 3 minutes on the first try and 2 minutes on the second try.
    - Aliases: Intensiv Sjøstjerne and Perfekt Smarting on the first try. Sliten Såpe and Pratsom Sveis on the second try.
3. **Femten fordelt på seks**
    - Time: circa 7 minutes.
    - Aliases: Syrlig Ekspert and Intern Sei.
4. **Telle marihøner**
    - Time: circa 10 minutes.
    - Aliases: Intensiv Lemmen and Ordentlig Purre.

Grublis
During task 1, the communication between the pupils was fluid. Each pupil would announce suggestions to each other. They would then either delegate suggestions to the other pupil to write down or write them down themselves. From the monitoring system, one could notice overlapping create_object messages. They seemed to solve the task by solving sub-tasks individually since the create_object messages were parallel. There were not a lot of position_update messages, and they only deleted their own lines. At the end of the task, the drawing bug occurred, but they moved along to the next task since they had completed it.

Byggeklosser
In task 2, the drawing bug occurred before the task had been completed, so the group restarted the platform. Restarting did not help, as the bug occurred again. Either way, the group almost completed this task before the bug happened again.

The collaboration during this exercise was seemingly characterized by delegating the figures amongst themselves and drawing them independently. This could be seen in the monitoring system, where the second try of the session starts with the users moving the different figures around. After having moved them in the shared workspace, both users started drawing in parallel. They also only deleted their own lines.

The pupils spent some time during the beginning of the second try of the session to redraw the figures they had drawn the previous try. When doing this, there was some explicit communication when moving the objects, exemplified with this quote by one pupil:

09:54:10 Sliten Såpe: "*I will bring the red one down here.*"

This action could be seen in the monitoring system, as there were move_object messages during that time produced by Sliten Såpe.

Femten fordelt på seks

The execution of task 3 involved a good amount of communication between the pupils. Intern Sei suggested a method to solve the task, and they distributed piles of cards amongst themselves. After figuring out the sum that had operated with (15) was wrong, they started over and moved the cards back:

10:03:30 Intern Sei: "*I can just move everything back up here again.*"

The pupils then gave it a new try. Together they dove in headfirst and tried moving the cards to figure out what the sum could be. At around 10:04:45, they opened the calculator, but it seemed that only Intern Sei used it from the conversation that occurred. Within thirty seconds of each other, they were able to find the solution independently. Syrlig Ekspert found the solution by trying out 20 as the sum, while Intern Sei used the calculator.

10:05:30 Syrlig Ekspert: "*What if we try 20?*"

10:06:00 Intern Sei: "*120 divided by 6. We need 20.*"

Telle marihøner

The pupils spent some time at the beginning of task 4 announcing how many ladybirds there were in each card and summed it up using the calculator. Then, led by Ordentlig Purre, they had to coordinate how to proceed:

10:10:15 Ordentlig Purre: "*Zooming out... Drag them over to this side, and then we can make them over here.*"

10:11:00 Discussion about how they were going to create a diagram.

10:11:40 Ordentlig Purre: "*Drop it there.*"

These activities can be seen in the monitoring system with Ordentlig Purre moving session objects at circa 10:10:15 and Intensiv Lemmen moving the created objects at circa 10:11:40. It seems that Intensiv Lemmen started drawing when Ordentlig Purre started moving the cards around, and Ordentlig Purre informed that the diagram should be moved to the new placement of the cards.

Approximately two minutes after that, the drawing bug appeared. However, they were able to complete the task using the polygon drawing tool, which worked as a loophole.

6.2.1 Interview

No.	Theme: Application	Answers
Q1	Was it fun?	Yes.
Q2	Did you enjoy doing the tasks in the application?	Yes.
Q3	What did you like most in the application?	That we could collaborate with each other.
No.	**Theme: Collaboration**	**Answers**
Q4	Was it fun to collaborate?	*Not asked*
Q5	Do you feel like both of you contributed?	Yes.
Q6	Did you distribute the work evenly?	Yes.
Q7	Was it a good collaboration even though both of you had your own tablets?	Yes.
No.	**Theme: Awareness**	**Answers**
Q8	Did you see what the other person was working on?	*Not asked.*
Q9	Were you ever unsure about what the other person was working on?	Yes, by using the bubble. Had to look for it a bit in the beginning, but found it after a while.
Q10	Was it easy to notice when the other person was working in the canvas?	*Not asked*

6.2.2 Patterns

The most notable patterns for group 2 were observed during Grublis (task 1) and Femten fordelt på seks (task 3). These are presented in Figure 38 and Figure 39.

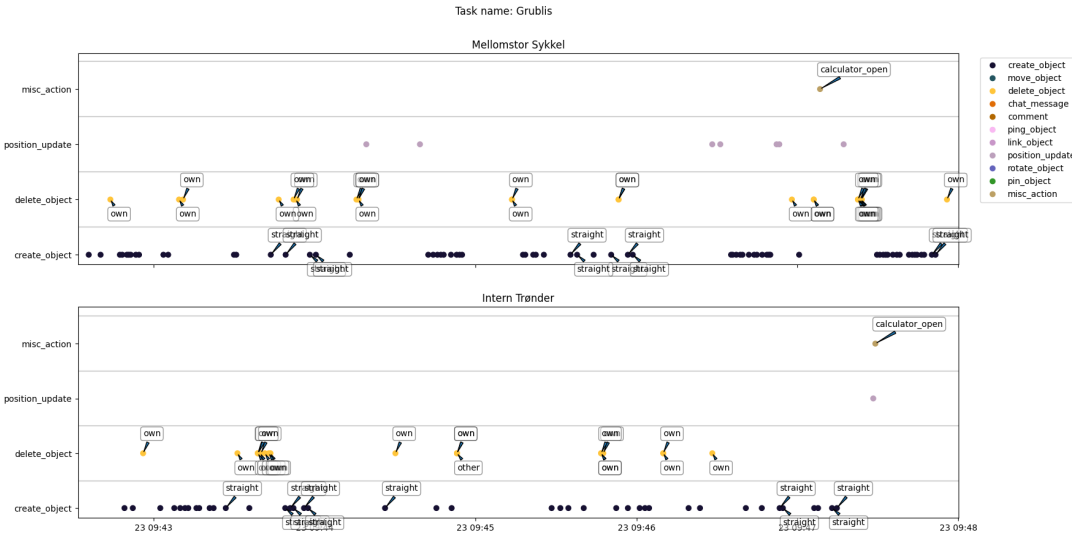


Figure 38: Parallel workflow using the same method to complete the task for group 2 during task 1: Grublis.

In Figure 38, the entire Grublis (task 1) session of group 2 is shown using the historical view of the monitoring system. The pattern in this session is not obvious, as there seems to be no apparent relationship between the create_object message clusters between the two pupils. Rather, the figure could indicate that the pupils worked in parallel during the whole session when collaborating. The workflow being described as parallel in Paragraph *Grublis* substantiates this. Even though they worked in parallel, both wrote down calculations, i.e., used the same method to complete the task.

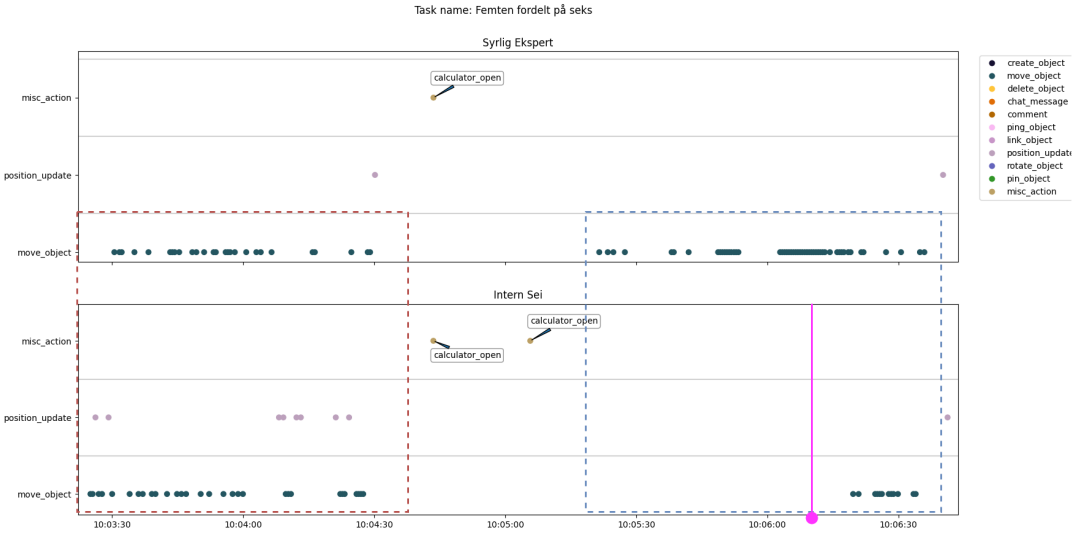


Figure 39: Parallel workflow using the same method to complete the task (red box), and a mostly independent workflow not using the same method (blue box) for group 2 during task 3: Femten fordelt på seks.

In Figure 39, group 2 went from working in parallel (red box) to working independently (blue box). The figure shows approximately the last three minutes of the group solving Femten fordelt på seks (task 3). As noted above in Paragraph *Femten fordelt på seks* in Section 6.2, the group spent the time from 10:03:30 until approximately 10:04:45 trying to figure out the sum by moving the cards. This part of the collaboration is marked in the figure with a red box. Then, after 10:05:00, Syrlig Ekspert tried moving the cards to see if the sum could be 20. Intern Sei decided to contribute after confirming that 20 was the correct answer by using the calculator. This part of the collaboration is marked with a blue box, with the moment Intern Sei confirms the sum marked with a pink line.

6.3 Group 3

The pupils in group 3 verbally communicated actively with each other and often distributed tasks explicitly. The drawing bug mentioned in Section 6.2 also appeared during the execution of task 1.

Below is an ordered list of their tasks, the time they spent on completing them, and the pupils' aliases.

1. **Telle marihøner**
   - Time: circa 5 minutes.
   - Aliases: Utmerket Whippet and Introvert Fornøyelse.
2. **Grublis**
   - Time: circa 6 minutes.
   - Aliases: Søt Forsker and Lav Modell.
3. **Femten fordelt på seks**
   - Time: circa 8 minutes.
   - Aliases: Trøtt Smarting and Introvert Bris.

Telle marihøner
From the beginning of task 1, the pupils actively communicated their thoughts. After Intovert Fornøyelse read aloud the description of the task, they together figured out how they wanted to solve it:

> *10:32:30*
> Introvert Fornøyelse: "*We can create those...*"
> Utmerket Whippet: "*Bar charts.*"
> Introvert Fornøyelse: "*Yes, that is what I was thinking of!*"

They were also actively announcing tasks to do:

> *10:33:00*
> Introvert Fornøyelse: "*We have to count.*"
> Utmerket Whippet: "*Yes.*"
> Introvert Fornøyelse: "*We can write up how many each card has so that we do not forget it.*"

After writing up how many ladybirds there were in each card, Utmerket Whippet announced: "*The highest number is 10.*" Now that this was clear, they proceeded to complete their bar chart. During this, they assigned themselves to duties:

> *10:35:30*
> Utmerket Whippet: "*I will write letters.*"
> Introvert Fornøyelse: "*I will write numbers.*"

The drawing bug appeared after delegating tasks, and Introvert Fornøyelse could not interact with the platform anymore. They then proceeded with Introvert Fornøyelse guiding Utmerket Whippet, who could still write, and completed the task.

Grublis
In task 2, Lav Modell distributed duties to Søt Forsker. Lav Modell suggested that they do five calculations each and delegated operators between them. There was not much communication between the pupils after this regarding the task. This could be because they explicitly assigned tasks that presumably could be done independently of one another. The pupils did announce when they had completed a calculation. For instance, during the creation of a second calculation, Søt Forsker announced:

> *10:41:30* Søt Forsker: "*Should be correct, I just have to double-check it.*"

From the monitoring system, one can see that after 10:40:20, a pattern emerged that was reminiscent of the pattern seen with group 1. From around 10:41:15, the create_object messages seemed to appear more in a parallel manner.

Femten fordelt på seks
During task 3, the pupils communicated actively to solve the task. The beginning was characterized by exchanging ideas to each other on how to solve the task:

> *10:47:40* Trøtt Smarting: "*There has to be more than 15 in each pile since one of the cards is 15.*
> *10:48:00* Introvert Bris: "*Don't we have to divide it by 6 in some way?*"

While trying to find a strategy to solve the task, one could in the monitoring system see that only Trøtt Smarting was moving the cards around. After getting input from the teacher that the sum is helpful when solving the task, the group agreed on a strategy:

10:48:40 Agree on trying 18 as the sum for each pile of cards

When solving the task using this strategy, a pattern similar to the one seen before in the move_object messages emerged. However, using 18 as the sum is not the right answer. After a while, the group realized this, and at 10:48:25, Trøtt Smarting said: "*This does not add up. Probably not 18.*" They then tried out a new strategy:

10:51:00

Trøtt Smarting: "*Have to sum up all the cards to find the sum.*"

Introvert Bris: "*I can do it.*"

Trøtt Smarting: "*I am already doing it here.*"

In the monitoring system, one can see that both of them opened their calculators. Trøtt Smarting was approximately 10 seconds ahead of Introvert Bris in opening it. Trøtt Smarting announced: "*It is 20 then*", after dividing the sum of the cards by 6. The conversations then progressed to solving the task with this information.

10:52:20 Introvert Bris: "*I will do it up here in the left.*"

During this time, one can from the monitoring system see position_update messages from both users, indicating that they had moved in the shared workspace. The work from 10:52:10 to 10:53:40 could be divided into two. Up until circa 10:53:00, most of the work had the pattern shown in Section 6.1.2. This pattern can be seen in the move_object messages. Only Introvert Bris produced move_object messages towards the end, meaning this pupil moved the last cards into piles.

6.3.1 Interview

No.	Theme: Application	Answers
Q1	Was it fun?	Yes. But there is room for improvement. Maybe you could add it so that we can put in tables, that lines could be straightened, and that we do not have to go out of pen-mode to zoom in/out.
Q2	Did you enjoy doing the tasks in the application?	Yes.
Q3	What did you like most in the application?	It was something else than working on paper. We could write and build things ourselves. It also had tasks we did not get tired of.

No.	Theme: Collaboration	Answers
Q4	Was it fun to collaborate?	Yes. We could also collaborate from home with this. The teacher could divide us into groups, and we could get our homework through this app.
Q5	Do you feel like both of you contributed?	Yes. We distributed tasks between ourselves. For instance, in Marihøner one of us counted while the other one wrote down letters. Sometimes the distribution of tasks happened automatically.
Q6	Did you distribute the work evenly?	Yes.
Q7	Was it a good collaboration even though both of you had your own tablets?	*Not asked*

No.	Theme: Awareness	Answers
Q8	Did you see what the other person was working on?	Yes, we were on the same "page".
Q9	Were you ever unsure about what the other person was working on?	No. We could see what the other person was working on by, for example, watching the border that appeared when object were moved.
Q10	Was it easy to notice when the other person was working in the canvas?	Yes, except for the calculator. Maybe you could have it so that we could see when the other person opened the calculator.

6.3.2 Patterns

Notable patterns were observed during each task for group 3. These are presented in Figure 40, Figure 41, and Figure 42.

Figure 40: Parallel workflow to a guided workflow for group 3 during task 1: Telle marihøner. The break point is indicated with a pink line.

Figure 40 shows the session for Telle marihøner (task 1). Before the pink line, one can see that the interactions with the tablet were largely parallel. From Paragraph *Telle marihøner* in Section 6.3, one can see that the delegation of task happened explicitly. After the pink line, it was mainly Utmerket Whippet that drew in the platform. At the timestamp of where the pink line is, the drawing bug occurred for Introvert Fornøyelse, which prevented any further drawing action from them during that session. After that, the collaboration consisted of Utmerket Whippet drawing, while Introvert Fornøyelse contributed with suggestions on what to draw. This distribution is visible in the plot, as the actions Introvert Fornøyelse performed on the tablet are either moving the cards or moving around in the workspace.

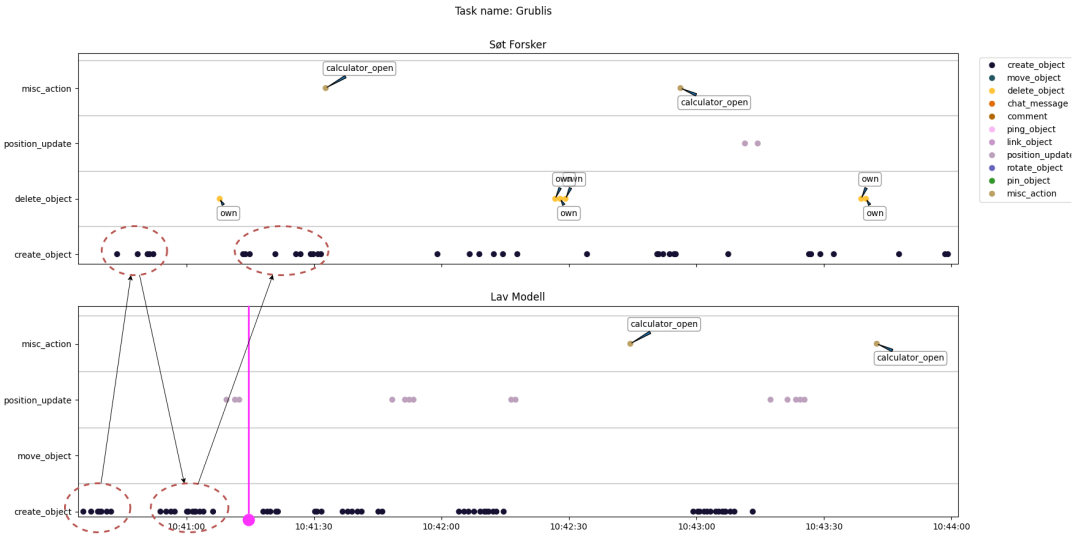


Figure 41: Pattern formed by cluster messages (red circles) to a parallel workflow for group 3 during task 2: Grublis.

Figure 41 shows an excerpt of the session for Grublis (task 2). At the beginning of the figure, one can see the same pattern mentioned in Section 6.1.2. As mentioned in Paragraph *Grublis* in Section 6.3, the create_object messages start overlapping at around 10:41:15 (pink line in the figure), and it seems the pupils started working more in parallel after that.

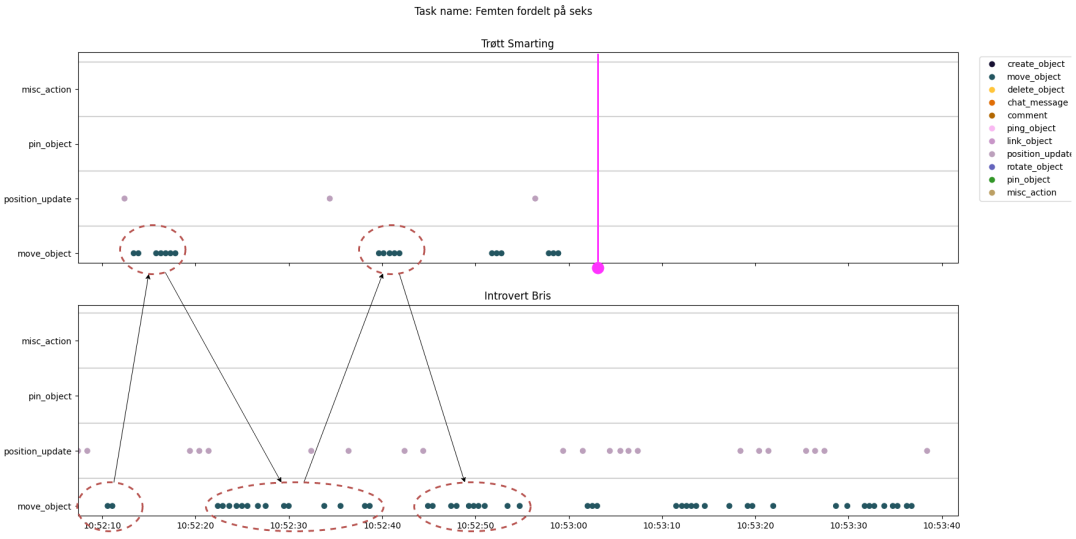


Figure 42: Pattern formed by cluster messages (red circles) to a singular contributor for group 3 during task 3: Femten fordelt på seks.

In Figure 42, the last 01:30 minutes of the session for Femten fordelt på seks (task 3) is presented. As noted above in the Paragraph *Femten fordelt på seks* in Section 6.3, the work during the time from 10:52:10 to 10:53:40 can be divided into two. This division is illustrated with a pink line in the figure. Before the pink line, one can see that the move_object message clusters form a pattern similar to the pattern displayed in Section 6.1.2. After the pink line, Introvert Bris moved the last cards into piles for approximately 40 seconds. There was nothing in the conversation that would have initiated this change in the workflow.

6.4 Interview with the teacher

In this section, the semi-structured interview with the teacher conducted at the end is presented. This interview aimed to extract the perspective and observations of the teacher regarding the test, and the questions asked can be found in Table 4. Each subsection will reproduce the opinions of the teacher related to a question.

6.4.1 Q1: How do you think the test went?

The teacher expressed that it was an unusual situation for the pupils and that eventually, they got more comfortable in it. The teacher noted that the first group needed a bit more time to get used to the test than the others, which the authors also noticed. The teacher also observed that there was a bug with the platform.

6.4.2 Q2: What did you think of the collaboration between the pupils?

The teacher thought the pupils used the platform well. When speaking of the groups, the teacher said:

> "*The first group did not talk with each other that much, but they looked at what the other pupil was doing and 'corrected' their activity. The other groups talked more with each other to coordinate.*"

The teacher also expressed seeing the platform's value as a collaborative tool:

> "*I can absolutely see the potential it has. Solving the mathematical problem goes a little faster since a pupil does not have to orient themselves about what the other pupil in the group is doing. It is easier to see what the other pupil is doing since they are on the same 'page' and can just drag around.*"

The teacher further explained that a pupil usually has to ask more questions in a classroom setting to figure out what the other pupil had done. There was value in pupils being able to coordinate amongst themselves easily, which was more important than solving the task faster.

> "*The work is more efficient, as one does not get the usual 'how far along are you'-questions.*"

The teacher noted that objects getting locked when someone selects them caused less arguing between the pupils. In addition to this, the mobility a pupil has in the platform was something that the teacher said they would not get when writing on paper. The teacher added:

> "*One could just move stuff in the platform if one does not want it there.*"

It seems that from the teacher's perspective, the platform served its purpose in facilitating collaboration between the pupils. The difference in how verbal each group was did not necessarily affect how well they performed. The teacher also felt there was value in enabling pupils to coordinate the work non-verbally through the shared workspace.

6.4.3 Q3: How did you experience the engagement of the pupils?

The teacher explained that the pupils immediately tend to get more excited when something is not on paper. They get engaged when there is something digitally visual in front of them.

Elaborating on the engagement of the groups, the teacher explained that group 1 collaborated just as well as group 3 did, even though:

"*... the pupils in the former were not as talkative as the pupils in the latter.*"

The teacher said that the platform facilitated both types of collaboration:

"*the more non-verbal and the more verbal collaboration.*"

6.4.4 Q4: Would you use such a platform in your teaching? Why, or why not?

"*Yes, absolutely.*"

The teacher said that, in a way, the platform was pretty simple to use. Additionally, the teacher emphasized that it works well with open-ended tasks because:

"*one can think and move objects, and test out different possibilities.*"

From the teacher's point of view, It seemed like it was easy to collaborate, and if a pupil has to be home one day, they could still participate and be a part of a group. Lastly, the teacher added that there was potential for using the platform in a classroom setting.

6.4.5 Q5: What worked well?

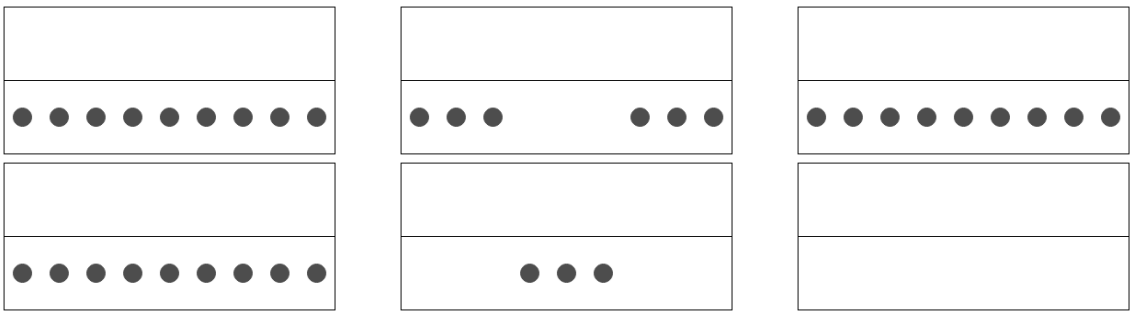
The teacher thought that the lobby worked well, as it was easy for the pupils to connect. The teacher also mentioned some other features in the platform, like commenting on objects and manipulating lines as independent objects. Pupils having the possibility to draw in their own area was mentioned as something positive by the teacher.

6.4.6 Q6: What could have been better?

The teacher mentioned the occurrence of the drawing bug and that it made the platform seem a bit fragile. The teacher also suggested several improvements to the platform, like being able to erase parts of a line, a "snap-to-grid" function, adding text to the shared canvas, and selecting multiple objects by clicking on them instead of using the multi-select tool.

6.5 Main patterns

The notable patterns that occurred for each group were used to identify the patterns of collaboration. Abstractions were made to visualize these patterns based on how they occurred in the monitoring system. This was done to describe and categorize the different types of collaboration that arose.



(a) Parallel (b) Ping-pong (c) Singular action

Figure 43: Main patterns of collaboration

The main patterns of collaboration identified are presented in Figure 43. Three types of collaboration were identified: *parallel*, *ping-pong*, and *singular action*.

Parallel collaboration
Parallel collaboration had the characteristic of parallel interactions in the platform. This type of collaboration happened when the pupils could delegate sub-tasks that could be solved at least somewhat independently of each other. The delegation of tasks happened before actions were taken in the platform. This seemed to happen explicitly through verbal communication, by either a pupil delegating the sub-tasks or the pupils announcing which sub-tasks they would perform. The sub-tasks themselves did not seem to cause the groups to lose coherence regarding the common task. In some sessions, the ability to have your own territory enabled the occurrence of this pattern, while in others, the pattern occurred regardless of this. This seemed to be somewhat dependent on the type of task that was solved in the session.

The parallel pattern of collaboration occurred when group 2 solved their first task Grublis. This can be seen in Figure 38, where this pattern was prevalent during the entire session in the create_object messages. During this task, the group solved sub-tasks independently and delegated these explicitly by speech. The sub-tasks were to create calculations by using operators they had delegated amongst themselves. Interestingly, they mainly deleted their own lines, indicating that they possibly had their own territory within the shared workspace. Even with this, the group announced suggestions to each other, indicating that the group did not lose coherence.

One can also see the pattern occur when group 2 solved their final task Femten fordelt på seks. The pattern can be seen in the red box of the plot in Figure 39 in the move_object messages. In this task, the delegated sub-task involved creating piles that could provide them with the correct sum. Even though they had individual sub-tasks, they could not work fully independently since all the piles had to add up to the same sum.

The pattern also occurred when group 3 solved their first task Telle marihøner. It can be seen in Figure 40, before the pink line. Unlike the previous two sessions discussed above, one must consider both the move_object and create_object messages to notice the parallel collaboration. The pupils delegated sub-tasks explicitly, and these consisted of counting up the number of ladybirds in each card and creating the axis of a bar chart. However, the drawing bug appeared at the pink line, effectively stopping the parallel collaboration.

Additionally, the pattern occurred when group 3 solved their second task Grublis. This can be seen in Figure 41, after the pink line in the create_object messages. Similar to when group 1 solved the same task, group 3 delegated operators and created calculations independent of each other. Before the pink line, the pattern of collaboration was different. It is interesting that right before switching their type of collaboration to parallel, one can see that the pupil Lav Modell moved around in the shared workspace. This could be because Lav Modell wanted their own territory, similar to how group 2 utilized the workspace for the same task. Additionally, each pupil announced when they had completed a calculation, indicating that this group also did not lose coherence when solving independent sub-tasks.

In Table 5, the occurrences of parallel collaboration are shown with links to the relevant figures and the tasks of where they occur.

Task reference	**Figure reference**	**Occurrence in the figure**
Group 2: Grublis	Figure 38	Entire plot in the create_object messages
Group 2: Femten fordelt på seks	Figure 39	Red box of plot in the move_object messages
Group 3: Telle marihøner	Figure 40	Before the pink line in the plot in the move_object and create_object messages
Group 3: Grublis	Figure 41	After the pink line in the plot in the create_object messages

Table 5: Figures and tasks where parallel collaboration occurred

Ping-pong collaboration
Ping-pong collaboration had the characteristic of alternating interactions in the platform. This type of collaboration happened when the pupils were dependent on what the other pupil was doing to complete their sub-task. The delegation of sub-tasks mostly happened implicitly, meaning it happened without any verbal communication. It usually seemed to happen by pupils picking the sub-task that the other pupil was not doing. For this to happen, there had to be an agreement on sub-tasks, and most of the time these were evident from the task description.

The ping-pong pattern of collaboration can be seen in the first and last task of group 1, which were Byggeklosser and Grublis respectively. The plots related to these tasks are Figure 36 and Figure 37, and the pattern can be seen in the create_object message clusters, marked by the red circles. In both figures, the clusters of messages are alternating between the two pupils. This could indicate that after drawing a 3-D figure or writing down a calculation, the pupils waited to see what the other pupil was doing in the platform. The teacher present during the test noted in Section 6.4.2 that group 1 were not that talkative, but "*...they looked at what the other pupil was doing and 'corrected' their activity*".

In the test with group 3 the pattern occurred in the second and last task, which were Grublis and Femten fordelt på seks respectively. The plots related to the task are Figure 41 and Figure 42. In both figures, the pattern occurs before the pink line, and the relevant message clusters are indicated with red circles. Interestingly, the pattern occurred when the group worked on Grublis even though they had explicitly delegated sub-tasks between them. This might indicate that before the pink line, the pupils chose to watch what each other were doing, even though they had delegated explicitly. After this short period of watching what the other did, they changed the type of collaboration, explaining why the parallel pattern was the most dominant for the rest of the session.

When group 3 was working on Femten fordelt på seks, they agreed slightly before 10:52:10 to create piles with 20 as the sum. Creating the piles were the sub-tasks in this case. Therefore, it makes sense that the pattern occurred before the pink line in the plot, since the pupils had to be aware of the sub-task the other pupil was doing to solve their own. The sub-tasks were implicitly delegated, meaning the pupils chose a sub-task by coordinating through the shared workspace non-verbally. They likely picked a sub-task by seeing, and sometimes foreseeing, what was available.

In Table 6, occurrences of ping-pong collaboration are shown with links to relevant figures and tasks in which they occur.

Task reference	Figure reference	Occurrence in the figure
Group 1: Byggeklosser	Figure 36	Entire plot in the create_object messages
Group 1: Grublis	Figure 37	Entire plot in the create_object messages
Group 3: Grublis	Figure 41	Before the pink line in the plot in the create_object messages
Group 3: Femten fordelt på seks	Figure 42	Before the pink line in the plot in the create_object messages

Table 6: Figures and tasks where ping-pong collaboration occurred

Singular action collaboration
Singular action collaboration had the characteristic of only one pupil performing actions in the platform. This type of collaboration is identified in three of the notable patterns from the tests, happening for different reasons each time.

The pattern occurred for group 2 during their second task, Femten fordelt på seks. This can be seen in Figure 39 in most of the blue box, made evident by the move_object messages. This pattern occurred after the pupils decided to verify the sum using different methods; one pupil placed the cards into piles, while the other used the calculator. After the pink line, the second pupil started moving cards into piles after verifying the sum with the calculator. Interestingly, this pupil was able to join in on moving cards without needing any conversation to get oriented in what had been done already. This type of *assistance* is characterized in the third part of the workspace awareness framework presented in Section 2.3.1. It emphasizes that workspace awareness enables users to provide the assistance deemed appropriate by assessing the situation, which seems to be what the pupil did.

For group 3, this pattern occurred when solving Telle marihøner (task 1) and Femten fordelt på seks (task 3). In the former, the pattern occurred in the create_object messages after the pink line in Figure 40. At around the timestamp of the pink line, the drawing bug appeared and prevented one of the pupils from writing in the platform. Because of the bug, they proceeded with one pupil guiding while the other wrote in the platform. This is reminiscent of the software development technique pair-programming[30], where two programmers work together on one editor.

When group 3 solved Femten fordelt på seks, their third task, this pattern occurred in the move_object messages after the pink line in Figure 42. In this case, there was nothing in the conversation to initiate this change in the collaboration. Also, there was no difference in methods to verify parts of the task like with group 2. Instead, it seems that it occurred naturally. Meaning, since there only were a couple of cards left to move into piles, the pupil Trøtt Smarting decided not to cause conflict by letting the other pupil finish the task.

In Table 7, the occurrences of singular action collaboration are shown with links to the relevant figures and the tasks of where they occur.

Task reference	**Figure reference**	**Occurrence in the figure**
Group 2: Femten fordelt på seks	Figure 39	Most of the blue box in the plot
Group 3: Telle marihøner	Figure 40	After the pink line in the plot
Group 3: Femten fordelt på seks	Figure 42	After the pink line in the plot

Table 7: Figures and tasks where singular action collaboration occurred

Analysis
The main patterns discussed were identified by looking for occurrences of notable patterns within each group. Group 1 and 2 had notable patterns in two of the tasks they completed, while group 3 had notable patterns in three tasks. Each group did not exhibit all of the main patterns in their notable patterns. Table 8 shows the patterns the different groups exhibited.

Group	**Parallel**	**Ping-pong**	**Singular action**
Group 1	*No*	Yes	*No*
Group 2	Yes	*No*	Yes
Group 3	Yes	Yes	Yes

Table 8: Table of the patterns the groups exhibited

In the semi-structured interviews conducted with the groups after their test session, every group expressed that they felt that both pupils in the group contributed to the work. Additionally, they all said that they felt they distributed the work evenly within the group. This distribution could indicate that the main patterns of collaboration mentioned above did not skew the workload within the groups. Alternatively, at the very least, the groups exhibiting the patterns did not experience the distribution of work as unfair.

Group 3 said during the interview that: "*Sometimes the distribution of tasks happened automatically*". This statement coincides with the ping-pong collaboration pattern, where the delegation of tasks happened implicitly, which was seen in two of their tasks. Group 1 expressed that they: "*... distributed the work by watching what the other person did.*" This group only had notable patterns where the ping-pong pattern occurred, and the statement substantiates the importance of implicit delegation for its occurrence.

One approach deemed important for the main patterns identified in the groups' collaboration is the *shared feedback* approach described in Section 2.3. Features related to presenting feedback on individual user activity were highlighting the objects each user currently had pressed and showing their "cursor" activity. Both features are detailed in Paragraph *Highlighting* in Section 3.5.1, and Paragraph *Cursor*

[30]https://en.wikipedia.org/wiki/Pair_programming

activity in Section 3.5.1. These features were important for enabling ping-pong collaboration because the pupils had to orient themselves according to what the other pupil was working on. For this to happen, the pupils had to be aware of the other pupil's location and activity. The features were likely also important for the singular action pattern. By seeing where and what the other pupil was working on, one could guide them. In addition to this, as it happened with group 3, one could also let the other pupil take over and finish the task if applicable. The benefits from using the shared feedback approach described in Section 2.3 were assumed to be why some of the main patterns emerged. For instance, being able to monitor each other's activities peripherally enabled ping-pong. Additionally, being able to explicitly assign and reassign themselves to sub-tasks enabled the parallel collaboration pattern.

The features related to presenting feedback were also crucial for parallel collaboration, especially when the pupils had sub-tasks that could not be solved entirely independently of each other. For instance, when group 2 solved Femten fordelt på seks, they had to be aware of what other objects the other pupil was working on to create individual piles simultaneously. In addition to this, the *intentional communication* mechanism described in Section 2.3.1 was deemed particularly important for the parallel pattern. Especially the first way to gather awareness information, by simply stating where they were and what they were working on. This was done by the pupils when they delegated tasks amongst themselves.

It could be argued that the *consequential* and *feedback* mechanisms were necessary for the main patterns seen in the test. These mechanisms are very similar to the shared feedback approach in that they both discuss how the shared workspace as an environment gives awareness information. As discussed above, this information was used differently in the three main patterns identified. From the first part of the framework described in Section 2.3.1, the who element was not directly relevant to the test as the pupils always knew who the other pupil was. It would be interesting to see how the main patterns would change or if they would emerge if three pupils were in each group.

Some of the activities described in the third part of the framework presented in Section 2.3.1 can be used to analyze the collaboration. In both group 2 and group 3, instances of pupils using deictic references to *simplify communication* can be found. For instance, in the conversation between the pupils in Paragraph *Femten fordelt på seks* in Section 6.3.

Several instances of ping-pong collaboration were enabled by *coordinating actions* and *anticipating* what the other pupil was doing. The former is evident since ping-pong mostly happened by delegating the sub-tasks implicitly, which was enabled by workspace awareness. Pupils would adjust their activity according to what the other pupil was doing. The latter is not as evident, but having the ability to foresee what a pupil could do is also a way of coordinating actions. Pupils would pick a sub-task by seeing what was available and anticipating what the other pupil was not about to do.

There were instances where pupils managed the transitions between individual and shared work, characterized by the framework as *management of coupling*. This can be seen in Figure 42, where the pupils went from ping-pong (shared) to a singular action pattern. Additionally, in the blue box of Figure 39, one pupil provided *assistance* by assessing what was needed.

7 Discussion

Answers to the research questions and the contributions of this thesis are discussed in this section. Section 7.1 aims to answer research questions presented in Section 1.3, and the contributions from this thesis are presented in Section 7.2. Considerations regarding the research and contributions are discussed in Section 7.3, while Section 7.4 evaluates the conducted project. Finally, Section 7.5 presents additional features that could be added to the platform if developed further.

7.1 Questions

The early stages of this project focused on developing the platform to be used in the final testing. Common features and attributes used in collaborative software platforms were identified and added to the platform. After a fully functional platform was developed, the later stages of the project were dedicated to designing the test described in Section 5.2, developing the monitoring system, and making final adjustments to the platform. This section aims to answer the research questions presented in Section 1.3.

RQ1: What are common features and technologies in collaborative software platforms?
There is often a focus on transparency within collaborative software platforms; that is, transparency in the sense that all users of the platform should be kept aware and updated about the state of the shared workspace. This is especially true for platforms that support simultaneous activity. For such platforms, it is essential to present all users with a cohesive representation of the state. Platforms often perform different forms of concurrency control to achieve this, such as locking, to ensure correct results from concurrent actions.

In addition to keeping the state consistent between users, platforms keep users aware of occurring actions within the interface. Providing users with this information allows them to dynamically switch between loose and tightly coupled collaboration effectively. An effective way of solving this is using the shared feedback approach. This approach allows for low overhead for both performers and observers, as the interface itself presents feedback on all users' activity.

Examples of shared feedback approaches are how Google Docs and Miro display users' positions within the shared workspace. In Google Docs, the position of a user relates to the location of their text cursor within the document. In Miro, the position is relative to the user's mouse pointer. When a user's position changes in both platforms, this change of position is reflected in other users' interfaces. Therefore, the moving user does not have to perform any additional action to update others of their relocation, as the action is reflected automatically within the interface. Providing instant feedback from actions is a common feature within groupware and presents users with greater flexibility in coordinating their collaborative work.

RQ2: Which features are necessary to facilitate collaboration in mathematics?
Throughout this project, several important features supporting collaboration in mathematics were identified and included in the platform. As presented in Section 6, the pupils were able to collaborate solving the different tasks. One could assume that the combination of features within the platform allowed for this collaboration to occur. However, the number of participants in the test was somewhat limited, and the assumption that the platform's features allowed for collaboration to occur can be easily challenged. Regardless, this thesis can be used as a document presenting features that are *potentially* necessary for facilitating collaboration in mathematics.

The design of these features is presented in Section 3, and how these features were implemented in the platform is shown throughout Section 4. Following is a selection of the features that were found to be most significant by the authors:

- Simultaneity when drawing lines
- Allow users to interact with objects in the shared workspace

- Making users aware of each other's actions

Having lines appear while they are drawn allows users to observe each other and be aware of what others are working on within the shared workspace. Combined with allowing users to interact with objects and having these actions replicated, could enable different types of collaborations to occur. The various features designed to make users aware of each other further increase the collaborative experience. This includes features such as lines appearing in real-time, highlighting selected objects, avatars displaying users' locations (also called "cursor" activity), and replicating actions with a low notification time.

RQ3: What patterns of collaboration occur when using the platform?
Tests with three pairs of pupils were conducted to identify patterns of collaboration that occur when using the platform. To identify the patterns, and because the authors could not physically be on-site during the tests, a monitoring system was created that logged and visualized interactions in the platform. From these tests, seven notable patterns were identified in the monitoring system. The notable patterns were then used to create abstractions that describe the main collaborative patterns. These were:

- **Parallel collaboration**: Parallel interactions
- **Ping-pong collaboration**: Alternating interactions
- **Singular action collaboration**: Only one pupil performing actions

Section 6.5 contains further discussion about these patterns.

Even though the main patterns were different, the semi-structured interview conducted with each group could indicate that neither of them skewed the workload within the groups. Each group felt they contributed equally and that the distribution of work was fair.

Features related to presenting feedback on users' activity, as emphasized by the *shared feedback* approach (read: Section 2.3), were assumed to be important for enabling the main patterns identified. For instance, take the features detailed in Paragraph *Highlighting* in Section 3.5.1 and Paragraph *Cursor activity* in Section 3.5.1. These features were important for enabling parallel collaboration when pupils had sub-tasks that could not be solved fully independently of each other. With the platform highlighting the objects the other pupil was working on, they could simultaneously solve their sub-task. In addition to this, with the platform indicating where the other user was by showing their "cursor" activity, the ping-pong collaboration pattern could occur. In this pattern, pupils had to be aware of where the other pupil was and what they were working on to orient themselves to pick a sub-task. These features were important in the singular action pattern where one pupil was guiding the other because it allowed the pupil to be aware of the other pupil's action. In addition to this, the other pupil did not have to communicate the actions taken, providing low overhead to the collaboration.

7.2 Contributions

This section presents this thesis's different contributions, which are products of the project conducted. These can be beneficial for similar projects in the future.

7.2.1 A framework for real-time collaborative software

This thesis can act as a document for researchers and developers implementing a real-time collaborative software platform. A framework describing important features and characteristics of such software is presented in Section 2. This framework is based on several literature works and can act as a guide when designing the features facilitating collaboration within groupware. Supporting this, Section 3 can be used as an example of a design that considers the different challenges and problems presented in this framework. Section 4 gives an example of a platform based on this design, and how the framework is used in a fully functional prototype. The framework primarily comprises two significant considerations within groupware:

1. Attributes and concurrency in groupware
2. Awareness within shared workspaces

A significant takeaway from this project is the importance of awareness within this type of software, as it is essential for allowing collaboration between actors.

7.2.2 Achieving simultaneity when drawing lines

During development, much effort was put into the real-time aspect of drawing lines and having them appear on other users' devices while they are drawn. Future projects tackling similar problems could use this thesis as inspiration to enable the streaming of partial objects as they are created. Section 4 presents how this is achieved in this project's platform, with the different subsections explaining the different parts that enable this functionality. Additionally, how the line-based objects are made accessible and interactive in the platform can be adopted and used in other projects. This system proved to be very modifiable, making adding new types of objects and manipulations a relatively straightforward affair.

7.2.3 Monitoring system

Using a monitoring system to identify collaborative patterns is an approach that was created during this project. Even though the initial reason for developing the system was to act as a work-around for not being on-site, it worked well for the purpose. The approach of logging events and visualizing them made it easier to identify collaborative patterns that occurred instead of only taking field notes. Different types of user actions that could be observed to categorize collaboration are provided in Section 4.3.2. Although these are specific to the project, they could be generalized and adapted to other contexts as needed.

7.2.4 Support for multiple collaboration patterns

As discussed in Section 6.5, different types of collaborations were identified. This thesis shows how the pupils' collaboration varied and that it was dynamic, adding to the findings of the third part of the workspace awareness framework presented in Section 2.3.1. Clearly, people collaborate in multiple ways, but it shows that a collaborative platform should be designed to enable variety within the collaboration. A dynamic and non-restrictive platform enables the users to choose how they want to collaborate when solving a shared task.

7.3 Considerations

As presented in Section 3, the features needed for this project were identified and adapted using the knowledge gained about groupware and awareness from Section 2. Therefore, the features chosen and implemented might have influenced the main patterns observed, and another combination of features could have given different results. In addition to this, bias from the authors could have informed the choice of features in the platform.

Due to only getting in contact with one teacher and time constraints related to the project, the test had few participants. Additionally, the results gathered from the test could have been denser with more participants, giving the authors more data to analyze. Due to the sparsity, the patterns noticed could have been influenced by other factors that were not present in the data. More patterns could have potentially emerged with more participants.

The drawing bug encountered during the test impacted how the tasks were completed, especially for group 2. This drawing bug impacted their collaboration, and there is a possibility that the singular action pattern would not have occurred for group 2 if this bug did not happen.

The analysis of the observations conducted potentially includes bias from the authors. There is also no guarantee that another person would have found the same patterns that the authors have. However, to increase the validity of the research conducted, verbatim quotes from the pupils and screenshots from the monitoring system were included. Another way to increase the validity of the research would have been to triangulate [3, p. 212] the observations with interviews after analyzing the data, where the pupils and teacher could have confirmed the main patterns observed by the authors. This triangulation differs from the interviews conducted after the test, where the goal was to elicit feedback regarding the collaboration.

7.4 Project evaluation

The initial goal for the project was somewhat open-ended, as the only criterion was that the project had to exist within the field of Educational Technology (EdTech). This resulted in the authors having an excellent opportunity to define the scope of the project. This opportunity led to increased motivation during all stages of the project and heightened the sense of ownership. The authors felt a strong desire to develop a piece of software when deciding what type of project to execute for this thesis.

In the early stages, considerable effort was put into researching what demands and opportunities existed within the field of EdTech in Norway. This effort was fueled by a desire to develop software that could be used in schools and education in the future. The first few weeks of the project mainly consisted of having conversations with different types of stakeholders; teachers, professors, developers, and companies related to education and EdTech. As a result of these conversations, the project was scoped towards designing a line-based collaborative mathematics platform and performing a test with the platform towards the end of the project. Conversations with most of these stakeholders were continued throughout the project, both to get feedback and ideas for the platform during development and to get in contact with interested teachers willing to test the finished platform.

The decision to implement the platform using Flutter was made based on a wish to try new technology. Flutter is a relatively new framework for mobile development. After a bit of research, Flutter seemed promising as the framework advertised good performance and rapid development. Developing the platform using Flutter was a good experience, and no significant features needed to be dropped due to framework limitations. Flutter includes several helpful components and classes, such as icons, fonts, buttons, and specialized containers. Building the various interfaces of the application felt very intuitive and quick. Compared to React Native, the authors felt that much time was saved using Flutter, especially when styling and aligning components. In the end, the platform was developed using a single codebase with minimal platform-specific code, which builds into native apps for both Android and iOS.

This project was executed during two semesters. The general plan for the project was to spend the first semester designing and developing the platform, and to use the second semester to plan, design, and perform the final test. Some features were added specifically to support the test during the second semester, while most core functionalities were implemented in the first semester. The general plan also included performing the final test before Easter and using the remaining time to analyze and discuss the findings. Getting in contact with teachers to help conduct the tests proved difficult due to the global pandemic. However, a teacher willing to test the platform using their pupils was found during the second semester. This teacher recruited three groups of willing pupils, which allowed the test to be conducted. Even though the authors could not be on-site, the combination of an audio link and a monitoring system allowed for sufficiently detailed observations. In the end, the project revealed some interesting findings and was completed following the general plan with no major delays.

7.5 Future Work

If the platform was to be developed further, additional features could be implemented. The platform was designed to support the final testing of the project, and only functionalities needed for this were implemented. This section presents features that could be added to the platform to make it suitable for general use-cases.

User accounts
Including user accounts in the platform was unnecessary for this project. However, if the platform was to be developed further, user accounts should be added. This way, users will have a persistent account, allowing them to establish connections with other users, such as friends and classmates. Log in and authentication could be implemented using third-party solutions such as Feide[31] or Google[32], to allow users to register effortlessly. A significant benefit of having a Feide integration in the platform is because Feide is widely used within the Norwegian education system, with over 1.3 million users registered [31].

Asynchronous interaction
Another point of improvement for the platform would be to support asynchronous interaction. Sessions should be kept persistent, allowing users to enter and exit sessions as they please. Storing session states within a database would allow users to re-enter a previous session and continue working from where they left off. As changes are made to the state, these should be sent to the database and other users. Upon re-joining a session, users would query the database and receive the session's current state. If the platform were to support this type of interaction, it would allow users to solve a task periodically, in the same way documents can be re-opened and edited in other collaborative software platforms such as Google Docs and Miro.

Teachers
The current platform only supports one type of actor, which is pupils. This choice was intentional as the platform was made specifically to facilitate the testing performed during this project. Additional types of actors, such as teachers, could be added to expand the platform's usefulness in an educational setting. Teachers could both have administrative and supervisory roles within the platform. They could assign tasks to individual pupils or groups and could receive the pupils' submissions in the platform. Another benefit of having a teacher role within the platform is allowing teachers to supervise pupils remotely. Pupils could then ask for help with a task within the interface, allowing teachers to enter sessions as supervisors.

Tasks
In the current platform, the tasks are stored locally as a JSON file. If tasks were to be stored in a database, it would allow for greater flexibility in adding, removing, or editing tasks. It would also enable teachers to design tasks and submit them to this database, even though this would require additional interfaces to be added to the platform. Teachers could also assign a set of tasks to pupils, which would provide teachers with greater control of the content provided to their pupils. Having a database with tasks could also allow the platform to be used for more subjects than mathematics, as the shared workspace may be flexible enough to be used in other subjects with minimal effort.

[31] https://www.feide.no/
[32] https://www.google.com/account/about/

8 Conclusion

This project was completed over two semesters, with the first semester dedicated to designing and developing the platform and the second semester focused on testing. Developing the platform went according to plan, allowing the monitoring system to be designed and implemented when it was needed. This monitoring system proved valuable for identifying collaborative patterns from the test. Due to only getting in touch with one teacher, the test had fewer participants than ideal. The sparsity in the gathered data might have omitted information that could have strengthened the analysis and findings.

Related to RQ1, it was found that common features and technologies in collaborative software often focused on a common attribute: transparency regarding the state of the shared workspace. Concerning RQ2, allowing users to be aware of each other and their actions is potentially the single-most-important feature. Enabling simultaneity in the workspace allowed users to interact effectively with the shared objects. As for RQ3, the identified collaborative patterns from the test were classified as parallel, ping-pong, and singular action collaboration. Parallel collaboration is where the pupils interacted with the shared workspace in parallel. In ping-pong collaboration, the pupils' interactions were alternating. Singular action collaboration is characterized as instances where only one pupil interacted with the shared workspace.

This thesis can be used as a document presenting a framework for creating real-time collaborative software, which highlights useful features to include. The design and implementation presented in this thesis can be used to illustrate how these features could be implemented when developing a collaborative software platform that supports dynamic collaboration. The thesis also provides an example of implementing simultaneity in actions related to mathematics, such as drawing and manipulating objects. Additionally, it presents a method of how a monitoring system can be used in conjunction with field notes and interviews to identify collaborative patterns.

The research presented in this thesis mainly focused on identifying collaborative patterns occurring as pairs of pupils worked together on solving mathematical tasks. Within the topic of collaboration in mathematics on tablets, there are many other exciting subjects to explore. For instance, would the characteristics of the patterns identified in this thesis change if more than two pupils were to collaborate? Would any of these patterns occur at all, and would new patterns be observed? Are the identified patterns presented in this thesis generalizable? It would also be interesting to evaluate how the different functionalities of the platform, such as highlighting and cursor activity, affect collaboration and if one could determine which of these features are most beneficial.

Bibliography

[1] Kjersti Nipen. Nettbrettene rykker inn i klasserommet. ingen vet helt hva det gjør med læringen., 2021. https://www.aftenposten.no/amagasinet/i/0paaq0/nettbrettene-rykker-inn-i-klasserommet-ingen-vet-helt-hva-det-gjoer-me.

[2] Helga Engs Sæl. Mål med implementeringen av en-til-en - FIKS - forskning, innovasjon og kompetanseutvikling i skolen. https://www.uv.uio.no/forskning/satsinger/fiks/kunnskapsbase/digitalisering-i-skolen/-mal-med-implementeringen-av-en-til-en/index.html.

[3] Briony J. Oates. *Researching information systems and computing*. SAGE Publications, 2006.

[4] Paul Wilson. *Computer Supported Cooperative Work:: An Introduction*. Springer Science & Business Media, 1991.

[5] Clarence A. Ellis, Simon J. Gibbs, and Gail Rein. Groupware: some issues and experiences. *Communications of the ACM - Vol 34, No. 1*, 34(1):39–58, 1991.

[6] Robert Johansen, Jeff Charles, Robert Mittman, and Paul Saffo. *Groupware: Computer Support for Business Teams*. Free Pr, first edition, 1st printing edition, 1988.

[7] Peter H. Carstensen and Kjeld Schmidt. Computer supported cooperative work: New challenges to systems design. In *In K. Itoh (Ed.), Handbook of Human Factors*, pages 619–636, 1999.

[8] Carl Gutwin and Saul Greenberg. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)*, 11(3):411–446, 2002.

[9] C A Ellis and S J Gibbs. Concurrency control in groupware systems. *Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, page 9, 1989.

[10] Saul Greenberg and David Marwood. Real time groupware as a distributed system: Concurrency control and its effect on the interface. *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, page 11, 1994.

[11] A. Karsenty and M. Beaudouin-Lafon. An algorithm for distributed groupware applications. In *[1993] Proceedings. The 13th International Conference on Distributed Computing Systems*, pages 195–202, 1993.

[12] Matthias Ressel, Doris Nitsche-Ruhland, and Rul Gunzenhäuser. An integrating, transformation-oriented approach to concurrency control and undo in group editors. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work - CSCW '96*, pages 288–297. ACM Press, 1996.

[13] Brice Nédelec, Pascal Molli, Achour Mostefaoui, and Emmanuel Desmontils. Concurrency effects over variable-size identifiers in distributed collaborative editing. *CEUR Workshop Proceedings*, page 8, 2013.

[14] Chengzheng Sun. Operational transformation in real-time group editors: Issues, algorithms, and achievements. *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, page 10, 1998.

[15] Mehdi Ahmed-Nacer, Pascal Urso, Valter Balegas, and Nuno Preguiça. Concurrency control and awareness support for multi-synchronous collaborative editing. In *9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 148–157, 2013.

[16] Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work - CSCW '92*, pages 107–114. ACM Press, 1992.

[17] Thomas Olsson, Pradthana Jarusriboonchai, Paweł Woźniak, Susanna Paasovaara, Kaisa Väänänen, and Andrés Lucero. Technologies for enhancing collocated social interaction: Review of design solutions and approaches. *Computer Supported Cooperative Work (CSCW)*, 29(1):29–83, 2020.

[18] Tom Gross. Supporting effortless coordination: 25 years of awareness research. *Computer Supported Cooperative Work (CSCW)*, 22(4):425–474, 2013.

[19] Carl Gutwin, Saul Greenberg, and Mark Roseman. Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation. *People and Computers XI*, page 24, 1996.

[20] Google. Google docs, 2021. `https://www.google.no/intl/en/docs/about/`.

[21] Regina Maria Ambrose and Shanthini Palpanathan. Investigating the effectiveness of computer-assisted language learning (CALL) using google documents in enhancing writing–a study on senior 1 students in a chinese independent high school. *IAFOR Journal of Language Learning*, 3(2):85–112, 2017. Publisher: International Academic Forum.

[22] Miro. Miro, 2021. `https://miro.com/online-whiteboard/`.

[23] Gerry Stahl, Timothy Koschmann, and Dan Suthers. Computer-supported collaborative learning: An historical perspective. *Cambridge handbook of the learning sciences*, page 1, 2006.

[24] D. Hernandez-Leo, E.D. Villasclaras-Fernandez, J.I. Asensio-Perez, Y.A. Dimitriadis, and S. Retalis. CSCL scripting patterns: Hierarchical relationships and applicability. In *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*, pages 388–392, 2006. ISSN: 2161-377X.

[25] Pierre Dillenbourg, Sanna Järvelä, and Frank Fischer. The evolution of research on computer-supported collaborative learning. In Nicolas Balacheff, Sten Ludvigsen, Ton de Jong, Ard Lazonder, and Sally Barnes, editors, *Technology-Enhanced Learning: Principles and Products*, pages 3–19. Springer Netherlands, 2009.

[26] Mia Carapina and Ivica Boticki. *Technology Trends in Mobile Computer Supported Collaborative Learning in Elementary Education from 2009 to 2014*. International Association for the Development of the Information Society, 2015. Publication Title: International Association for Development of the Information Society.

[27] Lars Bollen, Hannie Gijlers, and Wouter van Joolingen. Computer-supported collaborative drawing in primary school education – technical realization and empirical findings. In Valeria Herskovic, H. Ulrich Hoppe, Marc Jansen, and Jürgen Ziegler, editors, *Collaboration and Technology*, Lecture Notes in Computer Science, pages 1–16. Springer, 2012.

[28] C. Ferraris and C. Martel. Regulation in groupware: the example of a collaborative drawing tool for young children. In *Proceedings Sixth International Workshop on Groupware. CRIWG 2000*, pages 119–127, 2000.

[29] Yinghui Li, Zhichao Cao, and Jiliang Wang. Gazture: Design and implementation of a gaze based gesture control system on tablets. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):74:1–74:17, 2017.

[30] Rikke Friis Dam and Teo Yu Siang. What is design thinking and why is it so popular? `https://www.interaction-design.org/literature/article/what-is-design-thinking-and-why-is-it-so-popular`.

[31] Hvorfor tilby feide-innlogging? `https://www.feide.no/hvorfor-feide-innlogging-p%C3%A5-tjeneste`.

Appendix

A Special terms

Acronyms

CAW Computer Assisted Writing. 13

CRDT Conflict Free Data Types. 9, 10, 25

CSCD Computer-Supported Collaborative Drawing. 16

CSCL Computer-Supported Collaborative Learning. 3, 5, 15, 16

CSCR Computer-Supported Cooperative Reading. 16

CSCW Computer-Supported Cooperative Work. 3, 5, 10, 15

CSCWR Computer-Supported Cooperative Writing. 16

EdTech Educational Technology. 77

MCSCL Mobile Computer-Supported Collaborative Learning. 16

OT Operational Transform. 9, 10

WYSIWIS What You See Is What I See. 8, 17

B Code examples

B.1 Receive object action

```
void receiveObjectAction(CanvasObjectAction action, List<Line> lines) {
    switch (action.action) {
      case ObjectAction.create:
        handleCreateAction(action, lines);
        break;
      case ObjectAction.delete:
        handleDeleteAction(action, lines);
        break;
      case ObjectAction.move:
        handleMoveAction(action, lines);
        break;
      case ObjectAction.moveImage:
        handleMoveImageAction(action);
        break;
      case ObjectAction.rotate:
        handleRotateAction(action);
        break;
      case ObjectAction.pressed:
        handlePressedAction(action, lines);
        break;
      case ObjectAction.unpressed:
        handleUnPressedAction(action);
        break;
      case ObjectAction.link:
        handleLinkAction(action, lines);
        break;
      case ObjectAction.unlink:
        handleUnLinkAction(action, lines);
        break;
      case ObjectAction.pin:
        handlePinAction(action);
        break;
      case ObjectAction.addEmoji:
        handleAddEmojiAction(action);
        break;
      default:
        print("unsupported action");
    }
    if (action.action != ObjectAction.move) {
      removeUntrackedLines(lines);
    }
}
```

Listing 10: receiveObjectHandler in the CanvasObjectHandler class

C Application screenshots

C.1 Onboarding

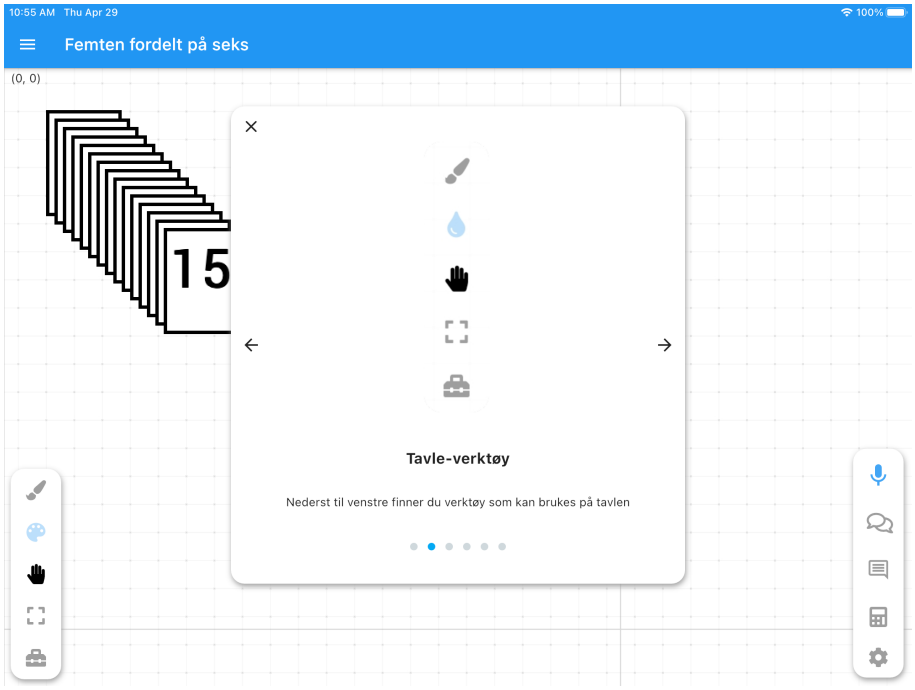


Figure 44: The onboarding consists of a slideshow with screenshots and descriptions

C.2 Information box

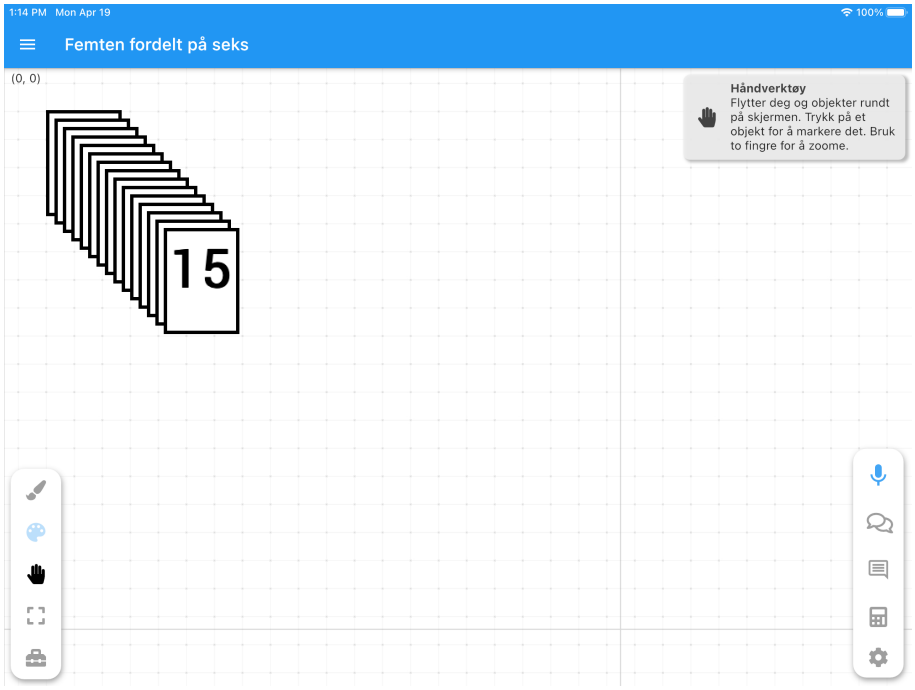


Figure 45: Information box is shown in the top right corner

C.3 Invitation

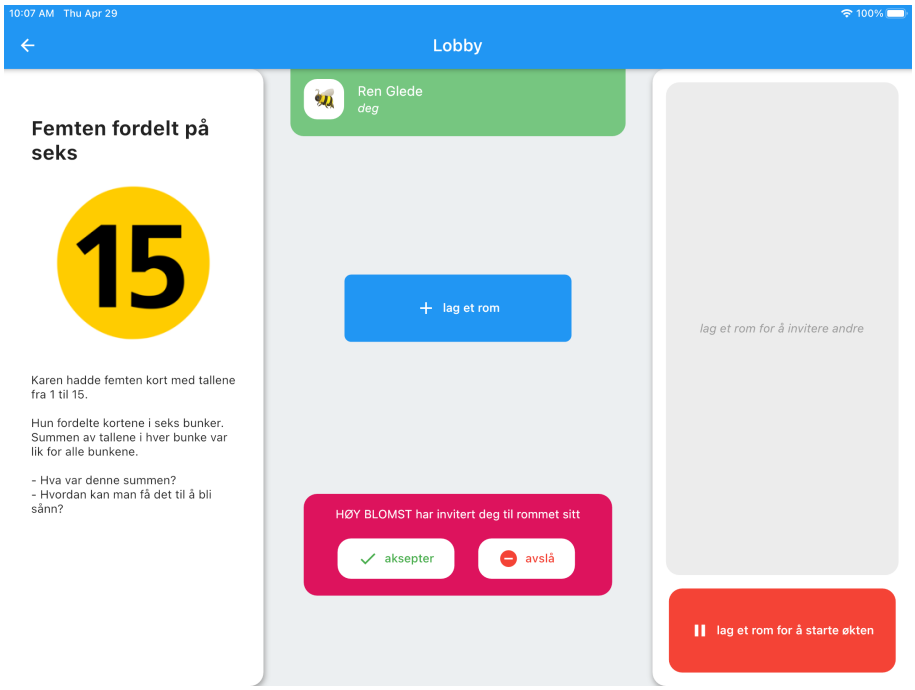


Figure 46: Invitations can either be accepted or declined

C.4 Chat

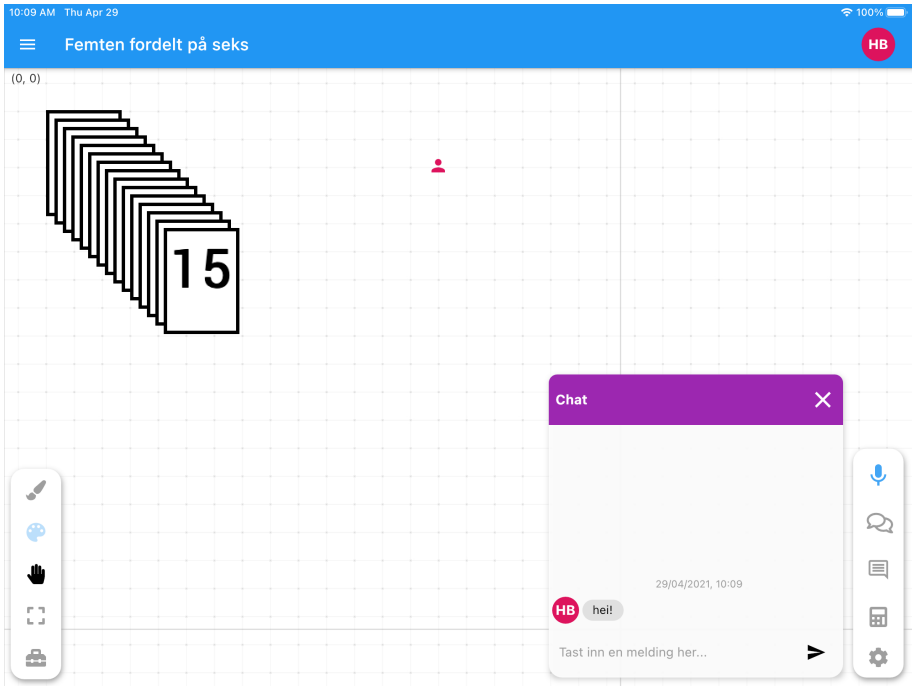


Figure 47: The chat allows users to write messages to each other

C.5 Comments

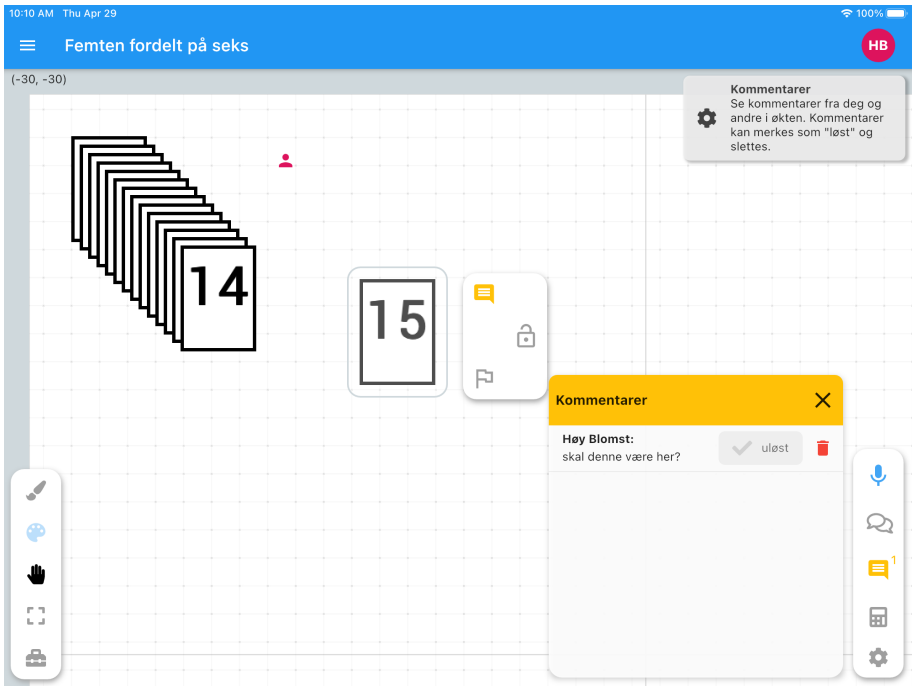


Figure 48: Comments are show in this box, pressing the comment highlights the related object(s)

C.6 Calculator

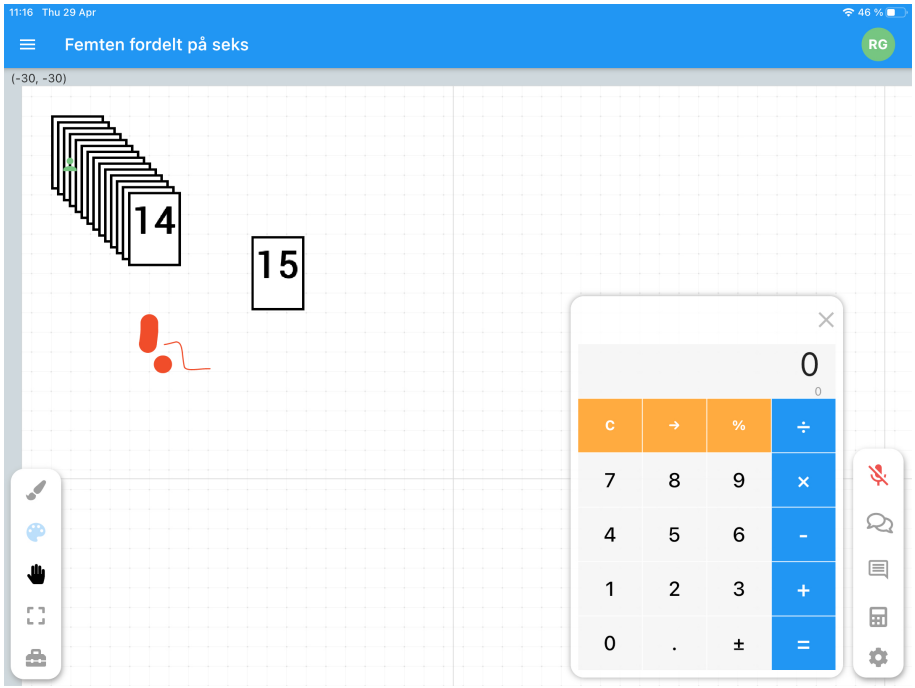


Figure 49: The calculator allows users to do simple calculations

C.7 Settings

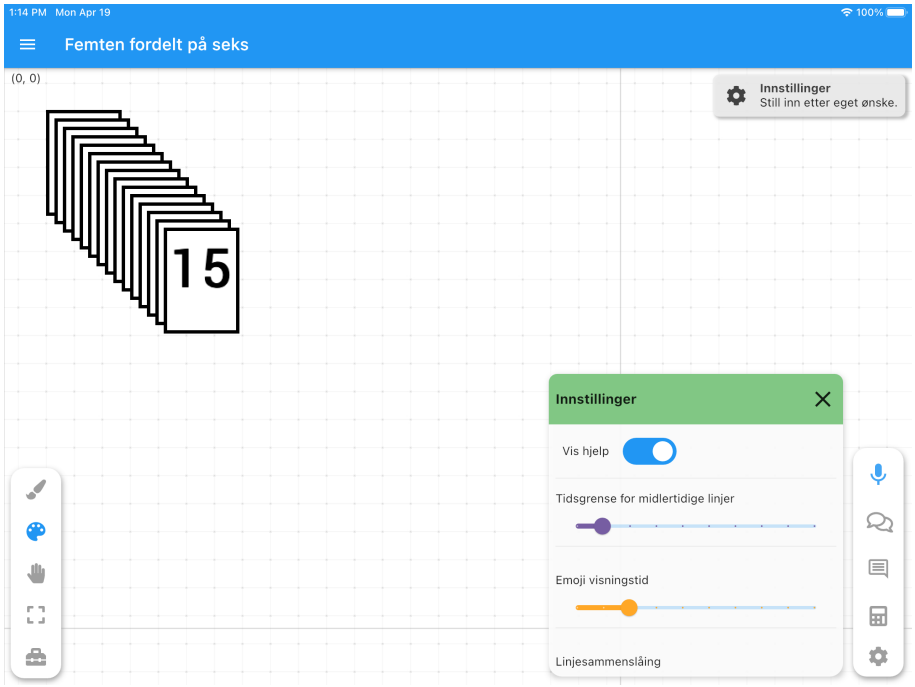


Figure 50: The settings menu lets users change certain parameters

C.8 Side menu

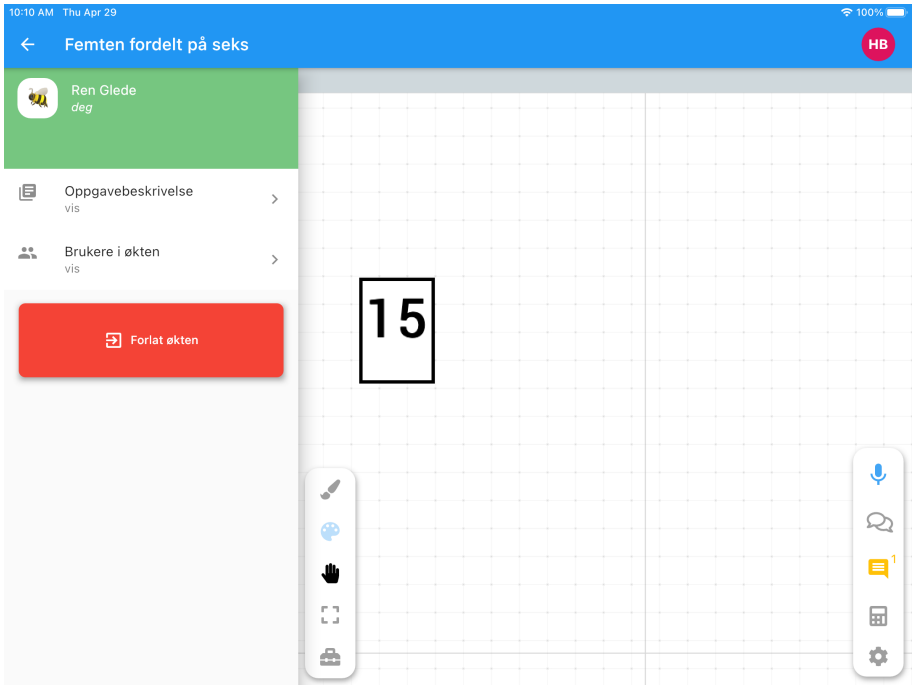


Figure 51: The side menu provides users with important information, and allows them to leave the session

C.9 Task description

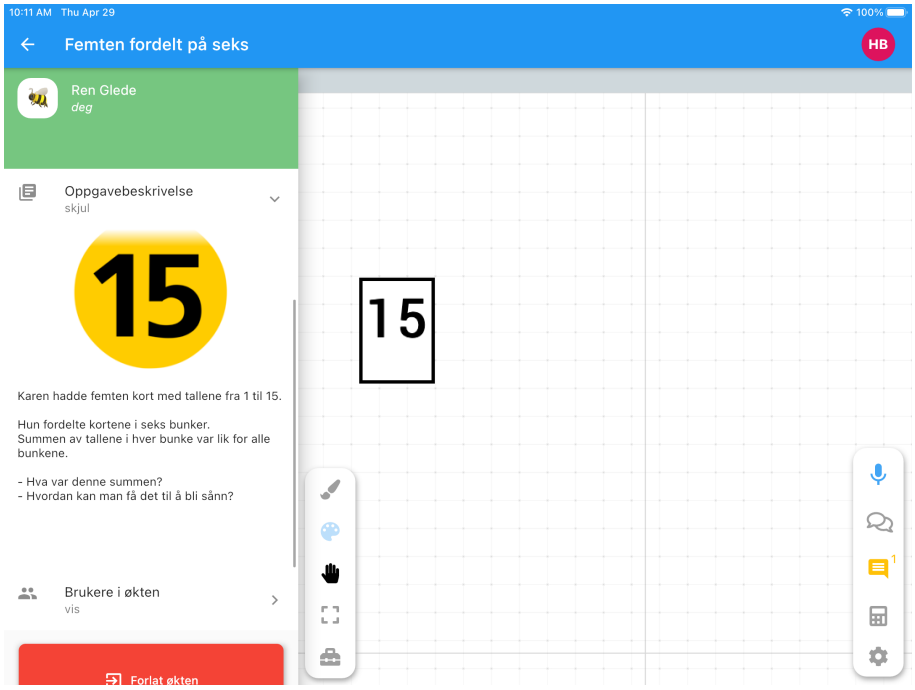


Figure 52: Users are able to re-read the task description from within the session

C.10 Users in the session

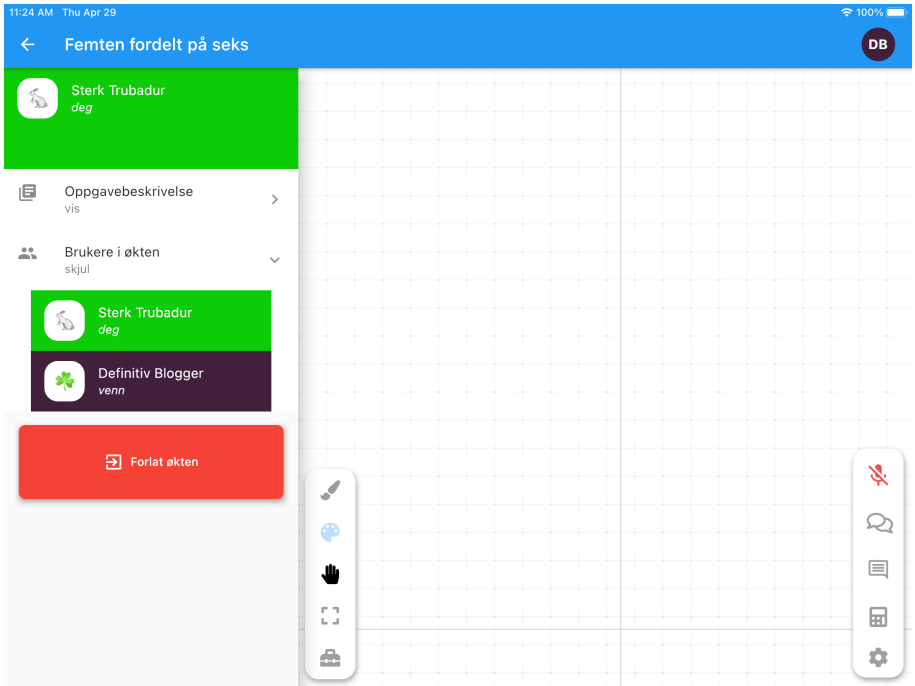


Figure 53: All users in the session is listed will full names, pressing the user-tiles centers the whiteboard around their location

C.11 Color picker

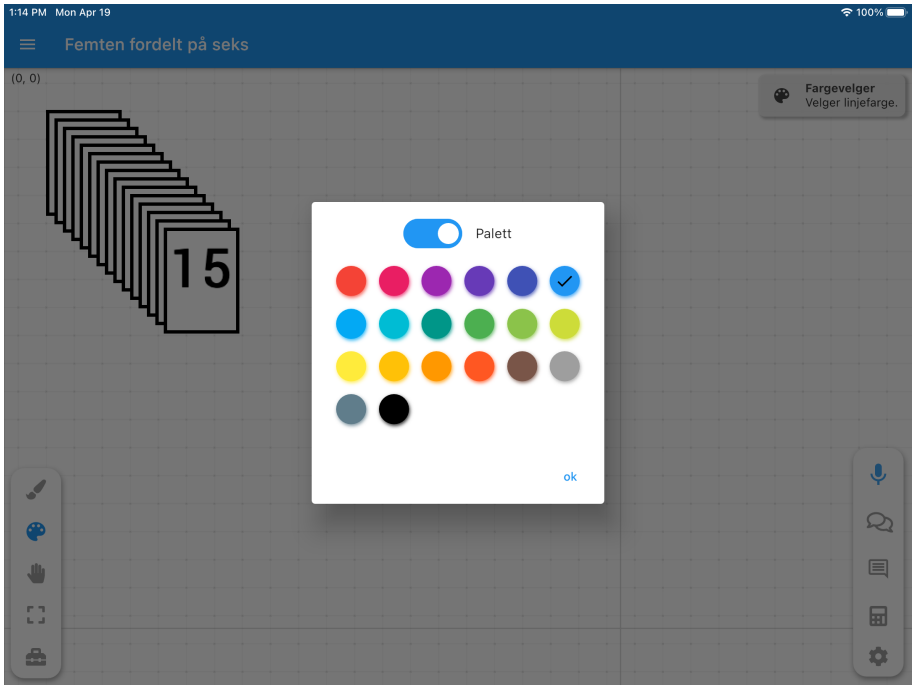


Figure 54: The color picker provides users with a color palette to choose from

C.12 Zoom

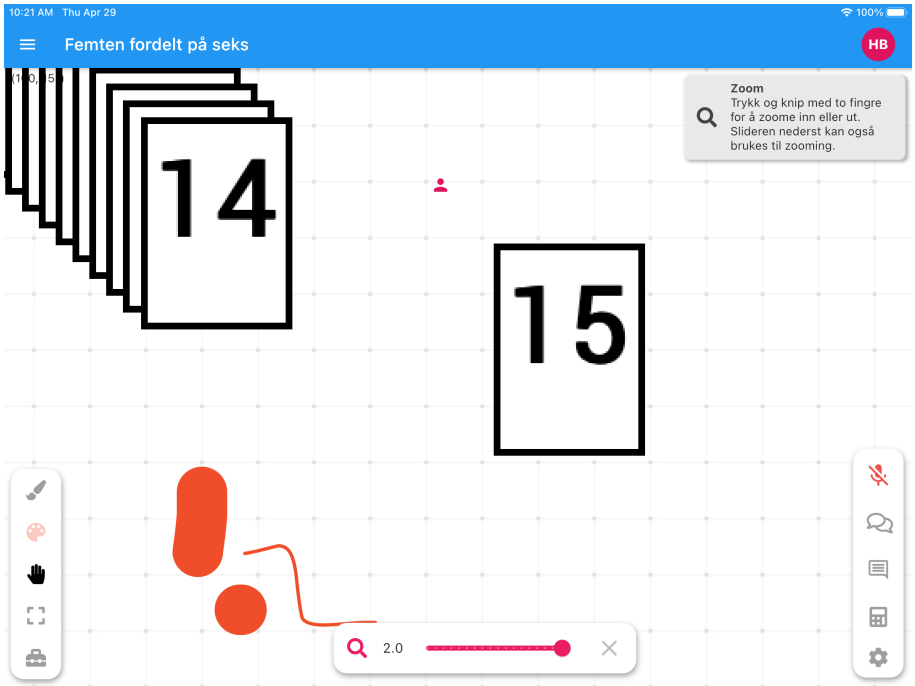


Figure 55: Users are able to zoom in and out. The possible range is from 0.3x to 2x zoom. Zooming is either done by pinching with two fingers, or using the slider in the center bottom part of the screen

C.13 Compass

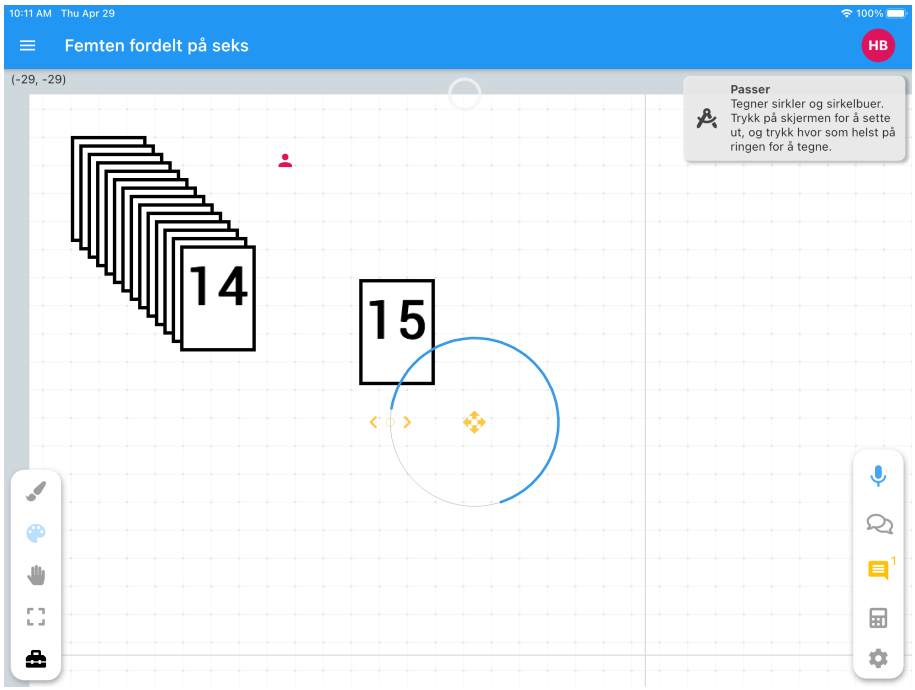


Figure 56: The radius of the compass can be changed by dragging the icon on the left side of the compass' circle. The compass is moved by pressing the central node. Drawing is done by pressing and dragging on the compass arc

D Usability test questions

Oppgaver

Task ID	Oppgavebeskrivelse	Utført
1	Velg økten "20 fordelt på 6"	☐
2	Lag et rom og inviter en annen bruker til økten	☐
3	Når alt ser klart ut, trykk på "start økten"	☐
4	Kommuniser med partneren din gjennom appen, og bli enige om en mulig taktikk	☐
fritt arbeid	**Jobb med oppgaven sammen med partner i et par minutter**	☐
5	Flytt på kortene	☐
6	Prøv å gruppere flere kort	☐
7	Prøv å låse posisjonen til et kort	☐
8	Prøv å flytte flere u-grupperte kort samtidig	☐
9	Send en stemme-melding	☐
10	Send en chat-melding	☐
11	Test kommenteringsfunksjonen på et bilde	☐
12	Prøv de ulike penn-verktøyene	☐
13	Ta en titt i verktøykassen, og forklar hva de er og hv de kan brukes til	☐
14	Finn informasjon om økten i side-menyen	☐

+ New

COUNT 15

Figure 57: Tasks used for usability testing

E Test schedule and instructions

Test av samarbeidsplattform 23.03.2021

Plan for testen:

Gruppe 1		Gruppe 2		Gruppe 3	
08:45	Introduksjon	09:45	Introduksjon	10:45	Introduksjon
08:50	Byggeklosser	09:50	Grublis	10:50	Telle marihøner
08:55		09:55		10:55	
09:00	Femten fordelt på seks	10:00	Byggeklosser	11:00	Grublis
09:05		10:05		11:05	
09:10	Grublis	10:10	Telle marihøner	11:10	Femten fordelt på seks
09:15		10:15		11:15	
09:20	Telle marihøner	10:20	Femten fordelt på seks	11:20	Byggeklosser
09:25		10:25		11:25	
09:30	Intervju med deltakere / diskusjon med lærer	10:30	Intervju med deltakere / diskusjon med lærer	11:30	Intervju med deltakere / diskusjon med lærer
09:35		10:35		11:35	
09:40	pause	10:40	pause	11:40	pause

Testen gjennomføres i 3 grupper bestående av to deltakere.
I appen er finnes det fire hoved-økter, og en introduksjon. Introduksjons-økten skal gjøres de andre oppgavene slik at deltakerne kan bli litt kjent med appen.

Deltakerne skal sitte i samme rom, med ryggen mot hverandre.

Instruksjoner underveis

========== Introduksjon ==========

Introduksjons-økt (ca. 5 min)

- be deltakerne om å velge introduksjons økten
- be en deltaker om å lage et rom og inviter den andre, og sette seg som klare (når de er klare)
- be deltakerne om å følge instruksjonene som står på kortene, disse er ment for å hjelpe dem med å forstå interaksjon med objekter på flaten.
- Etter ca 5 minutter med utforsking i flaten kan du avbryte dem, og be dem om å avslutte økten.
- Gå videre med test-opplegget:)

========== Hoveddel ==========

Velge oppgave

- Instruer deltakerne om å velge riktig oppgave basert på gruppenummer (vist i tabellen på første side)

I “lobby”

- Be **en** deltaker om å starte et rom
- Be samme deltaker om å invitere den andre brukeren til rommet
- Be deltakerne om å si brukernavnet sitt høyt
- Be deltakerne om å skru av mikrofonene sine i appen for å unngå feedback.
- Be deltakerne om å markere seg som “klare” når de er klare.
- Si ifra til oss når de har laget et nytt rom og er klare for å starte økten
- Be deltakerne om å starte øken

I samarbeidsflaten

- Be deltakerne se gjennom info-popup’en (kun første oppgave, eller etter behov)
- Hjelp deltakerne om de står helt fast med oppgaven eller brukergrensesnittet
- Når deltakerne er ferdige med oppgaven kan de avslutte økten via sidemenyen øverst til venstre i appen.
- Om deltakerne har slitt for lenge med en oppgave kan det være lurt å hoppe videre slik at de får testet senere oppgaver. Si ifra til oss om dette skjer :)

^ gjenta helt til gruppen har gjort alle oppgavene.

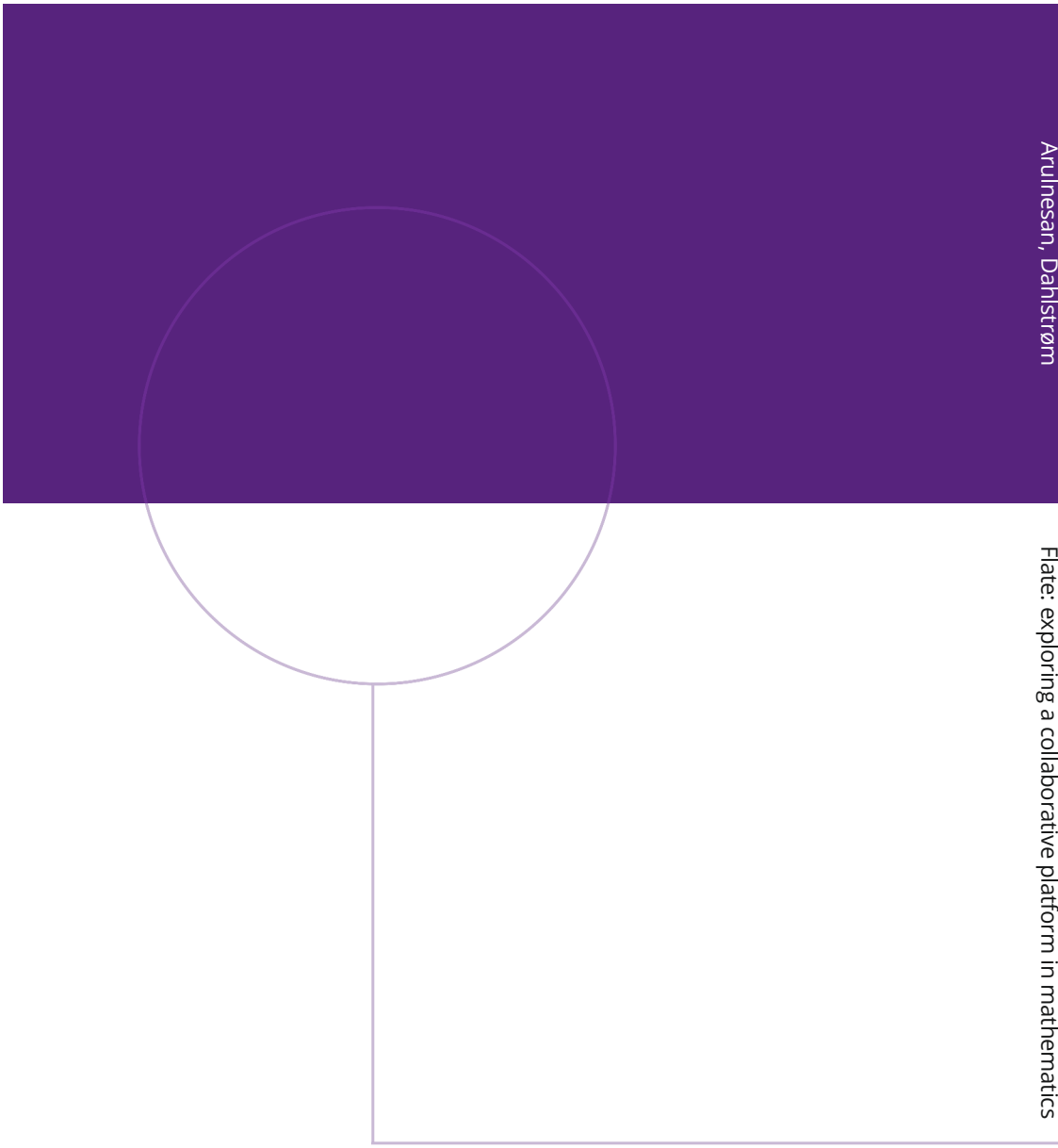
==========

Kjente feil og håndtering av disse

- En kjent feil er at en strek kan bli fastlåst på brettet uten at den kan slettes, be isåfall deltakerne om å ignorere dette.
- Om noe oppfører seg rart i en oppgave, be deltakerne om å avslutte økten og starte på nytt. Forhåpentligvis mister de ikke så mye fremgang.

Etter ferdig test:

Vi intervjuer elevene om hvordan de opplevde arbeidet med appen (ca 5-10 min).



Arulnesan, Dahlstrøm

Flate: exploring a collaborative platform in mathematics

NTNU
Norwegian University of
Science and Technology