

Mohammad Al Nayef
Alexander Carlsen
Gaute Wierød Rønning

A visual approach to improving the communication of task interdependencies in complex software development projects

Bachelor's project in Computer Engineering

Supervisor: Donn Morrison

May 2021

Norwegian University of Science and Technology

Faculty of Information Technology and Electrical Engineering

Department of computer science



Norwegian University of
Science and Technology

Preface

The choice of the thesis was based on the opportunity to learn and use some of the newest technology in the industry while adapting and refining our own learned knowledge from three years of studying at IDI. The assignment was interesting and somewhat open, letting us choose the technologies, methods and research questions to tackle.

Our process was working together, either in person if we could or using communication tools and version control tools. Throughout the project, the team members helped each other to be the best engineer possible, filled different roles to cover any weaknesses and tackled a variety of different hurdles and challenges. By following closely up on our system development methodology and having a good communication channel with both our supervisor and the product owner we managed to manufacture a project management tool.

This thesis was carried out in connection with NTNU at the faculty IE under the institution of IDI. The time frame of this thesis was from the beginning of the spring semester to the end of May 2021.

We would like to thank our product owner Favn Software AS for their supportive role during development and to thank NTNU and the provided supervisor in the form of Donn Morrison. The team greatly owes thanks to Equinor AS and Tryg Forsikring for participating in the testing phase of the development and for expressing excellent feedback. Lastly, we would like to thank IDI and the other teachers at IDI for providing a good environment for learning and guidance through the broad field of software engineering.

Team signatures:



Alexander Carlsen



Mohammad Al Nayef



Gaute Wierød Rønning

Date: 20.05.2021, place: Trondheim

Thesis description

The purpose of the project is to investigate and develop a possible solution for improving the communication of task interdependencies in software projects by the software consultancy firm Favn Software AS. The new solution needs to be secure, as well as intuitive and easy to use in order to save resources and time spent on coordination and communication. There are two main goals for the collaborative tool that is to be made; increasing productivity through improving communication and reducing the amount of superfluous and irrelevant information users have to work with.

In the Vision document, we describe the project stakeholders, the product and issue summaries, the user environment, and both the functional and non-functional properties of the product. The focus of this bachelor thesis moved away from microservices and towards task interdependencies during the project, which is also described in the project's vision.

Abstract

Task interdependencies is a core success factor of software development projects and proves to be one of the key challenges in terms of communication. Aligning developers' perceived technical constraints with the actual interdependencies of complex projects can lead to increased productivity. In the literature, collaborative tools have been proven to improve communication in development teams across two dimensions: time spent obtaining information and the relevance and understandability of the information itself. This paper builds upon this research and investigates how the use of collaborative tools centred around the visualization of task interdependencies affects communication in complex projects and compares it with the effect of general collaboration tools.

In order to research this, a specialized collaborative tool was developed as a full-stack web application. This technical solution was then utilized in the experiment together with an existing commercial general solution.

Our empirical evaluation shows that the use of specialized collaboration tools in the planning phase of a project improved communication across both dimensions. Time spent communicating was halved and the reported degree of understandability was significantly higher for the experiment participants that used the technical solution developed in this project, compared to the ones who used the general collaborative tool.

Table of content

| | |
|---|-----------|
| Preface | 3 |
| Thesis description | 4 |
| Abstract | 5 |
| Table of content | 6 |
| 1. Introduction and relevance | 7 |
| 1.1 Acronyms and abbreviations | 8 |
| 2 Theory | 8 |
| 2.1 Project management theory | 9 |
| 2.1.1 Task interdependencies | 9 |
| 2.1.2 Modularization | 9 |
| 2.1.3 Socio-technical congruence | 11 |
| 2.1.4 Coordination strategies | 12 |
| 2.1.5 Agile Development | 12 |
| 2.2 Technical theories | 14 |
| 2.2.1 NoSQL Databases and Relational databases | 14 |
| 2.2.2 OAuth2 | 16 |
| 2.2.3 Breadth-first search algorithm in tree data structure | 18 |
| 2.2.4 Microservices | 19 |
| 3. Choice of technologies and methods | 20 |
| 3.1 Methodology of the technical solution | 20 |
| 3.1.1 Choice of technologies | 20 |
| 3.1.1 a Database System | 20 |
| 3.1.1 b Authorization | 21 |
| 3.1.1 c Development Methodology | 21 |
| 3.1.2 Key user interaction design decisions | 22 |
| 3.2 Research Method | 24 |
| 4. Results | 26 |
| 4.1 Scientific results | 26 |
| 4.2 Product and system design | 30 |
| 4.3 Engineering results | 32 |
| 4.4 Administrative results | 33 |
| 4.4.1 Scrum Artefacts | 33 |
| 4.4.2 Project Progress | 34 |
| 4.4.3 Roles and division of labour | 35 |
| 5. Discussion | 36 |
| 5.1 Scientific discussion | 36 |
| 5.1.1 Limitations | 39 |
| 5.2 Engineering discussion | 40 |
| 5.2.1 Limitations | 42 |
| 5.3 Administrative discussion | 42 |

| | |
|---------------------------------------|-----------|
| 5.4 Ethical discussion | 44 |
| 5.5 Member reflections | 45 |
| 6. Conclusion and further work | 45 |
| 6.1 Future work | 46 |
| 7. References | 47 |
| 8. Attachments | 50 |

1. Introduction and relevance

The background of this project is Favn Software's demand for improving communication within development projects. Favn is a software consultancy firm established in 2020 and that mainly works remotely. This requires much more frequent use of collaboration tools, which is where they encountered a major challenge. Project leaders in Favn found it challenging to convey technical constraints through the tools. Additionally, due to the frequent use of such tools, they also encountered problems with superfluous information. A lot of the information was irrelevant for the user at the given point in time, as the tasks were not ready to be worked on due to technical constraints.

Communication affects a number of critical success factors in software development projects, such as the exchange of necessary information needed for effective collaboration and autonomous decision making. Improving communication leads to higher productivity, which contributes to increasing the profitability of Favn's consultancy business. Additionally, expanding this knowledge field can have long term effects such as lowering the average cost of software development in society by enabling a slightly more efficient utilization and distribution of resources.

The goal of this project is to investigate which features in collaboration tools contribute to improved communication of task interdependencies and develop a technical solution that mitigates the aforementioned problems, in order to increase overall productivity in software development projects. Cataldo and colleagues (2008) and Anders & Zmud (2015) researched task interdependencies and modularization, however, this research did not revolve around the use of collaborative tools with the focus on task interdependencies, as this is where the research of this project diverges from theirs. With this goal in mind, the team is going to answer these questions by developing a technical solution and thereafter performing an experiment with the solution and a control group consisting of a combination of computer engineering students and developers in Favn. Based on the aforementioned needs and challenges encountered by Favn, the following research question for this project is formed:

“How can collaborative tools centred around visualization of task interdependence contribute to improved communication in complex software development projects?”

To properly address the research question this report firstly defines relevant theory, secondly describe the technologies used to find a fitting solution for the demands of Favn, thirdly

defining the produced results, fourth discussing the results and its limitations, and lastly concluding further work and the findings of the experiment.

1.1 Acronyms and abbreviations

- RDBMS - Relational Database Management System
- OAuth - Open-standard Authorization protocol or framework (Richer & Sanso, 2017, p. 236)
- Specialized collaboration tools - collaborative tools that are centred around or contain features for specifically communicating task interdependence (Cataldo et al, 2008).
- Unspecialized collaboration tools - general collaboration tools that do not contain features for specifically communicating task interdependence Catalfo et al, 2008).
- API - Application Programming Interface.
- SQL - Structured Query Language (Mason, 2015)
- NoSQL - Query Languages that are not Structured like SQL. (Mason, 2015)
- CAP - Consistent, Availability, Partition tolerance, an early theorem to define data consistency in NoSQL databases. (Mason, 2015)
- BASE - Basically Available, Soft state and Eventually consistency, an evolved version of the CAP theorem. (Mason, 2015)
- JSON - JavaScript Object Notation, a lightweight data format. (Mason, 2015)

2. Theory

The following chapter describes theories and concepts relevant to the nature of the communication of interdependent tasks in software development. It delves firstly into software project management in general and then further into the use of modularization and dependencies as a framework for development management. It also provides an overview of the theories behind the technologies used in our technical solution.

2.1 Project management theory

2.1.1 Task interdependencies

Task interdependencies is the measure of the effect of modifications on one module on the scope of modifications needed on other modules to accommodate the change, as well as the extent to which a task requires other project- or organizational resources to communicate and share information (Andres & Zmud, 2015). According to Andres and Zmud (2015), task interdependence is one of the three core variables that affect the success of software development projects, the others being coordination strategies, which this paper will delve into later, and goal conflicts, which are not relevant for our research.

In projects with a low degree of task interdependence, each project member's contribution is additive. Complex projects with higher degrees of task interdependence require team members to integrate their work with others, and thus heightens the need for coordination. Straus and McGarth (1994) observed that these projects are characterized by frequent information exchanges to clarify task assignments, project progress, and goals. This also leads to diminished productivity as a result of more time spent reconciling differences in goals and perceived task requirements. Such software projects typically have reciprocal workflows, where individual team members' progress can be halted when they are dependent on output from other team members to complete their own tasks (Andres & Zmud, 2015).

In order to mitigate the aforementioned problems with software projects characterized by a high degree of task interdependencies, two solutions are proposed: improve communication and reduce task interdependence (Van de Ven et al, 1976; Andres & Zmud, 2015).

Communication with regard to software projects can be improved through the use of coordination strategies and collaborative tools across two dimensions: the relevance of information and time spent obtaining and sharing the relevant information. Task

interdependence can be reduced by the implementation of modularization together with the use of the analytical framework “socio-technical congruence” (Cataldo et al, 2008).

2.1.2 Modularization

Modularization is the decoupling of interconnected compartments in a complex system into modules that have more internal than external connections (Kharrazi, 2019, p.414-418). The utilization of modularization in task structures demands an overall modular system design. In regards to development projects, this means reducing complex systems into smaller components that can only be assigned to one team. This makes complexity in large projects more manageable as well as compartmentalizes risk (Baldwin & Clark, 2000). In addition, a modular task structure enables development teams to work independently and in parallel, which reduces the need for communication between work teams (Parna, 1972).

A common problem with the practical utilization of modularization is that intergroup communication and information sharing sometimes is reduced too much. This leads to problems being discovered later and proving more resource-intensive to solve (Grinter et al, 1999). Its implementation also leads to information hiding, which in some cases can be detrimental to the project’s success, both in collocated and distributed development teams. In order to make modularization work, it is paramount that less communication and information sharing are replaced with sufficient coordination (Cataldo et al, 2008).

Another problem with the common use of modularization in most development projects is that it takes only a fraction of technical dependencies into account, typically limited to syntactic relationships. These relationships are predicted by observing which modules a given module shares data points with or sends or receives call functions to and from. Syntactic relationships in software can be used to make accurate predictions about limitations imposed by a module on directly interconnected modules (Cataldo et al, 2008). Such relationships prove however an inaccurate basis for predicting task interdependencies resulting of- or changing based on systemic change, which can be categorized into the evolution of product requirements, integration of interfaces made by geographically distributed teams, and dynamic dependencies that are results of continuous design decisions (Cataldo et al, 2007; Kraut & Streeter, 1995; Simon 1962). Syntactic relationships also fail to predict constraints resulting from social factors such as individuals varying competencies and behaviours, as well as organizational structures (Burton & Obel, 1998).

According to Gall and colleagues (1998), logical dependencies provide more accurate predictions of constraints in task structures. They propose that task interdependencies can be uncovered by tracking which files require modifications if a given source-code file is changed. "... when a modification request requires changes to more than one file, it can be assumed that decisions about the change to one file in a modification request depend in some way on the decisions made about changes to the other files involved in implementing the modification request" (Cataldo et al, 2008).

Task interdependencies based on logical dependencies, such as semantic dependencies where the change in behaviour of one module modifies the behaviour of other modules, prove challenging to communicate and require more coordination than constraints predicted based on syntactic relationships. Cataldo and colleagues (2008) argue that collaborative tools and management techniques play key roles in meeting modular system design's demands for coordination efforts, by reducing the gap between perceived and actual task interdependencies.

2.1.3 Socio-technical congruence

The congruence framework is an analytical extension of traditional coordination concepts that are used to explore the discrepancies between the project's actual task interdependencies and the perceived ones (Cataldo et al, 2008). The framework is based on the categorization of software development into two fundamental dimensions: the technical- and the social elements. The technical element consists of the product, the processes, the tasks, and the technologies utilized in the development. The social element consists of the organization and its organizational structure, as well as the individuals and their attitudes and behaviours (Cataldo et al, 2008).

Detecting a project's gaps between perceived and actual dependencies, and then adequately adjusting the project's coordination pattern can have major effects on productivity in a project (Simon, 1962). Empirical evidence from Cataldo and colleagues (2008) show that when coordination patterns in projects are congruent with the coordination needs the resolution time of modification requests is reduced by 32%. It is evident that the socio-technical congruence framework proves useful in detecting how dissatisfactory coordination can impact software development projects (Cataldo et al, 2008).

The greatest limitation of the socio-technical congruence framework is its dependency on archival data, meaning it is not possible to utilize this framework in the early stages of

development projects (Cataldo et al, 2008). The use of standardized design and modelling languages in the earlier project phases can mitigate this flaw, especially graphical representation of the overarching system design (Clements et al, 2002). The combination of these strategies offsets the common problems related to modularization and additionally contributes to aligning developers' coordination patterns with the project's respective coordination needs (Cataldo et al, 2008).

2.1.4 Coordination strategies

Coordination refers to connecting individuals or different organizational parts together to accomplish a set of shared goals (Van de Ven et al, 1976). A key feature of a successful coordination strategy is that it facilitates the exchange of necessary information needed for effective collaboration and autonomous decision making. Its control mechanisms must also ensure that communication is executed in an efficient manner.

Coordination strategies are characterized by three dimensions: “formality (vertical versus horizontal communication), cooperativeness (extent of shared decision-making), and centralization (locus of decisional autonomy)” (Andres & Zmud, 2015). Based on these dimensions, coordination strategies can be divided into two main types: organic strategies that are informal, cooperative, and decentralized, and mechanistic strategies that are formal, controlling, and centralized.

As found in chapter 2.1.1, the common problems associated with projects with a high degree of task interdependencies can be partly solved by improved communication. In regard to coordination strategies, informal horizontal communication channels promote a more frequent exchange of information (Van de Ven et al, 1976). Additionally, transferring decision-making authority to organizational units directly responsible for and working closely with development problems has been shown to increase task execution efficiency as well as reducing decision-making time (Andres & Zmud, 2015). This points towards organic coordination strategies being a possible solution for improving communication.

One of the challenges with organic coordination is according to Andres and Zmud (2015) that its use under conditions of low task interdependence can result in a more costly development process due to the decision-making structure being overloaded with “superfluous information and unnecessary interactions”. The authors observe that mechanistic coordination proves to be more efficient in such projects.

2.1.5 Agile Development

Agile development is defined as a set of software development methods that are iterative and evolutionary (Williams, 2005). Some development methods that are based on the principles of the Agile manifesto are SCRUM, eXtreme Programming (XP), and lean programming (Dingsøy, 2008). Derived from the principles of the Agile Manifesto, all methods of this subset of software development methods seek to address the core principles of the manifesto.

For further defining the goals of the methods mentioned above, we need to define the core principles of the manifesto. Firstly, the manifesto describes collaborative development, where the individuals and interactions are prioritized over the process. Secondly, there is a shift towards minimizing unnecessary work, primarily constructing working software in preference of insignificant documentation. Thirdly, there is a reprioritization for including the other stakeholders early and throughout the project's life cycle. Lastly, the acceptance of unpredictability in any software development project sets the precedent for prioritizing adaptability rather than strict predetermined plans. (Dingsøy, 2008).

The third step of the Agile manifesto ensures good developer to product owner communication and gives the project's end results a higher success rate (Cataldo et al, 2008). Another upside to this third step is that the iterative process forces refinement of the requirements and needs of the product owner by demoing and testing the product at the end of every iteration.

“The customer adaptively specifies his or her requirements for the next release based on observation of the evolving product, rather than speculation at the start of the project. There is quantitative evidence that frequent deadlines reduce the variance of a software process and, thus, may increase its predictability and efficiency” (Williams, 2005, p. 210).

Green and colleagues (2010, Page 322) stated in regard to projects not utilizing an Agile development methodology that: “According to research done by the Standish Group Inc. in 2009; “44% of all projects were challenged (late and over budget), and/or with less than the required features and functions and 24% failed which are cancelled prior to completion or delivered and never used” This indicates that there is a connection between the use of Agile development methods and the success of software projects.

Agile development methods consist of four distinct phases. Phase I is defined as the planning phase, including stating product requirements, user stories, wireframes design, and system architecture. Phase II is centred around analysis and prioritization of the product backlogs and

other Agile artefacts based on the chosen Agile Development Methodology. Phase III is an iterative and continual process of development and design. Lastly, Phase IV describes the release point, where the product is repeatedly tested, as well as being comprehensively documented. Green and colleagues (2010) made in their empirical research a number of observations in regard to Agile development methods. In Phase I and II, rich communication is needed to establish the best possible groundwork for the project. Proper communication of the project's requirements has been shown to increase its rate of success (Andres & Zmud, 2015; Green et al, 2010; Cataldo, 2007). Continuous product demonstrations and reviews at the end of each iterative development cycle are a key part of securing high-quality information flow within a project (Green et al, 2010). The quality of development projects are directly linked to the use of different collaborating and management tools, where a combination of synchronous and asynchronous methods results in the optimal utilization of Agile development methods (Cataldo et al, 2008; Green et al, 2010)

2.2 Technical theories

2.2.1 NoSQL Databases and Relational databases

NoSQL is defined as all alternatives to the conventional Traditional Relational Database Management System (RDBMS). RDBMS is based on the ACID (Atomicity, Consistency, Isolation, Durability) theorem and uses this theorem to secure data consistency and high data integrity by using a strict table structure, where the data is normalized. These types of systems have existed and have been the industry standard for decades. However, relational databases face serious challenges when met with the current market's growing demand for solutions capable of handling the huge amounts of data, often called Big Data, associated with large scale data collection, handling and analytics. In order to designate data as Big Data, it must be in accordance with one or more of the three core criteria: large volumes of data (more than a single standard computer can handle), high velocity (high frequency of data read/writes), and high degree complexity (unstructured data like text documents, video etc.). NoSQL databases are both able to support data with the characteristics of Big Data and provide faster data access and scalability than RDBMS databases (Mason, 2015).

As a result of a multitude of factors, there is currently a major transition in the technology industry from RDBMS to NoSQL databases. The primary reason is that systems based on the ACID theorem are complex and strict, and are not necessary for a wide range of applications.

A major reason for this transition is the growth of data volumes, velocity, and complexity. Velocity, cost, scalability, and ease of development suffer when the feature set has a high degree of complexity and has high volume or velocity. The throughput of a NoSQL database is significantly higher compared to a RDBMS, enabling the adaptation and handling of data in a more efficient manner. Thirdly, relational databases are based on the core philosophy of “One size fits all”. Lastly, the vast majority of RDBMS has expensive and labour-intensive object-relational mapping to create the system, whereas NoSQL databases have no need for this kind of mapping (Strauch, 2011).

A benefit of NoSQL databases is that they are more cost-effective than traditional relational databases. For instance, in a comparison between a RDBMS and a NoSQL database solution by Mason (2015), he observes that traditional relational databases cost on average \$30,000+ per terabyte, whereas an average NoSQL database has a cost of \$1000 per terabyte. The cost savings and performance gains of NoSQL databases are a result of a non-strict approach to data consistency, the use of inexpensive commodity servers, and the adoption of the CAP (Consistency, Availability and Partition tolerance) theorem, which later evolved into the BASE theorem. The BASE (Basically Available, Soft state, Eventual consistency) theorem states that a NoSQL system will over time converge on consistent data, while all data operations are streamlined. (Mason, 2015)

Utilizing NoSQL databases in software development projects has some fundamental challenges. Since the BASE theorem applies to NoSQL systems, a lack of data integrity can occur. This type of data integrity problem can be defined as data with a non-strict structure. To prevent this from being a problem within server-database systems, programmers have to take measures to write complex query code that alters the data to better fit the ACID theorem which resolves this integrity problem (Mason, 2015).

The database structure must be defined and written in advance, which can be a burden if the existing data is not pre-structured. NoSQL-based systems on the other hand have dynamic schemas that facilitate changes without completely rewriting and rebuilding the system structure. When the system encounters new types of data, the database is automatically updated, saving time in the constructing phase of the database system. Scaling in NoSQL differs from RDBMS systems which are scaled vertically. Therefore, relational database systems require more processing power (RAM) or additional CPUs to run existing servers. In contrast, NoSQL can be scaled both vertically and horizontally, enabling different parts of the

database to be scaled independently, meaning the workload is balanced over multiple CPUs, making the NoSQL system more efficient overall. (Mason, 2015).

Document-oriented databases are a type of NoSQL where the data is stored in documents. In a document-oriented database, the data is denormalized, split into collections of different document structures, with no strict structure to each document type. This creates a hierarchical system of collections and documents (Mason, 2015).

Example of a NoSQL document shown below:

```
Book Title: Business Intelligence and
Analytics: Systems for Decision
Support
By Ramesh Sharda, Dursun Delen,
Efraim Turban
Publication Date: 2015
Edition: 10th
Publisher: Pearson
Publisher Location: Upper Saddle
River, NJ.
ISBN-13: 978-0-13-305090-5
```

Figure 1 An example of a NoSQL document, defined as type BOOK (Mason, 2015, p. 262).

2.2.2 OAuth2

The OAuth 2.0 specification defines a delegation protocol for conveying authorization decisions across a network of web-enabled applications and APIs. It is important to note that OAuth2 is not an authentication protocol. This is usually misunderstood among developers because OAuth 2.0 is commonly used inside of authentication protocols, and the process often embeds several authentication events inside of the process. (Richer & Sanso, 2017, p. 236)

To clarify why OAuth 2.0 is not an authentication protocol, we will define what authentication is. Authentication is what tells an application who the current user is and whether they are currently using the application. This is often used in security architecture to prove that the user is who they claim to be. However, OAuth 2.0 is not a technology for validating user claims. OAuth will ask for a token, and if authorized, will get that token

which in turn will be used to access some API. OAuth does not provide any data of who authorized the application or whether there was a user there at all. (Richer & Sanso, 2017, p. 237)

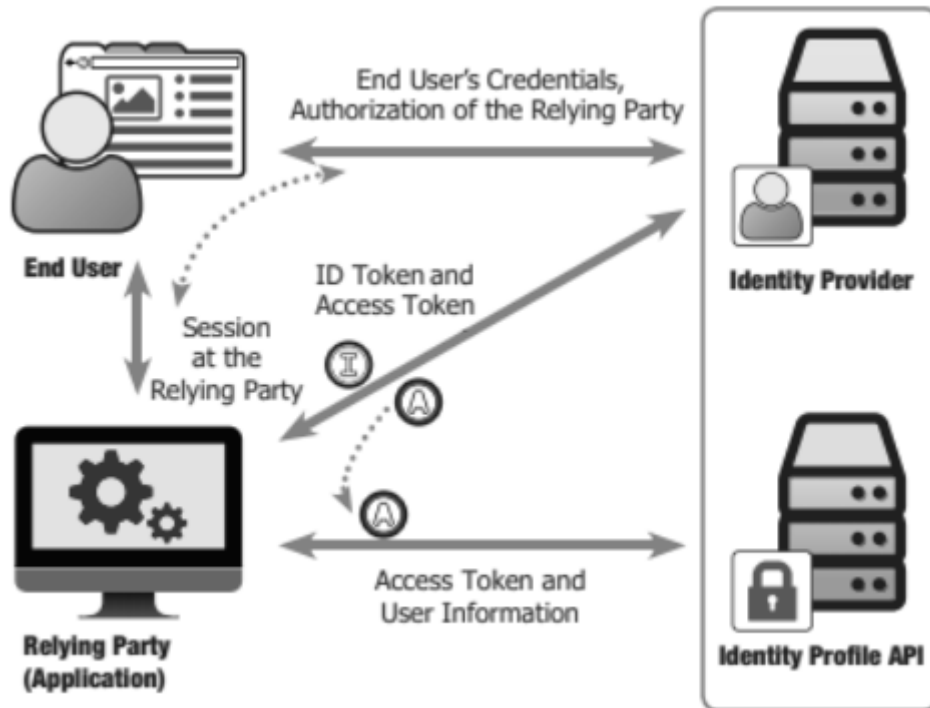


Figure: 2: Components of an OAuth-based authentication and identity protocol (Richer & Sanso, 2017, p. 241)

As shown in figure 2, the identity provider is contacting the client application directly after getting a request from the end-user to do so. The user can decide which credentials the application can get access to, and none of the user's protected credentials is communicated to the client application through the OAuth 2.0 protocol. This is the first benefit of using OAuth 2.0. Furthermore, the user and the client authenticates to one party, and neither needs to impersonate the other (Richer & Sanso, 2017, p. 242).

There are a lot of responsibilities that have to be thought of by developers when utilizing the OAuth 2.0 protocol. Some of the pitfalls of using OAuth 2.0 according to Richer and Sanso (2017, p. 242-246) are:

- Using access tokens as proof of authentication: some developers rely on the idea that when the system gets an access token, the user is authenticated, which is not true. The token itself does not ensure anything about the authentication event, because the token could have been issued from a long-running session or be automatically authorized for

some non-personal scope. To overcome this pitfall some protocols such as OpenID provide a secondary token alongside the access token that communicates the authentication information directly to the client.

- Access to a protected API as proof of authentication: Having a valid access token is not enough to prove that a user is authenticated. This is because some client applications use refresh tokens which may create access tokens without the user being present. The access token itself will persist long after the user is no longer present. This problem can be countered by only checking for user information when the token is fresh.
- Injection of access tokens: This will mainly happen when the client uses the implicit flow, in which the token is passed directly to the client as a parameter in the URL hash. Attackers can pass a token to the client system as if it was requested by that client. This can be mitigated by using the authorization code flow instead of the implicit flow. Therefore this means that the client will only accept tokens from the authorized server's token endpoint.
- Lack of audience restriction: This problem occurs when servers do not provide any mechanism of audience restriction for the returned information. The client can get a token that is not meant to be sent to that client, and if the client uses that access token (which may be valid), that client will get useful information that is not meant to him/her.
- Different protocols for every potential identity provider: This challenges the applications which use OAuth 2.0 because these applications need to have several endpoints for every identity provider. Examples of the difference between providers are having the user ID in different attributes in the token for example one provider will return the *user-id* attribute and another will return *sub-attribute*. This problem can be mitigated by providers using a standard authentication protocol built on top of the OAuth standard.

2.2.3 Breadth-first search algorithm in tree data structure

A tree is a data structure that helps visualize the connections between linked nodes, which can represent any data or objects. When the tree contains a huge amount of these nodes connected with each other, an efficient search algorithm is needed to save time and capacity (Akanmu, 2010).

Breadth-first search or BFS is an algorithm used to graph data or to search through trees. The algorithm starts at the root node and explores all the neighbouring nodes, at the same time it marks the explored nodes and places them in a queue. Then for each of those nearest nodes, it explores their unexplored neighbours, and so on, until it finds the goal.

The algorithm uses the First-In-First-Out model which means the first node added to the queue will be deleted first (Akanmu, 2010).

2.2.4 Microservices

Microservices are defined as software utilizing containers, a way of virtualizing and separating applications or parts of systems in the cloud. Microservices evolved from Service-Oriented Architecture, which means software that emphasizes being self-manageable and lightweight (Pahl, 2016).

A singular microservice is an independent, cohesive, and isolated process, communicating with other similar processes via messages. An application built, deployed, and distributed with each of its modules being a microservice is called a microservice architecture. The antonym for an application not using a microservice-based architectural pattern is defined as a monolithic architecture, where no modules within the application are independently executable and all modules share the same hardware resources (Giallorenzo, 2017).

Monolithic applications struggle to meet the increased demand in the market for systems of high velocity, that are easily maintainable, and have adaptive hardware scaling. Monoliths cause a technology lock-in, as the framework, dependencies, and libraries are hard to adapt and change. This is because the monolith and the modules within are strictly dependent on the parameters set by previous developers. Deployment of these apps is also sub-optimal, because of the non-adaptive deployment environment that causes conflicting requirements of constituent model resources. This non-adaptiveness is comparable to the RDBMS mentioned in chapter 2.2.1 NoSQL Databases in regard to the design philosophy that can be summarized as “one size fits all”. To deploy a monolithic application, the developer must choose a non-adaptive deployment environment that best fits the system requirements. This is dissimilar to Microservices, where the environment is freely configurable and can be adapted to optimally deploy the application. (Giallorenzo, 2017).

Microservices are an option to cope with the complications described above. To combat the problems with technology lock-in and technical dependencies, deployment through

microservices lets the developer gradually push new versions of the system to the production version of the application. The new system versions can co-exist with the old one, and can be gradually modified to communicate properly with older versions if needed (Giallorenzo, 2017).

The ability to gradually deploy changes means continuous integration works well with microservices, as it eases software maintenance. The only constraint on technical dependencies is the functionality behind communication between services. The different modules in microservice applications can be altered at any time, without directly influencing other modules. In other words, the different parts of the system can be modified without requiring rebooting or redeploying the whole application. The use of microservices drastically increases the scalability of systems, as the developer can deploy as many instances of services with their own set load via a workload manager, which in turn implies no duplication of the components pre-existing in the application. Lastly, since microservices are independent and isolated modules a developer can test and investigate the functionality on different modules without affecting others, limiting the scope of a bug occurring. (Giallorenzo, 2017).

3. Choice of technologies and methods

3.1 Methodology of the technical solution

3.1.1 Choice of technologies

3.1.1 a Database System

A NoSQL database was chosen for the project, despite the fact that the use of Relational Database Management Systems is common among application developers. This is because RDBMS has limited scalability and flexibility, as has been explained in chapter 2.2.1. MongoDB was selected as the NoSQL database system. MongoDB uses clusters for hosting the service, which results in low operational costs on their platform. As shown in figure 1, MongoDB uses documents filled with JSON-like information. Some features of Mongo include the use of bson filters to accurately filter data in the server API, indexing on fields, load balancing “sharding” to balance the workload of the system in an efficient manner, and ad-hoc queries that can return specific fields of documents within a given collection.

Designing the domain model knowing the database system would adapt to any new data changes, made the design process lean and effective. Queries made from our database to the database server were of high velocity, benefitting the entire system with quick data fetching. As mentioned by Mason (2015), the cost of operating a RDBMS per terabyte is on average \$30,000, compared to the \$1000 per terabyte cost of the NoSQL systems. Since this system is user-based with no current limit to the maximum number of users and FavN has plans to potentially sell this collaborative tool as a service in the future, keeping the operating costs low increases the potential revenue. Mason (2015) also states that scaling of a NoSQL system can be done both horizontally and vertically, providing the product owner with an adaptive system that can be scaled based on the users' and operators' needs.

One noticeable weakness of NoSQL database systems results from its characteristic complex data resolvers and lack of data integrity within the server structure. Writing the data resolvers is a labour-intensive process, requiring more use of development resources when creating the server API. If not addressed properly the lack of data integrity can cause inconsistencies in data returns to the client or even data corruptions and overwrites.

3.1.1 b Authorization

In the first version of our project vision, we decided to create an authorization system based on email and password and create a user dashboard, but after the second meeting with the stakeholders at (18.02.2021), we agreed to create a login system based on OAuth 2.0. This decision was taken because the stakeholders wanted to focus on the actual functionality of the system and start testing these functionalities as soon as possible. Furthermore, OAuth provides the system with a secure authorization technology as mentioned in 2.2.2 *OAuth*, and the system does not need to save any passwords. The only personal data saved in the database are the user's first, last name, email, and Google ID.

3.1.1 c Development Methodology

Based on the statements of Green and colleagues (2010), Agile Software Development methodology is a key part of securing rich information within the project, improving the overall quality of development, and increasing the project's success rate. Therefore, in addition to the team's former experience, the Agile development method SCRUM was chosen for the project.

SCRUM is an iterative Agile development methodology that divides the project's timeline into work periods called sprints. The SCRUM team meets with the product owner and discusses product functionality. This information evolves into the product backlog, an overview of all the requirements for the planned system and their importance. Each sprint the team selects items from the backlog, then adds estimated hours to completion on each item, together with a burndown chart for the items combined. At the end of every sprint, there is a demo with the product owner, a sprint review, and planning for the next sprint.

SCRUM has different predefined roles, where the core team is made up of the developers and a SCRUM master. The SCRUM master is responsible for the flow of information between the team and product owner, in addition to distributing labour between the members of the main team. Within the team, the developers are the ones carrying out the work. The product owner, in this case Favn, is the entity defining the project goals and vision, communicating their needs and demands to the SCRUM team, as well as giving feedback at the end of each sprint on the sprint reviews.

SCRUM has the added benefit of creating rich process documentation in the form of artefacts. The product backlog and sprint backlogs are examples of these artefacts. A product backlog is a list of items, features, or other system structure changes, containing the main

labour items for the team to work on. Each sprint will have its own distinct backlog called a sprint backlog, where items are obtained from the product backlog and create a basis for the planned work for each sprint.

3.1.2 Key user interaction design decisions

Our technical solution is based upon three key user-interaction design decisions: modularization, visualization of task interdependencies through interactive flowcharts, and information filtering. All three key decisions were based upon Favn Software's primary requirement: increasing productivity in software development projects by improving communication.

The main framework used in the interactive structure of our technical solution is based upon is Modularization. It is characterized by a reductionist approach to complex projects structuring data and is thus a quite good fit for implementation into digital collaboration tools. Modularization of software development projects reduces risk and complexity, as well as increases productivity by lowering demand for communication (Cataldo et al, 2008). Additionally, the compartmentalization of tasks helps reduce the project's overall degree of task interdependencies (Andres & Zmud, 2015). As described in chapter 2.1.1 Task interdependencies, lower degrees of task interdependence is linked to lower coordination and communication needs (Straus & McGarth, 1994; Andres & Zmud, 2015).

As explained in chapter 2.1.2 and 2.1.3, there are certain challenges connected to the use of modularization that can be solved by the use of the socio-technical congruence framework proposed by Cataldo and colleagues (2008). Based on this insight, the implementation of modularization in the technical solution is designed to enable continuous modifications of the task interdependencies as new technical constraints are discovered during development.

The visualization of task interdependencies is a key feature of the technical solution. Coordination is a critical success factor in software projects, and a substantial portion of a project's demand for coordination is driven by its inherent technical dependencies (Andres & Zmud, 2015). According to Cataldo and colleagues (2008), when perceived task interdependencies are aligned with the actual technical constraints of the projects, resolution time for modification requests are drastically reduced. They also point to the use of coordination tools as critical to properly communicating task interdependencies. Based on this, we decided to illustrate dependencies by structuring tasks into a flowchart. This solution

communicates interdependencies in an intuitive way while it clearly separates the description of a task's constraints from the information of the task itself.

The last key element of our technical solution is the filtering of tasks based on their relevance for the user and the simplification of key information. Andres and Zmud (2015) point to the possibility of superfluous information decreasing a project's productivity. They state that it can overload the decision-making process and disrupt already ongoing and effective tasks, leading to wasted time and effort. Based on these insights, several filtering views were designed, where for instance information regarding certain projects and phases are gathered in one view while a user's assigned tasks that are currently ready to be worked on across the whole workspace are collected in another. Simple, yet informative metrics projecting overviews of the progress and status of projects and phases were also calculated based on available data and visualized with graphs.

3.2 Research Method

We conducted an experiment with the goal of investigating whether visualization of task interdependencies in collaboration tools contribute to improved communication in order to answer the research question;

“Can collaborative tools centred around visualization of task interdependence contribute to improved communication in complex software development projects that utilize modularization?”,

Based on Andres and Zmud's (2015) insights on how communication in projects with a high degree of task interdependence can be improved, the experiment will investigate whether the use of collaboration tools contribute to:

- A: Developers spending less time obtaining relevant information in regard to task interdependencies.
- B: Information of higher relevance and accuracy being shared.

The experiment is based on an artificial software development project with the goal of developing a simple e-commerce website consisting of a set number of modularized predefined tasks. A total number of eighteen people participated in the experiment, half with experience as software developers working in Favn and half being computer engineering students.

The participants were divided into groups of three and assigned one of two roles. Each group consisted of one “project leader” that was randomly selected. The rest of the group were

assigned to be “project members”. The “project leader” role was centred around sharing information, whilst the “project member” roles were focused on obtaining and interpreting information. The only communication channels the participants were to use was one randomly selected collaboration tool out of a selection of two. One offering visualization of dependencies, the other containing no functionality directed towards communicating dependencies.

Whilst it might seem preferable to compare the performance of the same groups using both collaborative tools, this would result in the participants being tasked with obtaining information about the same project twice. This familiarity would skew the results in favour of the tool that was used the second time. It was contemplated that by randomly selecting which tools were used first, we could minimize the uncertainty caused by familiarity. However, it was concluded that the uncertainty posed by familiarity will be constant regardless of sample size, whilst the uncertainty from comparing samples consisting of different individuals can be reduced in the future by repeating this experiment and increasing the number of participants.

The decision of limiting the study to the comparison of collaboration tools centred around visualization of task interdependencies, we will call these specialized collaboration tools, and general collaboration tools lacking this focus, unspecialized collaboration tools, was based on the fact that several studies have proved the use of both specialized and unspecialized collaboration tools in software development projects result in improved productivity, lower defect rates, and richer communication (Cataldo et al, 2008; Andres & Zmud, 2015; Clements et al, 2002; Giallorenzo, 2017). This provides a strong indication that collaborative tools centred around visualization of task interdependence contribute to improved communication when compared to the use of no collaboration tools. However, these studies do not investigate the effect of the use of specialized tools compared to unspecialized tools.

Collaboration tool A was the technical solution we developed as part of this bachelor project. It is a specialized tool that visualizes task interdependencies through the use of flowcharts, where each node within the chart is a task and its edges represent dependencies between itself and other tasks. The collaboration tool allows for a limited degree of ambiguity since what the edges signify can be interpreted in different ways. A dependency can be interpreted as the first task demanding completion before work can start on the second task, or merely that some design decisions in the second task should be based upon design decisions in the first task.

Collaboration tool B used in the experiment was Todoist, an unspecialized collaboration tool. The tool consists only of two elements: tasks and sections, a method for grouping and categorizing several tasks together. Since there are no predefined methods for communicating task interdependencies with collaboration tool B, the project leader has a high degree of autonomy in deciding how interdependencies will be communicated through the tool.

The participants designated as “project leaders” received nine tasks constituting a workflow with implied technical constraints based on their description. They were then told to interpret the tasks’ interdependencies and plan a workflow based on their perceived technical constraints. Thereafter, they were tasked with communicating these perceived task interdependencies solely through one of the predetermined collaboration tools.

The participants designated as “project members” were tasked with interpreting the interdependencies the “project leaders” had attempted to communicate. They were explained the basics of the predetermined collaboration tool their group of three were using and then presented the task interdependencies as the “project leader” had communicated them. They were then told to obtain three pieces of information. Firstly, to assess which tasks were ready to be done right now. Secondly, to determine which tasks were dependent on a given task. Lastly, to decide which tasks could be done in parallel with a given task, in other words, which tasks the given task had no interdependencies with.

The participants were measured depending on their respective roles. The participants designated as “project leaders” will post factum self-assess the clarity of the task interdependencies they have communicated through their selected collaboration tool on a scale from 1 (low degree of intelligibility in the shared information) to 10 (high degree of intelligibility in the shared information).

The participants designated as “project members” will be measured in two ways: during the experiment and after the fact. When the participants are tasked with obtaining pieces of information, the time between when they are delegated observation and information finding tasks and when they provide confident answers are recorded. Additionally, they are asked to self-assess the degree of their confidence in their interpretation of the communicated task interdependencies on a scale from 1 (low degree of confidence) to 10 (high degree of confidence).

4. Results

4.1 Scientific results

In order to evaluate the connection between the use of specialized collaboration tools and the time developers spent obtaining information, the time elapsed between the participants designated as “project members” were told to obtain certain pieces of information and they reached a confident conclusion was measured. They were told to obtain the following pieces of information that the participant in their group designated as “project lead” had attempted to communicate through the use of the randomly selected collaboration tool:

T1: Which tasks that are ready to be done right now, in other words; which tasks that are not completed are not dependent on any unfinished tasks-

T2: In regard to one specific task, which tasks that are dependent on the selected task.

T3: In regard to one specific task, which tasks can be done in parallel with it, meaning which tasks are not dependent on the given task and that the given task is not dependent on.

The following table shows an overview of the data that was collected from measuring time (in seconds) for the sample of N=6 assigned to using collaboration tool A. The visualizations of the task interdependencies in the experiments were relatively similar and utilized the edges in the flow-chart in more or less the same manner.

Table 1: Time (seconds) project members spent obtaining information in collaboration tool A.

| Time/Task | T1 | T2 | T3 |
|------------------|--------------------|---------------|--------------------|
| project member 1 | 3,06 | 8,73 | 9,35 |
| project member 2 | 3,58 | 3,63 | 9,5 |
| project member 3 | 6,33 | 30,31 | 12,78 |
| project member 4 | 11,23 | 11,23 | 17,45 |
| project member 5 | 16,73 | 14,18 | 8,76 |
| project member 6 | 9,43 | 10,55 | 21,13 |
| Mean | 8,393333333 | 13,105 | 13,16166667 |

The data points collected and shown above are the same that were collected from the sample of N=6 assigned with interpreting task interdependencies communicated through collaboration tool B, shown in the table below.

Table 2: Time (seconds) project members spent obtaining information in collaboration tool B.

| Time/Task | T1 | T2 | T3 |
|-------------------|--------------------|--------------------|--------------------|
| project member 7 | 41 | 21,55 | 17,25 |
| project member 8 | 36,1 | 38,33 | 26,8 |
| project member 9 | 33,69 | 10,1 | 40,73 |
| project member 10 | 76,89 | 20,35 | 8,71 |
| project member 11 | 62,2 | 36,46 | 26 |
| project member 12 | 25,68 | 50,1 | 44,5 |
| Mean | 45,92666667 | 29,48166667 | 27,33166667 |

In order to assess whether the time spent obtaining information differs depending on which collaboration tool is used, the data points above from each collaboration tool were grouped together by tasks assigned to the experiment’s participants. An Equal Variance T-test was then performed to determine whether the difference between the means of time for each tool was statistically significant. T-tests are hypothesis testing tools designed to investigate if assumptions are applicable to populations (Hayes, 2020). An Equal Variance T-test was selected since the samples that are to be compared are of the same size. The use of the test was based on the assumption that the distribution of time spent interpreting information can be approximated as a two-tailed normal distribution. For all three information-fetching assignments given to the participants, we defined our null hypothesis as: “There will be no significant difference in time spent obtaining information and P-value calculated with the use of the Equal Variance T-Test are also displayed.

Table 3: T1 - time (seconds) project members spent obtaining information.

| T1: Time (seconds) | | | |
|--|---------------------------|-------------|---------------------------|
| project members spent obtaining information | Degrees of freedom | Mean | T-value (Absolute) |
| Collaboration tool A | 5 | 8,393333333 | 2,570543 |
| Collaboration tool B | 5 | 45,92666667 | 2,570543 |

P-value from T-test: 0,001053072467

Table 4: T2 - time (seconds) project members spent obtaining information.

| T2: Time (seconds) project members spent obtaining information | Degrees of freedom | Mean | T-value (Absolute) |
|---|-------------------------------|-------------|-------------------------------|
| Collaboration tool A | 5 | 13,105 | 2,570543 |
| Collaboration tool B | 5 | 29,48166667 | 2,570543 |

P-value from T-test: 0,04250922489

Table 5: T3 - time (seconds) project members spent obtaining information.

| T3: Time (seconds) project members spent obtaining information | Degrees of freedom | Mean | T-value (Absolute) |
|---|-------------------------------|--------------|-------------------------------|
| Collaboration tool A | 5 | 13,161666667 | 2,570543 |
| Collaboration tool B | 5 | 27,331666667 | 2,570543 |

P-value from T-test: 0,03792374447

After the participants were finished with their respective tasks, they were asked to self-assess the degree of understandability of their given presentation of task interdependencies. For both participants designated to be “project leaders” and “project members”, the same rating system was used, a range from 1 to 10, albeit the specific framing of the questions differed based on roles. When comparing the degree of understandability between the two collaboration tools and deciding whether they significantly differ, Equal Variance T-Tests were utilized for the same reasons it was used for comparing the time data. The following null hypotheses H4 and H5 for the rating comparisons were defined: “There will be no significant difference in the ratings of the two collaboration tools degree of understandability”.

Table 6 below displays a comparative view of the ratings from the “project members”. It shows the mean-values, the degrees of freedom, the T-value and the P-value resulting from the Equal Variance T-Test.

Table 6: Reported degree of understandability (Project member).

| Reported degree of understandability (Project member) | Degrees of freedom | Mean | T-value (Absolute) |
|--|-------------------------------|-------------|-------------------------------|
|--|-------------------------------|-------------|-------------------------------|

| | | | |
|----------------------|---|-------------|----------|
| Collaboration tool A | 5 | 8,666666667 | 2,570543 |
| Collaboration tool B | 5 | 4,5 | 2,570543 |

P-value from T-test: 0,00008133583839

The table below shows an overview of the self-assessed ratings from the “project leaders”. It contains the same data points and statistics measures as in table 6 above.

Table 7: Self-reported degree of understandability (Project Lead).

| Self-reported degree of understandability (Project Lead) | Degrees of freedom | Mean | T-value (Absolute) |
|--|--------------------|-------------|--------------------|
| Collaboration tool A | 2 | 8,333333333 | 4,302653 |
| Collaboration tool B | 2 | 4,333333333 | 4,302653 |

P-value from T-test: 0,02239420592

4.2 Product and system design

Our technical solution, named Taskflow, is a full-stack application designed to be used as a coordination tool by tech companies to help them better manage their projects. The application is based on the three key user-interaction design decisions introduced in chapter 3.1.2: modularization, visualization of task interdependencies through interactive flowcharts, and information filtering.

Taskflow has a hierarchical functional structure. On the top of the hierarchy, there is the workspace element, which is a collection of projects and tasks that can be accessed by one or several users. When a user signs up for the first time, a default workplace will be created. This workplace is editable and the user can add members to the workplace by using their emails. It is also possible for the user to create multiple new workplaces. Tasks can either be categorized and structured into projects and phases within projects or be accessed workspace-wide. This allows for the structuring and compartmentalizing of complex projects into several phases or several tasks. Projects contain by default one phase, but can be structured into multiple phases that contain their own tasks. In addition to allowing tasks to be categorized, modularity is also enabled by the subtasks functionality that allows complex tasks to be reduced into smaller and more easily manageable parts. Taskflow modularizes complex projects by splitting these projects into several phases or several tasks.

The application has several user roles as described in the project vision document. Every role has different access and rights. The following table describes these rights:

Table 8: Authorization matrix for the technical solution, showing the functionality that can be accessed by the different user roles.

| Authorization matrix | workplace owner | workplace admin | workplace member | project lead | project member | task owner | task assignee |
|----------------------------------|-----------------|-----------------|------------------|--------------|----------------|------------|---------------|
| Create workplace | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Edit workplace | ✓ | ✓ | | | | | |
| Add/remove a member to workplace | ✓ | ✓ | | | | | |
| Add/remove admin to workplace | ✓ | | | | | | |
| Create project | ✓ | ✓ | ✓ | | | | |
| Edit project | ✓ | ✓ | | ✓ | | | |
| Archive project | ✓ | ✓ | | ✓ | | | |
| Add/remove members to a project | ✓ | ✓ | | ✓ | | | |
| Create phase | ✓ | ✓ | | ✓ | | | |
| Edit phase | ✓ | ✓ | | ✓ | | | |
| Edit roadmap (task dependencies) | ✓ | ✓ | | ✓ | | | |
| Create task | ✓ | ✓ | ✓ | | | | |
| Edit task | | | | | | ✓ | ✓ |
| Add subtasks | | | | | | ✓ | |
| Edit subtask | | | | | | ✓ | ✓ |
| Archive task or subtask | | | | | | ✓ | ✓ |

Taskflow visualizes task interdependencies through an interactive flowchart, hereby referenced to as the roadmap. The roadmap contains all tasks added to a phase. These tasks can be completely independent or structured into a flow based on their dependence on other

tasks. The independent tasks can be worked on anytime during the specified phase, while tasks dependent on other tasks can be worked on once all of the tasks it is dependent on has been completed. The tasks are sorted horizontally based on their dependencies, where the first task from the left side is to be done first. Task interdependencies are visualized by showing tasks as cards with connection lines between them. Furthermore, the tasks in a roadmap have different colours which describe the status of these tasks and help users to focus on what they are really doing or going to do. These colours are listed as following:

- Orange: the task is ready to be worked on.
- Red: the task is overdue, meaning that the deadline has passed. It is still possible to work on the task.
- Green: the task is done, and the tasks dependent on it are now ready to be done unless they are also dependent on other incomplete tasks.
- Grey: the task is not ready to be worked on due to technical constraints and it is not overdue.

These colours are used to filter the information as described in 3.1.2 Key user interaction design decisions. Besides using colours in the roadmap, the information is also filtered in My tasks and Created tasks pages. The tasks in these two pages are filtered based on several criteria such as deadline date and status. These two pages are split into sections where the *overdue* tasks are shown in the first section, and after that, the uncompleted tasks which have the earliest deadline are shown within several sections depending on whether the deadline is today, tomorrow or within the next 7 days. This applies also to subtasks.

The application makes it easy to follow up with the status of a project or a phase. Each project has a progress bar with colours depending on whether the progress is on track or it is taking longer than it should do.

4.3 Engineering results

In this chapter, the results of the development process will be compared to the project goals defined in the planning phase of the project. *A detailed explanation of these goals can be found in the attached file (project vision v 2.0)*. The main engineering goal of the project was to create a full-stack web application that serves as a collaborative tool that communicates

and visualizes task interdependencies. Besides that, the minor goals for the application were that it had to be secure, easy to use, and intuitive.

The main goal was successfully reached. The application was successfully created with almost all of the planned technologies and all of the functional criteria mentioned in the project vision document was fulfilled. The team originally planned to structure the system architecture as Microservices, but after discussing the priorities of the project with Favn halfway through the project, it was agreed that we would focus more on the roadmap and less on scalability in this specific capacity, and Microservices was thus not implemented. The product is also well documented both in the context of the system documentation and for further development. *See the attachments, System Documentation, for a more detailed description of the system.*

In order to ensure the technical solution's ease of use and a high degree of intuitiveness, user testing both during and after the development phase of the project was performed. During the development process, we received continuous feedback from Favn Software and modified the technical solution based on this. There was also a larger user test round at the end of the project in collaboration with a development team from Tryg Forsikring and one project leader from Equinor. This provided valuable feedback for the current state of the solution and suggestions for further work. They stated that the technical solution was both intuitive and easier to use than the collaborative tools they currently utilized for smaller projects. To achieve the needs required for efficient remote user testing, the application has been deployed to the cloud services Heroku and Google Cloud's App Engine.

The final prototype had a server-side test coverage of 95.1 %, 10% more than the original goal stated in the project vision. Besides, integrated testing has been created in continuous integration of the code, so the code will not be published before these tests pass. *See the attachments, System Documentation, for a more detailed description of CI/CD.*

One of the goals mentioned in the project vision was to create a secure application. To ensure that the system was secure, cookies that would only be shared with the hosted server and no third-party cookies were used. Alongside this use of cookies, the system used OAuth 2.0 to authorize users, and the hosted client used an SSL certificate to secure the communication between client and server.

4.4 Administrative results

4.4.1 Scrum Artefacts

The software development methodology SCRUM was strictly used during the entire project. This provided some valuable documentation in the form of SCRUM artefacts, previously mentioned in chapter 3.1.1 c.

The product backlog shows all the functionality planned for implementation, based on the user stories defined in the requirements document. In addition, a kanban board was created based on the product backlog, giving an overview of the work status of each item. Each sprint also has its own sprint backlog that defines the current functionality in focus, their labour estimates in hours, priority, and assignee.

For each sprint, there also exists a written sprint retrospective. A moment of reflection for the team on the last completed iterative cycle. These retrospectives contain what has been accomplished in the sprint, planned items that were not accomplished, team dynamics that improved workflow, and the dynamics that halted workflow. The team dynamics were mostly positive in each sprint, and there were few uncompleted items.

For each of the meetings held with the product owners a written notice and minutes were created, with cases labelled in ascending order. These minutes include quotes in form of questions and answers from the team to the product owners, in addition to feedback for each sprint, also known as sprint reviews. Notices were not created for the remote meetings for user testing purposes, however detailed user story reports were written and contain live feedback from the test subjects.

4.4.2 Project Progress

The Gantt diagram addressed when different stages of development, documentation, and other work categories would take place in a time-related context, in addition to document the estimated overall time used on each unique stage in the project. In general, the overall plan visualized by the Gantt diagram was realistic on what the team could achieve in development, and functioning as a project timeline. The Gantt diagram describes the progress of different phases of the project and their timeline. These phases are outlined in more detail in the burndown charts.

The burndown chart defines the planned goals for each iterative development cycle and describes the progress day by day, and also shows the optimal and planned hours worked per day, compared to the hours actually worked. Based on the average of each of the burndown charts, the overall progress was steady. The graphs of hours worked compared to the optimal workflow were generally matching each other. The estimates in the burndowns were made from the team's initial calculations, however, these were subject to change during development within the sprint.

In sprint 1 progress was a bit ahead of the optimal workflow, with 120 hours completed. Sprint 2 documents implementation of the first functionalities described in the product backlogs with 187 hours completed, however, the implementation of a mailing system was not accomplished and the implementation of refresh tokens moved to a later sprint in order to enable earlier ad hoc testing of the entire system. Sprint 3 resulted in labour items that were mostly non-functional, for example creating a backend and frontend structure. The longest sprint was the third, with 232 work hours planned and 220 completed. In this sprint, the main functionalities implementation was planned and continued over in sprint 4. Sprint 4 had 218 hours planned and 210 hours completed, finishing up the prototype before internal testing with FavN and others. There is a deviation in sprint 4, as the exam period occurs in this period of the project timeline, therefore sprint 4 is planned over a wider range of dates. Sprint 5 was a combination of user testing, adaptations to feedback from the testing, and preparation for remote testing with Equinor and Tryg, with 192 hours planned and 186 completed. *See the attachments, chapter x, for burndown charts and their individual results in more detail.*

4.4.3 Roles and division of labour

The Agile Software Development methodology of SCRUM utilizes different roles for the different roles for development. There is the product owner with their requirements and needs, the SCRUM master making sure the SCRUM methodology is followed and the artefacts produced, and finally the team members. This project's development team consisted of three members, Alexander took the role of SCRUM master, acting as the middleman between FavN and the SCRUM team, while the other project participants acted as SCRUM members. FavN played the role of the product owner.

The division of the labour sheet shows an almost symmetric distribution of total work hours, where all members have worked around 500 total hours.

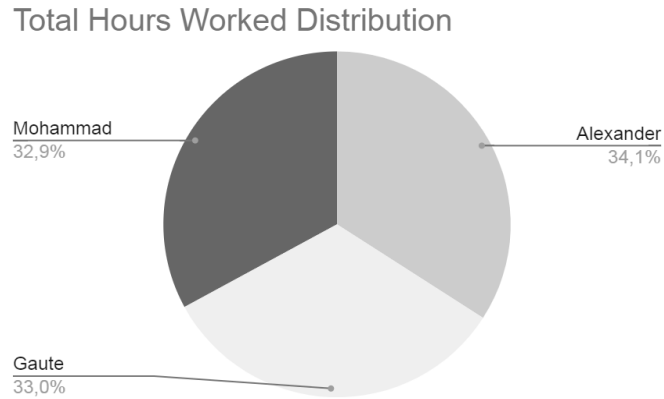


Figure 3. Total hours worked distributed between the team members of the project.

All members have worked a mostly even amount of hours in every defined category of labour, with some differences from member to member. For example, one member of the team focused on the experiments and the analysis of the scientific results, while the two others focused more on properly testing the system with unit testing. *See the attachments, chapter x, for division of labour charts.*

5. Discussion

5.1 Scientific discussion

The purpose of this study was to examine how collaboration tools centred around visualization of task interdependencies can improve communication across two dimensions: the relevancy of the information and the time spent obtaining new information. All five of the hypothesized measurements of communication proved there to be a significant difference between specialized collaboration tools and general ones, as can be seen in table eight below:

Table 9: Summary of research findings.

| Self-reported degree of understandability (Project Lead) | Mean difference (%) in the score for tool A compared to tool B | P-value |
|---|--|------------------|
| H1: There will be no significant difference in time spent obtaining information regardless of which collaboration tool that was used - T1 (seconds) | -81,72% | 0,001053072467 |
| H2: There will be no significant difference in time spent obtaining information regardless of which collaboration tool that was used - T2 (seconds) | -55,21% | 0,04250922489 |
| H3: There will be no significant difference in time spent obtaining information regardless of which collaboration tool that was used - T3 (seconds) | -51,84% | 0,03792374447 |
| H4: There will be no significant difference in the ratings of the two collaboration tools degrees of understandability - Project member (1-10 rating) | 92,59% | 0,00008133583839 |
| H5: There will be no significant difference in the ratings of the two collaboration tools degrees of understandability - Project leader (1-10 rating) | 92,30% | 0,02239420592 |

In regard to time spent obtaining information (H1, H2, H3), our research shows a reduction for the participants assigned to using specialized collaboration tools, compared to the ones who used unspecialized collaboration tools. On average, time spent interpreting task interdependencies and their practical implications was minimized by at least half and up to around 80% when collaboration tool A was utilized.

The p-values for the hypotheses were: $P_{H1} = 0,001053072467$, $P_{H2} = 0,04250922489$, and $P_{H3} = 0,03792374447$. All of these fall within the critical region, below the significance level of 0.05, meaning that our null hypotheses can be rejected and the reductions in time can be determined to be statistically significant. This indicates that the use of collaboration tools centred around visualization of task interdependencies improves communication in software development projects across the first dimension, time spent obtaining information, as defined by Cataldo and colleagues (2008).

This conclusion can be drawn based on the findings of Andres and Zmud (2015) that points to the use of general collaborative tools contributing to better communication compared to projects where no collaborative tools were used, which contributes to improved overall productivity. Assuming a transitive relation, if specialized collaboration tools provide better communication than unspecialized ones, and the use of unspecialized ones result in improved communication compared to no collaboration tools, then the use of specialized collaboration tools in projects will result in better communication than when no tools are used.

These findings are consistent with previous research by Cataldo and colleagues (2008) indicating that the partition of complex systems into modularized tasks combined with the use of collaborative tools can contribute to improved communication in projects and thus increased productivity. This is also supported by Andres and Zmud (2015), who stated that “software development is an information-intensive activity benefiting greatly from organizational structures and processes that facilitate team members’ access to needed design, administrative, and problem-solving information.” They also state that communication channels that facilitate open, clear, and frequent discourse help interdependent team members to develop more fitting task integration strategies.

Our data shows that the participants designated as “project members” rated the degree of understandability (H4) of collaboration tool A higher than that of tool B. On average, “project members” rated the communication facilitated by the specialized tool as having a degree of around 92% more understandability than the unspecialized tool. The difference in ratings had a p-value of $P_{H4} = 0,00008133583839$, indicating a very high degree of confidence.

The data from the “project leaders” ratings of the degree of understandability of the task interdependencies they communicated (H5) indicates a similar relationship between specialized and unspecialized tools as the one reported by the “project members”. Also here collaboration tool A was given an average rating of around 92% higher than collaboration

tool B. With a p-value of $P_{H5} = 0,0223942059$, the null hypothesis of there being no significant difference between the tools can be rejected.

The comparative ratings of the collaborative tools indicate a clear preference for developers both sharing and obtaining information regarding task interdependencies for collaborative tools that facilitate this communication through the use of interactive flowcharts. Such visualizations appear to be more intuitive and less ambiguous than the use of custom sectioning and grouping mechanisms combined with written descriptions of technical constraints.

We theorize the degree of intuitiveness is affected by two factors: how ambiguous the user-interface structure for communicating task interdependencies is and the extent of visual contrast between the task interdependencies and the task itself. During our experiment, it was observed that the “project leaders” allocated to collaboration tool B each structured the information in unique ways within the tools parameters. Moreover, the “project members” interpretations varied across an even broader spectrum. Collaboration tool A on the other hand was used and interpreted in mostly a similar manner. Participants were observed to understand interdependencies quicker across both collaboration tools when the indicators communicating this were visually distinct from the rest of the information.

5.1.1 Limitations

Despite the significant findings in this study, it has a number of limitations that must be taken into consideration. First, as a result of the limited scope and time frame of this bachelor project, only the short term use of specialized collaboration tools has been investigated for small sample sizes. Furthermore, the use of collaboration tools was limited to solely the planning phase of a project. Thus, the findings are not necessarily applicable to how collaborative tools affect communication when the development process has started and when technical constraints change or are created while projects are underway.

The measure of perceived understandability by the participants is also a limitation of the experiment. They were asked to rate the collaboration tool on a scale from one to ten, without being given any reference points or specific criteria for each grade. Instead, they were told a rating of one indicates that no relevant information can be gathered by using the tool, and ten means all of the information conveyed through the tool is understandable for the recipients. This creates some inherent uncertainty in the measurements, as the ratings between one and ten are ambiguous and open for interpretation by the participants. The statistical analysis of

the data indicated however that the results have a high degree of confidence. Even when the uncertainty is taken into account, we are left with a significant difference in the ratings of the two collaboration tools.

The design of the experiment inherently possesses a clear limitation. Since the participants were assigned to use only one of the collaboration tools, some of the differences between the two sample groups can be explained simply by the variety of individuals constituting the sample groups. As mentioned in chapter 3.2 Research methodology, this was decided in order to avoid that the participants interpreted the same project twice and their second performance being affected by their familiarity with their earlier answers. This trade-off is mainly a result of the experiment being a short term investigation, and long term studies with larger sample sizes could have compared the same individuals' use of different collaboration tools without being dependent on comparing the exact same projects across tools to ensure comparability.

Another limitation of the study is that it only provides insights into specialized collaboration tools that use flowcharts to communicate task interdependencies. It does not clarify the relation between specialized tools based on other visualization methods and unspecialized tools. Additionally, it does not provide comparisons between collaboration tool A or other tools with functionality for communicating interdependencies.

5.2 Engineering discussion

In this chapter, we will evaluate our choice of technologies and discuss the consequences and problems that occurred as a result of these decisions. The direct and indirect impact these choices and the selected technologies had on the final product prototype and the development cycles will be discussed. The technical results will additionally be compared to the planned product.

The bachelor issuer had no demand for specific technologies being used for the backend system. However, they required a fast, scalable, and cost-effective solution. Finding a fitting solution to cover these application performance requirements were critical to ensuring an improved final product. In addition, the team prioritized using new and modern technology both for the application's adaptability and further work, and to improve their scope of knowledge within the field of full-stack development. Therefore, the choice of Golang combined with a NoSQL database was made, which covers most of the demands set by Favn

and the team. Additionally, Gqlgen was selected for the API, since graphql is more network efficient than Express REST APIs.

The use of Gqlgen and Golang affected development since the time taken on constructing the data resolvers took longer than the team anticipated, however structuring the backend skeleton and setting up database methods took less time than planned. In terms of the planned results, the resulting backend meets the performance requirements set by FavN, while also fulfilling the team's personal requirements in regard to widening their field of knowledge.

Industry-level security was an important factor to FavN. The resulting product, therefore, utilizes OAuth 2.0 for authentication and allows users to be created with existing Google users. Based on the results from user testing, both companies included in the testing process stated that this improved the useability of the application. The only issue found during user testing was that some of the employees did not have a Google account, and had to take the extra steps to create an account just to test the product. A challenge with using Oauth2 for authentication from the beginning of the project was that this constrained the testing of frontend- and UI features in the earliest phase of development, especially since the implementation took longer than planned.

FavN Software's only requirement for the frontend was that it should be made in ReactJS, in order to easily enable further development if the project was deemed fit. Additionally, Apollo Client was utilized for communicating with the backend through graphql. This resulted in the client being needed to be strict with what data could be received and sent. This resulted in the implementation of the data resolvers being a more labour intensive process than planned. The general complexity in regard to interconnected data visualization and updates, as well as authorization within the frontend, was greatly underestimated, and this resulted in a lot more time spent on these elements than originally estimated.

The ReactFlow Renderer library was used to make the interactive flowcharts visualizing task interdependencies used as roadmaps in the technical solution. The library provided the team with lots of built-in functionality, like a controller bar for the map, fully customizable nodes and minimap, and an easy to implement the system. Using this framework saved a lot of time and proved to be easier than we anticipated. The time cut by using ReactFlow Renderer mitigated the extra time that was spent on the aforementioned challenges that occurred in other parts of the frontend.

The design of the frontend and its requirements for strict data types when sending and receiving data, in turn, meant that unit testing for the server was an extra valuable process, as the team could test the planned database functionalities and their return objects before testing them on the apollo client. This led to improved data integrity and reliability of the server. With a test coverage of around 95%, the resulting backend to frontend communication was predictable and stable for the most part throughout development.

To summarize, in terms of the predetermined technical requirements of FavN the project was overall successful. Our choice of backend technologies fulfilled the requirements, and no unanticipated problems of notable scale occurred despite the team's unfamiliarity with the technologies. Albeit challenging, Oauth2 authentication was also successfully implemented. The predicted problems with the frontend were mitigated and time saved by the use of frameworks, however, unanticipated problems and general system complexity resulted in the frontend being approximately as resource-intensive as planned.

5.2.1 Limitations

Due to the limited technical demands defined by the project vision document, the team faced no limitations with what was planned and required by FavN and those technologies that were implemented to cover these technical needs. However, there were some limitations with the technologies chosen by the team, in particular, some of the backend and frontend libraries had some missing functionality.

The biggest limitation with the backend comes with the problems of the GqlGen library used, a framework for creating a resolver structure based on the current information in the NoSQL schema. This framework works well in terms of generating said resolvers, however, it comes with almost no support for the implementation of other server functionalities, like writing HTTP requests, setting cookies, and other middleware implementations. Therefore, the team had to spend extra time writing their own middleware, as this was the easiest solution to this problem, and the middleware is not optimal for future implementation, as its support for further backend functionality is limited.

For the frontend, the ReactFlow Renderer library used to construct the graphical roadmap had major limitations in terms of adding additional functionality to the existing framework package. This meant that during development it was particularly difficult to implement keybindings, logic checks, and extra functionality revolving around the graphical roadmap.

Furthermore, future development with this library is limited and may result in a conversion to a different graphical node library.

5.3 Administrative discussion

Each sprint being followed by a meeting and demonstration of the results of the sprint ensured good communication between the team and the thesis issuer Favn. This contributed to the overall success of the project. The background for arranging these frequent meetings was based on a study by Green and colleagues (2010), which state that a key part of securing high-quality information flow within a project is through continuous product demonstrations and reviews at the end of each iterative development cycle.

The division of labour was mainly executed to involve every team member in each labour category, which gave the members an opportunity to both work in fields they excel at and gain more knowledge. In general, a trend emerged where a member would have fewer labour hours in categories that they did not prefer. This was a result of the team prioritizing the overall project goals connected to the product over the team's personal goals of gathering knowledge whenever these priorities collided.

The Gantt chart made time allocation more efficient with less time spent on deciding what each work period should be and gave the team frequent deadlines. The decision to use Gantt diagrams to visualize the overall project was based on research by Willams (2005) which stated that; "there is quantitative evidence that frequent deadlines reduce the variance of a software process and, thus, may increase its predictability and efficiency". In the early stages of the project, the actual progress followed the progression planned and outlined in the Gantt diagram. Later in the project, the transition from development to documentation, and the parallel user testing with more development in the last sprint was not accomplished as originally planned. Some of this deviation comes from decisions made by the team to improve the quality of the product prototype by prioritizing user testing over documentation. Other deviations are the consequences of unplanned events occurring. Sprint 4 for example ended up lasting longer than planned because the team failed to properly anticipate how the exam period of March would affect the team members availability.

The burndown charts created from the SCRUM methodology were used to efficiently allocate items of labour to the team members and improved the overall workflow by keeping track of progression within each sprint, making reallocation of work resources to items overdue

easier. These artefacts help planning and to secure a more smooth implementation of the functionality goals set by Favn. There is some functionality planned not implemented which is apparent in the Burndowns, however, in total for all the sprints, there are very few working hours planned that have not been accomplished.

One crucial part of the administrative results was the sprint review and retrospectives. The reviews made the product owners adapt their needs and goals as the product evolved, giving the team a better chance of delivering prototypes that fulfilled their demands at the end of this project.

Most artefacts created during the lifespan of the project helped to document the entire process from planning to end of development, describing both planned workflow and completed workflow, providing comprehensive documentation of administrative results.

5.4 Ethical discussion

There are very few ethical issues related to the experiment, as it was performed on consenting adults with full knowledge of its implications. Therefore, the team members have focused on reflecting on the computer-ethical issues around the technical solution created for this project. Computer ethics falls into both of the categories of applied- and professional ethics (Søraker, 2013).

The main ethical issue connected to professional ethics in the research work and the application created is privacy. Privacy is a known issue in computer ethics because every computer system has to save data about the user using it (Søraker, 2013). However saving or sharing data with others without permission from the user is neither ethically correct nor in line with GDPR, the privacy laws of the European Union. Taskflow uses Google as a provider for OAuth 2.0, which lets the user choose what kind of data to share with Taskflow. The application saves only email, first name, second name, and Google ID, and this information is saved when on user creation. In addition, Taskflow does not track how the user interacts with the application, except when they create projects, phases and tasks. This is in accordance with one of the moral pillars of professional ethics for developers, that you should collect no more data than what is necessary. It can however be argued that forcing users to have a Google user in order to create a Taskflow-user can be problematic, as Google is known for not limiting their data collection efforts.

The second element of professional ethics for developers that is relevant for this project is the respect of copyright. Throughout our research and the development process, the team members have been strict with referencing studies that have been used and using frameworks in accordance with their licenses.

Since there are several members who have contributed to the research and the development process, the team members have reflected on the division of responsibilities as an ethical issue connected to applied ethics. As mentioned in chapter 5.3 *administrative discussion*, the division of labour was mainly executed to involve every team member in each labour category, this is done to involve everyone in the different phases and elements of the project. This allowed the team members to share their experiences with the other team members, and better learn how to use new technologies.

5.5 Member reflections

The group's work dynamic has been good. The internal communication of the team was mostly excellent despite working remotely a majority of the time. This has been made possible by the usage of synchronous and asynchronous channels, as well as comprehensive and continuous documentation during development. Every member has worked within every category of work, including attending the obligatory lectures and the workshop in the relevant subject. There has been little absence of leave and all members have worked around 500 hours and attended meetings where their presence was necessary. The allocation of tasks between the members has been efficient. Working remotely was viewed as mostly positive, as we got more freedom to work when we were available during the day.

6. Conclusion and further work

This study contributes to the software management literature by providing initial insights into the connection between the use of collaborative tools centred around visualizations of task interdependencies and productivity in software development projects. Our empirical evaluation of the data gathered in our experiment answers our research question and concludes that the use of specialized collaboration tools improves communication across both of the dimensions defined by Cataldo and colleagues (2008). Significant improvements within the time spent obtaining information as well as the understandability of the discourse contribute positively to the overall productivity of software development projects.

Although the experiment only included a flowchart-based collaboration tool, our findings combined with the current literature in the field indicate a broad positive relationship applicable to some degree to all specialized tools. Despite the limitations of our findings, it is indicative that the design of the collaborative tool used plays a key role in software development projects.

6.1 Future work

While this study answers the research question within its limitations, there are several elements about the relationship between collaborative tools and communication in projects that could use further research. These elements can be categorized as improving the accuracy of the findings of this study and exploring the relationship between these findings and other collaborative tools and methods.

In order to improve the accuracy of our findings, a long term observational study should be held. This would mitigate several of the limitations our short term study faced, such as measuring how the use of the collaborative tool affected communication when used outside of a project's planning phase. Additionally, by observing developers over longer time periods across different collaboration tools and projects, one could compare how the same individuals perform and thus minimize the uncertainty connected with comparing completely different sample groups. The observation of collaboration tools in actual use would also allow for more accurate measurements than time recording and simple rating, such as long term effect on the success-factors of the observed projects. A long-term observational study with larger

sample sizes would provide more accurate data, yet would also be a lot more resource-intensive.

The logical next step based on the findings in this study that is not as resource-intensive as the aforementioned long term observational studies is expanding the experiment to include several other collaboration tools. The perhaps largest limitation of our findings is that we cannot draw conclusions for other collaborative tools other than our technical solution.

Whilst the results of this study indicate that specialized tools improve communication more than unspecialized ones and the complete lack of collaboration tools, they do not say anything conclusive about different combinations of task interdependence visualizations and of which that are the most effective.

In regard to further development of the technical solution, this will be Favns responsibility after the end of this project. Some of the main features planned for further development include; a kanban board to view the current sprint tasks and their status, a component that shows the overview of the resources in a project and how they are allocated, a system for inviting new members by mail, and a shift in general design focus from text to graphical drag and drop cards. The technical future work suggested here is the result of the user testing with Favns Software, Tryg Forsikring, and Equinor.

7. References

1. Kharrazi, A. (2019). *Encyclopedia of Ecology (Second Edition): Volume 4: Resilience*. ScienceDirect: <https://doi.org/10.1016/B978-0-12-409548-9.10751-1>
2. Cataldo, M., Herbsleb, J.D., Carley, K.M. (2008). Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. *ESEM: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, October 2008*, 2-11. <https://doi.org/10.1145/1414004.1414008>
3. Andres, H.P., Zmud, R.W. (2015). A Contingency Approach to Software Project Coordination. *Journal of Management Information Systems: Volume 18, 2002 - Issue 3*, 41-71. <https://doi.org/10.1080/07421222.2002.11045695>
4. Straus, S. G., & McGrath, J. E. (1994). Does the medium matter? The interaction of task type and technology on group performance and member reactions. *Journal of Applied Psychology*, 79(1), 87-97. <http://dx.doi.org/10.1037/0021-9010.79.1.87>
5. Van de Ven, A.H., Delbecq, A.L., and Koenig, R. (1976). Determinants of coordination modes within organizations. *American Sociological Review*, 41, 2 (1976), 322–338. <https://doi.org/10.2307/2094477>
6. Cataldo, M. et al. (2007). On Coordination Mechanism in Global Software Development. *In Proceedings of the International Conference on Global Software Engineering (ICGSE'07)*, Munich, Germany.
7. Baldwin, C.Y., Clark, K.B. (2000). *Design Rules: The Power of Modularity*. MIT Press.
8. Parnas, D.L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of ACM*, 15, 12, 1053-1058. https://link.springer.com/chapter/10.1007/978-3-642-48354-7_20
9. Burton, R.M. and Obel, B. (1998). *Strategic Organizational Diagnosis and Design*. Kluwer Academic Publishers.
10. Kraut, R.E. and Streeter, L.A. (1995). Coordination in Software Development. *Communications of ACM*, 38, 3, 69-81.
11. Simon, H.A. (1962). The Architecture of Complexity. *In Proceedings of the American Philosophical Society*, 106, 6, 467- 482.
12. Grinter, R.E., Herbsleb, J.D. and Perry, D.E. (1999). The Geography of Coordination Dealing with Distance in R&D Work. *In Proceedings of the Conference on Supporting Group Work (GROUP'99)*, Phoenix, Arizona.
13. Gall, H. Hajek, K. and Jazayeri, M. (1998). Detection of Logical Coupling Based on Product Release History. *In Proceedings of the International Conference on Software Maintenance (ICSM '98)*, Bethesda, Maryland.
14. Clements, P., Garlan, D., Little, R., Nord, R., and Staffor, J. (2002). Documenting Software Architectures: Views and Beyond. *25th International Conference on Software Engineering, 2003. Proceedings, 2003*, 740-741. <https://doi.org/10.1109/ICSE.2003.1201264>

15. Mason. R.T. (2015). NoSQL databases and data modeling techniques for a document-oriented NoSQL database. *Proceedings of Informing Science & IT Education Conference (InSite) 2015*, 259-268. Retrieved from: <http://proceedings.informingscience.org/InSITE2015/InSITE15p259-268Mason1569.pdf>. Regis University, Denver, Colorado
16. Richer, J. & Sanso, A. (2017). OAuth 2 in Action. *Manning Publications, Shelter Island, New York*. 236-241, Retrieved from: <http://manning-content.s3.amazonaws.com/download/a/6008973-e9d9-469a-8ac5-1cb0e6b91c99/SampleCh13.pdf>
17. Williams. L, (2005). A Survey of Agile Development Methodologies, 209-225. *North Carolina State University*. Retrieved from: <http://www.fet.uwe.ac.uk/~p-chatterjee/2011/readings/AgileMethods.pdf>
18. Dingsøyr T. , Nerur S., Balijepally V.G. , Moe N.B. (2008). A decade of agile methodologies: Towards explaining agile software development. *Trondheim, Norway, Norwegian University of Science and Technology*. Retrieved from: <https://doi.org/10.1016/j.jss.2012.02.033>
19. R. Green, T. Mazzuchi, S.Sarkani. (2010). Communication and Quality in Distributed Agile Development: An Empirical Case Study. Retrieved from: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.193.3125&rep=rep1&type=pdf>
20. Strauch. C. (2011). NoSQL Databases. *Stuttgart Media University*. Retrieved from: <http://csce.uark.edu/~xintaowu/BDAM/nosql dbs.pdf>
21. Pahl. C, Jamshidi. P. (2016). Microservices: A Systematic Mapping Study. *Department of Computing, Imperial College London, London, U.K*. Retrieved from: <https://www.scitepress.org/Papers/2016/57855/57855.pdf>
22. Dragoni. N, Lanese. I, Larsen. S.T, Mazzara. M, Mustafin. R, Safina. L. (2017). Microservices: How to make your application scale. *Technical University of Denmark*. Retrieved from: <https://arxiv.org/pdf/1702.07149.pdf>
23. Giallorenzo. S, Dragoni, Lafuente. A.L, Mazzara. M, Montesi. F, Mustafin. R, Safina. L. (2017). Microservices: yesterday, today, and tomorrow. *Technical University of Denmark*. Retrieved from: <https://arxiv.org/pdf/1606.04036.pdf>
24. Akanmu T. A., Olabiyisi S. O., Omidiora E. O., Oyeleye C. A., Mabayoje M.A. and Babatunde A. O. (2010). Comparative Study of Complexities of BreadthFirst Search and Depth-First Search Algorithms using Software Complexity Measures. *Proceedings of the World Congress on Engineering*. retrieved from: https://www.researchgate.net/profile/Stephen-Olabiyisi/publication/257562301_Comparative_Study_of_Complexities_of_Breadth_-_First_Search_and_Depth-First_Search_Algorithms_using_Software_Complexity_Measures/links/0deec51e43365ae011000000/Comparative-Study-of-Complexities-of-Breadth-First-Search-and-Depth-First-Search-Algorithms-using-Software-Complexity-Measures.pdf
25. Johnny Hartz Søraker (2012-2013). Innføring i computer-etikk. AITeL, HiST. retrived from: http://www.iie.ntnu.no/fag/_innfng/johnny/Soraker_Innforing_i_Computer-etikk_HI_ST_2013.pdf

26. Hayes, A. (2020, March) T-Tests. *Investopedia*:
<https://www.investopedia.com/terms/t/t-test.asp>

8. Attachments

Table of contents

| | |
|--------------------------------------|-----|
| 8.1 Vision Document | 53 |
| 8.2 System Documentation | 68 |
| 8.3 Process Documentation | 80 |
| 8.4 Equinor Testing Report | 122 |
| 8.5 Tryg Forsikring Testing Report | 124 |
| 8.6 Tryg Forsikring Testing Report 2 | 127 |
| 8.7 Requirements Documentation | 128 |

8.1 Vision Document

Version 2.0

Audit History

| Date | Version | Description | Author(s) |
|-------------|----------------|--|----------------------------|
| 13.01.2021 | 1.0 | First Draft, the stakeholders' first vision of the project | Alexander, Gaute, Mohammad |
| 18.02.2021 | 2.0 | Second draft, the priority of functional properties is changed, some functions are deleted and other functions are added | Alexander, Gaute, Mohammad |

8.1.1 Introduction

The purpose of this document is to determine the TaskFlow project vision. We will address and define the main issue. Characterize the different users and stakeholders. Specify the user environment and user needs. Exemplify the products role, dependencies and describe the functional and non-functional properties of the system.

This project will be carried out by computer engineering students as a bachelor thesis at the beginning of spring semester 2021. The students are linked to NTNU at the IE faculty in the IDI institute.

Our product can be described as a project management tool for tech and devops companies to smaller agile operations, with a focus on task interdependencies, moving away from a focus on microservices.

8.1.2 Product and issue summary

8.1.2.1 Issue summary

| | |
|---------------------------------|---|
| Main issue(s) | <ol style="list-style-type: none">1. Managing a bigger project and its logistics2. Updating and maintaining a complex system |
| concerns | DevOps / Tech companies of all sizes |
| which results in these problems | <ol style="list-style-type: none">1. Communication problems, ineffective planning, resources wasted / hard to allocate tasks properly.2. Data corruption during updates, difficulty deploying a system to foreign hardware, issues |

| | |
|--------------------------------|---|
| | scaling a traditional system without wasting server resources. |
| a successful solution could be | <ol style="list-style-type: none"> 1. Easier to balance workload over the devs, better project overview. 2. Easier to deploy, update, maintain and scale compared to most systems made in 201x. |

8.1.2.2 Product summary

| | |
|------------------------|--|
| For | Any tech company with projects (DevOps) |
| who | requires a task manager for any type of project |
| Name of product | will be TaskFlow |
| be | an easy and intuitive resource saver and logistic planning tool |
| unlike | ClickUp and Jira |
| our product advantages | interactive timeline charts/roadmap with dependencies, project statistics such as burndown charts, smartDaily todo board with automatically prioritize tasks |

8.1.3 Stakeholders & users

8.1.3.1 Stakeholders

| Name | Detailed description | Role during development |
|------|--|-------------------------|
| Favn | Thesis issuer and guide, will review the functional properties of the system and the system as a whole | Product Customer |
| NTNU | Thesis guide and thesis evaluator, supports the team with guidance if needed | Supportive role |

8.1.3.2 User Profiles

| Work/ educational background | Detailed description | Expected degree of technology knowledge | Role in software |
|------------------------------------|--|--|---|
| IT/Tech | DevOps or other tech company that uses some type of project/task management software/tool to solve logistics surrounding an operation. Needs to be able to organize a large number of detailed tasks in a structured way (such as a Kanban board) and view which tasks can be worked on right now. | High | Project lead, project member |
| Designer | Participates in projects, either by completing a design during specific phases or continuously iterating on the design through the whole project lifespan. Focuses on longer processes rather than smaller tasks. Needs structured information on due dates and general requirements. | low | Project member |
| Economics, sales and management | Administration who uses data gathered from any type of management software to calculate costs set due dates, resource planning e.g. Needs to easily be able to follow up designated tasks and projects. | low | Organization administrator, Project monitor |

8.1.3.3 User roles in TaskFlow

| Role | Role description | System utilization |
|----------------------------|---|--|
| Organization administrator | Has total overview over project scope, running costs, development progress and users involved with the specified system and other users in the same organization. | Administrates organization members, roles and details. Determines organization goals. |
| Project lead | Has an overview of team members, what tasks are being worked on and their dependencies, together with e tasks' progress. | Plans and creates the project. Manages the project members, roles and functional tasks within the project. Follows up on the individual tasks done by members. |
| Project Monitor | Participates in planning projects, Scrum rituals and enforces resource and time limitations. | Tracks progress in current projects and functions as a Scrum master |
| Project member | Has an overview of the current tasks for today and for the given sprint | Utilizes the task software by finding and doing the relevant tasks assigned to the given project member. |

8.1.3.4 The user environment

Here we will define the user environment, using Favn as the prime example. In this case, TaskFlow might be used as a simple to-do software or a more detailed scrum planner. The software might be usable with other systems, like Clockify, and Git. Favn might also use the

project management tool to define and work towards bigger or more abstract goals, with an overview of all current running projects.

8.1.3.5 Summary of users' needs

This summary will give a general description of the user's needs, a more detailed description of all user cases can be found in the product backlog.

| Case | Priority | Involves | Today's solution | Proposed solution |
|---|----------|--|--|--|
| Management and Project leaders need to be able to assign tasks and follow up on these | High | Management, Project leaders, Developers, Designers, Sales-people | A combination of Kanban boards in Github and Notion, as well as notes based on oral communication and meetings | Create software that combines structured task boards and smaller project-independent tasks |
| Employees need to have an overview of their tasks | High | Users | A combination of Kanban boards in Github and Notion, as well as notes based on oral communication and meetings | Create a dynamic overview of the tasks that can currently be done, that sorts tasks based on due dates of tasks and estimated hours required to complete them. |

| | | | | |
|---|---------------|-------------------------------|---|---|
| <p>Employees need to be able to easily differentiate between completed tasks and tasks that needs work, and see when assigned tasks dependent on others are available to be worked on</p> | <p>High</p> | <p>Users</p> | <p>Some functionality in ClickUp allows for some task dependencies, but are not so easily manageable on a larger scale.</p> | <p>A software that allows the users to edit their own tasks</p> |
| <p>A user needs to get updated when something is changed in the project they are involved in or with a task they are assigned to</p> | <p>Medium</p> | <p>Users, Project Lead</p> | <p>None</p> | <p>Send an email to all the involved users when something is changed</p> |
| <p>users need to be able to create their own workplace and add people to the workplace. Beside being able to mark employees as an admin to manage the workplace.</p> | <p>High</p> | <p>Users, workplace owner</p> | <p>It is possible in different technologies to create a workplace and add people it</p> | <p>An easy and understandable interface so all the users who have an account on TaskFlow will be able to create their own workplace</p> |

| | | | | |
|--|-------------|--|---|--|
| <p>A member of a workplace needs to be able to create projects in the workplace and assign a project lead to that project</p> | <p>High</p> | <p>Users, project lead</p> | <p>It is possible to create a project in Gitlab, but everyone will have the same rights to edit the project and add tasks to it</p> | <p>A page where it is possible to add a project for all the workplace members, and it should be possible to edit the project, assign a project lead, and follow with the progress of the whole project</p> |
| <p>A project lead or a workplace admin/owner needs to be able to create one or several phases in a project, update these phases and create a graphical view of the tasks and their dependencies in a phase</p> | <p>High</p> | <p>User, project lead, workplace owner/admin</p> | <p>In Gitlab, it is possible to create epics and a roadmap of the issues in the epic, but is not possible to add and dependencies on these issues</p> | <p>The phase interface will be a part of the project page, and has its own progress bar. Moreover it will be able for the user with the right access to edit these phases and create a roadmap for the whole phase and add dependencies to these roadmaps.</p> |

8.1.3.5 Alternatives to our product

There are several alternatives to our product such as:

- Jira
Jira is a software development tool used by agile teams to plan, track and release software. Jira Software supports Scrum, Kanban, a hybrid model or another unique workflow.
- Clickup
ClickUp is a cloud-based collaboration and project management tool suitable for businesses of all sizes and industries. Features include communication and collaboration tools, task assignments and statuses, alerts and a task toolbar.
- Gitlab
GitLab is an open DevOps platform, delivered as a single application.

8.1.4 Product overview

8.1.4.1 The product's role in the user environment

TaskFlow is a system where users can create a workplace and invite other users to the workplace. The product is going to be used as a tool to plan a project, a phase or some individual tasks. However, The system is going to be the platform where the user is going to save, create and save the tasks, and assign these tasks to other users. Moreover the user might be able to mark these tasks as done or doing and track the progress of a project or a phase. The project leaders will be always able to edit projects so they can reach the idea of an agile process.

8.1.4.2 Assumptions and dependencies

The target group for this system is the largest dependency. If the target group is changed during the project we would need to redefine the users and the user roles in their entirety. This system software does have some prerequisites, e.g. the type of projects the software is to manage. Tech and development projects will match well with this software, like short and small scaled systems to bigger scrum projects in large scale.

A field in which this software is not ideal to be used is the construction industry, where there are some guidelines that are not taken into account. This software is also not suitable for maintenance projects or continuous tasks that do not have a clarified start or end point, with no deployment.

8.1.5 Functional properties of the product

| Function | Description | Priority | Est. Hours |
|---|--|----------|-----------------|
| General security in storing data | Safe to create a user with sensitive information | High | 20 |
| Site navigation | The user should be able to easily navigate between the pages in the webapp | High | 10 |
| create a user and log in using Google OAuth 2.0 | A function that make it possible for user of the system to get access to their saved information | High | 40 |
| Edit user | Update the personal information when needed, this includes changing the user's password also | low | 0 "future work" |
| Log out | A function that | Medium | 15 |

| | | | |
|---------------------------|--|------|-----------------|
| | deletes locally stored authorization tokens and redirects the user to the landing page | | |
| Decide to stay logged in | User receives a access token with longer durability than the default tokens | Low | 0 “future work” |
| Create / edit a workplace | Users of the system will be able to create and organize their workplaces | High | 30 |
| Create / edit project | A user will be able to create a project workspace where they can manage members and tasks | High | 20 |
| Edit project information | A project lead will be able to edit the information surrounding the project, e.g. name, publicity, scope and so forth. | High | 10 |
| Create/edit phases | A project lead can create different phases for the project, e.g. | High | 50 |

| | | | |
|--|--|------|----|
| | planning phase, documentation phase, sales/release phase and so on | | |
| Create/edit tasks | A project member can create a task in a given project. This task can have subtasks and have dependencies to other tasks | High | 50 |
| Monitor full project status overview | A project lead or a project monitor can see the progress on the project, the tasks completion rate, burndown charts, completion percentages, sprint progress and ... | High | 30 |
| Create / edit a graphical roadmap of tasks with dependencies | Most of the tasks in the projects are hidden until they are relevant and ready to be worked on, and the team has an overview of the entire work process of the project | High | 40 |
| Users can Create / | The main task me be | High | 30 |

| | | | |
|---------------|--|--|--|
| edit subtasks | thought of as an epic when it has several subtasks | | |
|---------------|--|--|--|

8.1.6 Non-functional properties and other requirements

- Web-application in React
- Backend created in Golang
 - Monolith
 - GraphQL
 - Should be relatively fast
- The project should be made with data-gathering in mind
 - NoSQL-database
- Docker-images for:
 - Docker-compose
 - Database
 - Server
 - WebApp
- Mono-repo in Github
- Google-authenticator
- WCAG coverage
- 85+% test coverage on server

8.1.7 References

- <https://www.trustradius.com/products/jira-software/reviews>
- <https://www.softwareadvice.com/project-management/clickup-profile/>
- <https://about.gitlab.com/what-is-gitlab/>

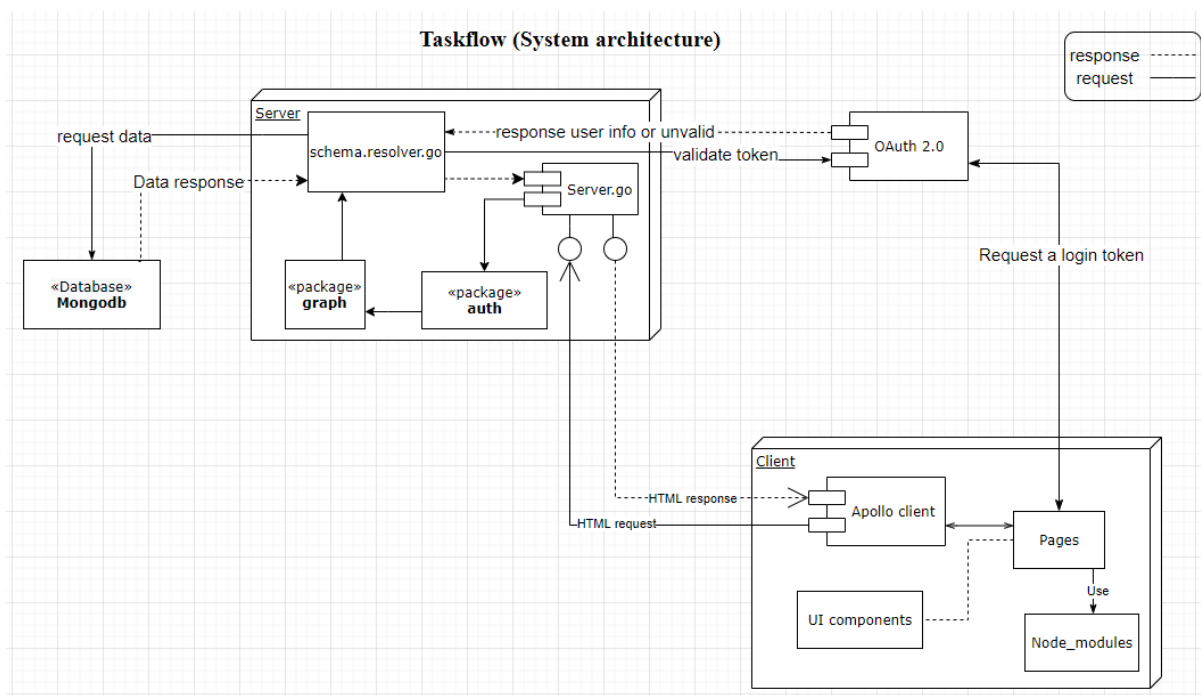
8.2 System Documentation

Version 1.0

8.2.1 Abstract

This document's purpose is to illustrate and model how the system created in this project operates on different levels of abstraction. We look at how the system's biggest parts communicate and interact. Further we go deeper into details about the project structure, usage of the system, source code documentation, system securities, continuous integration and other models describing different parts of the whole project. The document is connected to the Taskflow thesis main report.

8.2.2 System Architecture



The figure describes the system architecture of Taskflow application. The two main components are client and server. The client is where the system has user interfaces and an apollo client which will send and receive http requests to and from the server. when a page requires some data, a graphql schema will be created and sent as an http-request to the server. However, when the server receives a request from the client it will find out which type of request it is, then the request will be sent to the authentication package to find out whether the request is sent from a logged in user or not logged in user. After that the request will be sent to the graph package so the resolvers will be invoked. When a resolver gets a graphql schema

it will handle the request contained in the schema, and the resolver most likely will request or send data to the database, after that the resolver will return a response to the client using the server.

There is a special use case when the system needs to log in a user. The login request starts from the login page, this page will send a request to OAuth provider to log in a user, if the user is successfully logged in the provider will send back an authentication token, this token will be sent to the server as an http request to the login endpoint. Login resolver will send a validation request to the OAuth provider to make sure that the token is valid, after getting the response from the provider the request will be handled as a normal request described in the previous section.

8.2.3 Project Structure

```
|— Tasksflow-Bachelor /  
| |— client /  
| | |— node_modules /  
| | |— public /  
| | |— src /  
| | | |— components /  
| | | |— graphql /  
| | | |— pages /  
| | | |— index.js  
| | |— Dockerfile  
| | |— README.md  
| |— service1 /  
| | |— auth /  
| | |— db /  
| | |— graph /  
| | |— helpers /  
| | |— templates /  
| | |— tests /  
| | |— Dockerfile
```



This figure shows how Taskflow project is organized. The main two folders inside the project are client and service1.

The client folder contains all folders and files needed to create UI-components. inside the client a node_module folder can be found, this folder is where all the installed external libraries are saved, public folder is where the public data can be saved for example public pictures, and src folder is where the programmed code is, inside src folder there is a components folder, this folder contains all the UI components needed to create Reactjs components. The graphql folder contains all queries and mutations which are sent to the server inside html-requests, under graphql there is a file called index.js, this file is used to run the client.

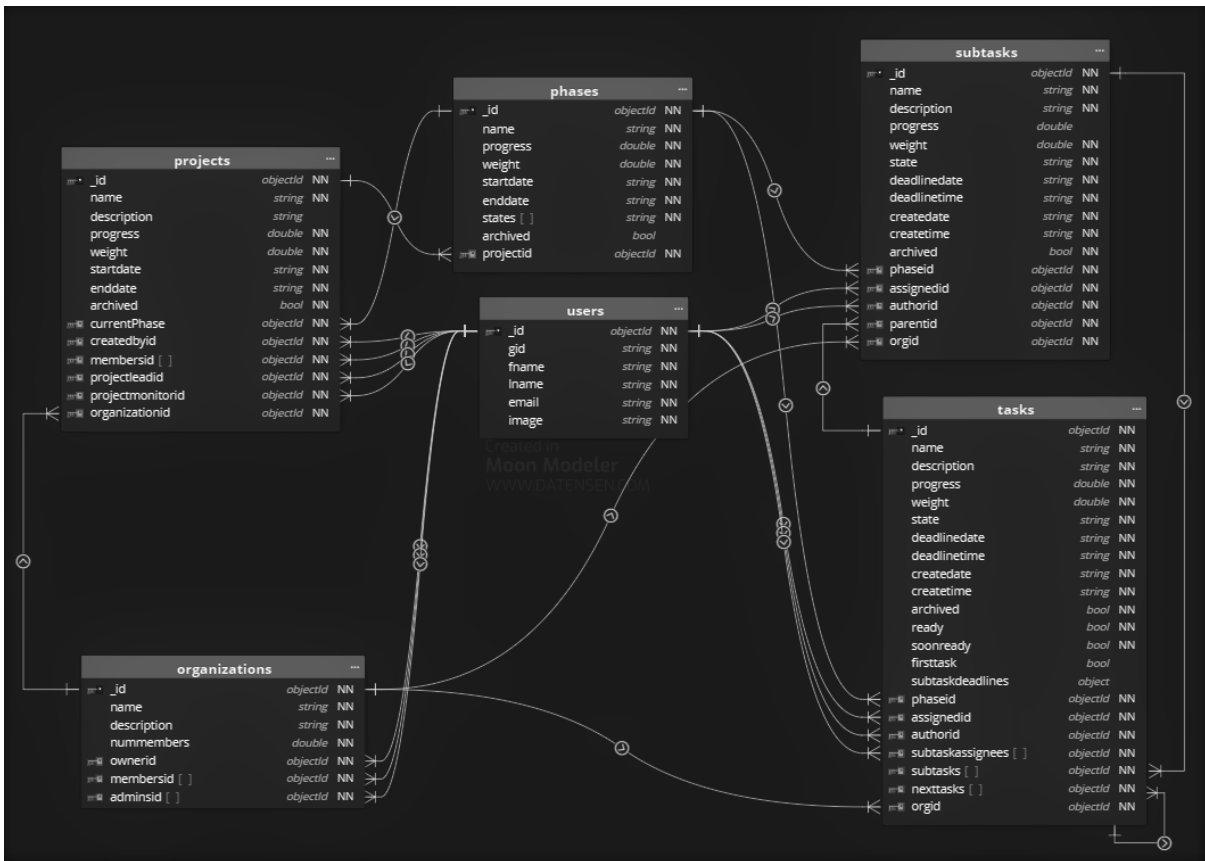
The service1 folder contains the server of the application. All the contained folders are necessary packages to create server endpoints.

Finally, the Github folder is where the .yml file is used to create actions in Github.

Class Structure

Based on the nature of the server structure a class diagram is not necessary, as there are no connections or interactions between the different “classes” (structs) on a class based level.

8.2.4 Enhanced entity-relationship (Database model)



This database model shows the relationship between entities in Taskflow database. Taskflow uses MongoDB as the main database where all data is stored.

The entity-relationship diagram shows clearly that the user entity is the core of the database. This corresponds to the product's use cases where the user is controlling the most of the functions. The rest of the dependencies are built up hierarchically, where the organization which is also called workplace in frontend is on the top of the hierarchy, and projects are next on, under projects there is the entity of phases and the entity of tasks at the same level depending on whether the created task is contained on a phase or not. finally the subtasks entity comes under tasks.

8.2.5 Server API resources

The server uses graphql endpoints. The documentation of these endpoints are attached as html pages. See the attached folder "graphql Docs"

To run these html pages, the folder can be unzipped and inside the folder called schema, there is a file called index.html, running this file in any web browser will give an overview of the documentation of the endpoints. For example the schema of a project will be shown as following:


```
- "owner": 1,  
- "tokenExpiration": 1620405436,  
- "userID": "6023f04ae4c6144d12529e47"  
- }
```

- Refresh token: It is valid for 72 hours from creation time. The token is used to refresh the access token as long as he/she uses the system. Like access tokens we store tokenExpiration and user ID in the token encrypted by using JWT.

To make the system even more secure we have written middleware functionality that checks both tokens and validates the user on login. If we have a valid user, they will get information from the database, resolved from the graphql server. However, if the user is not a valid user, either one of the tokens has a false signature, or some hacker tries to use XSS or an injection to access server data, they will be denied authorization by all secure methods resolving data from the server.

In the client tokens the middleware sets some claims based on data stored in the database. In these claims are user workplace roles, split into “Owner” and “Admin”. Only these roles can edit certain information, for example the workplace members, name and description. If a user tries to edit these parameters, either by having a logged in user or via injection, it will be denied by the resolver if the right to edit is not set in the claims. Also, an injection or XSS attack to set the claims by making a false token will be denied access by the middleware. We also avoid Cross-site injection by using cookies and disabling the sameSite attribute and use our domain as domain attribute.

8.2.7 Installation and executables

List of the most important external libraries:

- [react-flow-renderer](#): A library for building node-based graphs (we used it to create a graphical roadmap)
- [jwt-decode](#): A library that helps decoding JWTs tokens which are Base64Url encoded.
- [graphql](#): The JavaScript reference implementation for GraphQL.
- [react](#): a JavaScript library for creating user interfaces.
- [react-sprint](#): spring-physics based animation library.

- [@apollo/client](#): Apollo Client is a fully-featured caching GraphQL client.
- [gqlgen](#): Go library for building GraphQL servers.
- [jwt-go](#): A Golang implementation of JWT, used to create and encode tokens

The project is a web application and will be hosted for development and a production build. You can find it here <<https://task-flow-1.herokuapp.com/>>

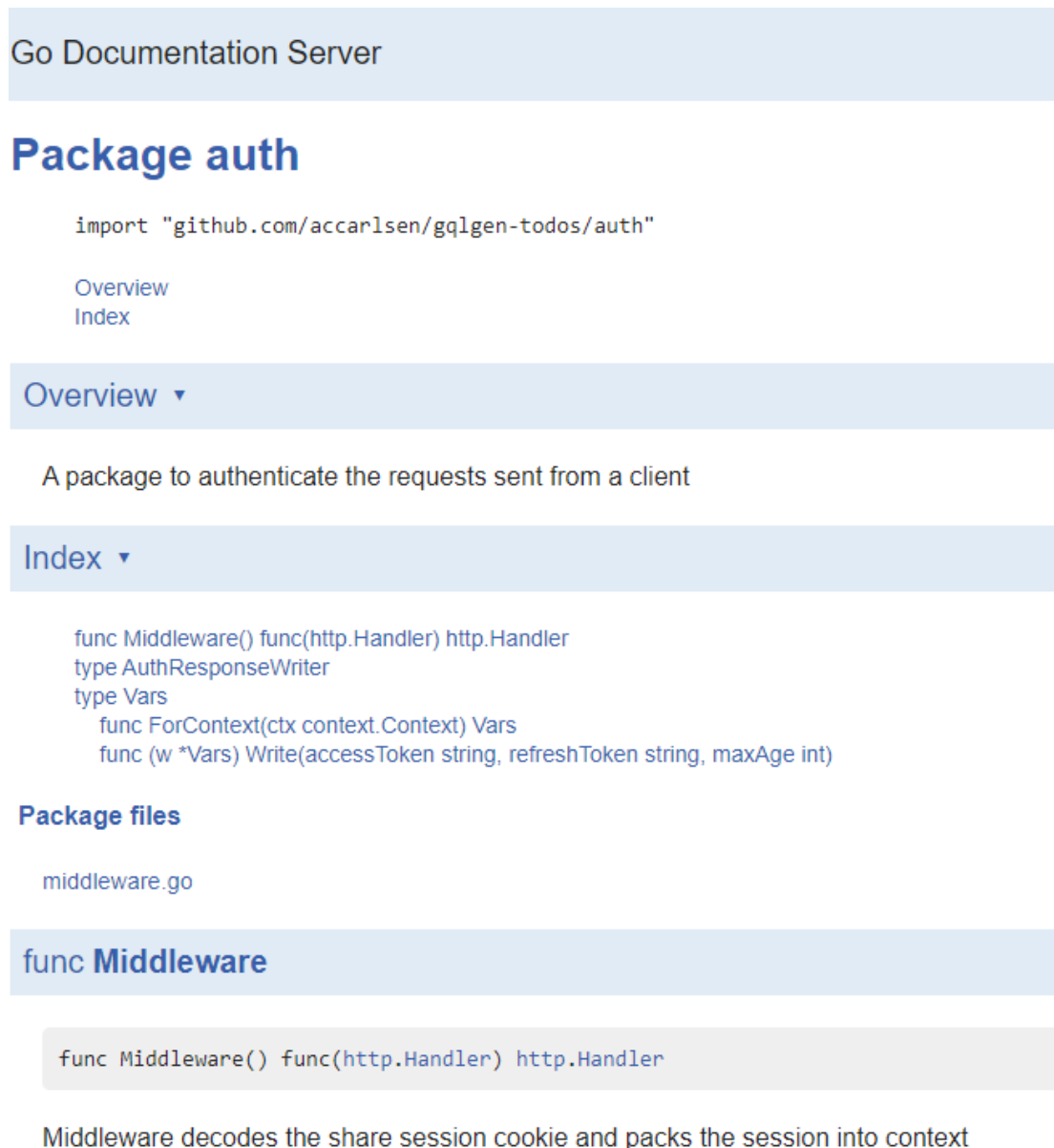
However, if you want to install the system, you have to contact FavN to get access to the repository, then you have to pull the code from “<https://github.com/favnsoftware/taskflow-bachelor>”.

- Server:
 - download go lang from: <https://golang.org/dl/>
 - create a .env file in the following route “taskflow-bachelor/service1”, then fill the .env file with the following:
 FRONTEND_HOST= <Your frontend host>
 DOMAIN = <your domain to let the server know which domain to set cookies. for example, localhost>
 - on your terminal run the following command: cd service1
 - then run: go get
 - and finally run: go run server.go
- Client:
 - download nodejs from: <https://nodejs.org/en/download/>
 - add .env file in the following route “taskflow-bachelor/client”, and fill it with the following variables:
 REACT_APP_BACKENDHOST= <your backend host for exmaple, <http://localhost:8080/query>>
 - on your terminal run: cd client
 - then run: npm install
 - and finally run: npm start

8.2.8 Source code documentation

Documentation of the source code can be found as an attachment folder called *GoDoc*. The documentation is html pages which describe how the code in the server side (service1) was developed, and this documentation can be used for further development.

To run these html pages, the folder can be unzipped and inside that folder there is a file called `index.html`, running this file in any web browser will give an overview of the documentation of the code. An example on how Godoc looks like:



The screenshot shows the Go Documentation Server interface for the package `auth`. At the top, it says "Go Documentation Server". Below that is the package name "Package auth" in a large blue font. A code block shows the import statement: `import "github.com/accarlisen/gqlgen-todos/auth"`. There are two links: "Overview" and "Index". The "Overview" section is expanded, showing the text: "A package to authenticate the requests sent from a client". The "Index" section is also expanded, showing the following code: `func Middleware() func(http.Handler) http.Handler`, `type AuthResponseWriter`, `type Vars`, `func ForContext(ctx context.Context) Vars`, and `func (w *Vars) Write(accessToken string, refreshToken string, maxAge int)`. Below the code is the section "Package files" with a link to `middleware.go`. The "func Middleware" section is expanded, showing the signature: `func Middleware() func(http.Handler) http.Handler`. The description for this function is: "Middleware decodes the share session cookie and packs the session into context".

This documentation is created by [the godoc library](#). However, the server is a [gqlgen](#) server which creates resolvers automatically based on a file called `schema.graphqls`, these resolvers

are created but not exported, so it is not possible to create GoDoc for these resolvers.

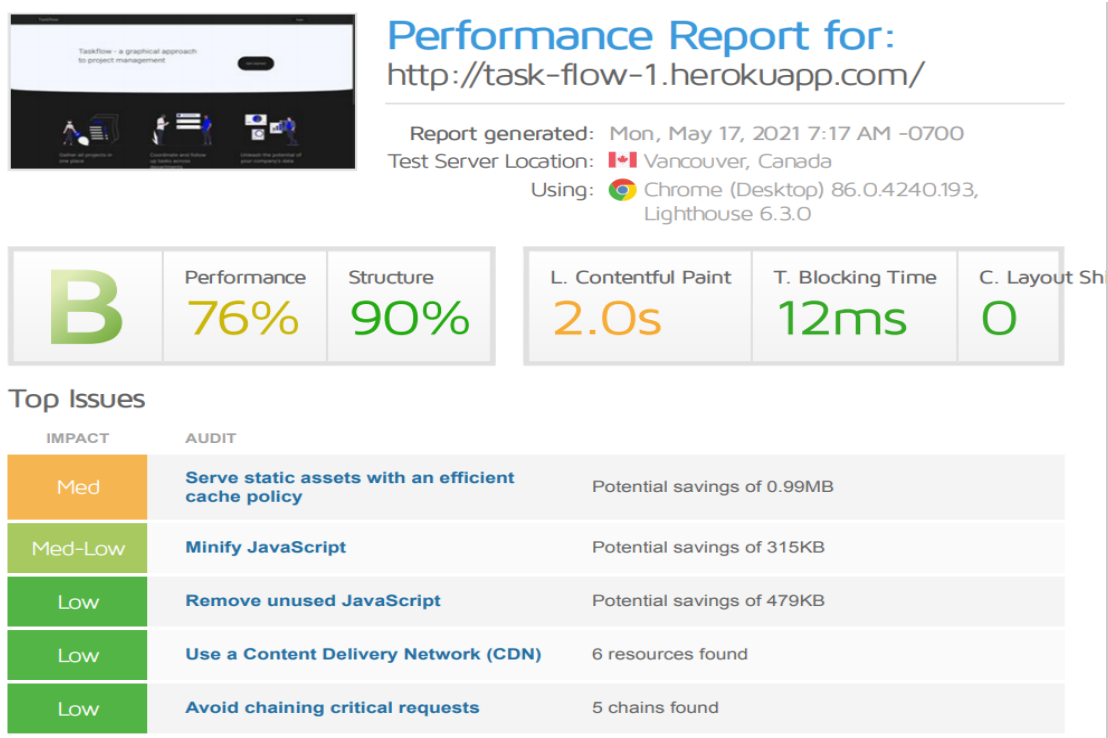
8.2.9 Continuous Integration and testing

Github Actions are used to continuously run tests and deploy code. The application is splitted into two systems, client and server, both systems are completely separated from each other. Therefore the hosted application can be found as two projects on Heroku. To ensure that these systems work as expected when a new code is pushed to Github main branch, we run CI/CD in Github Actions. The main goal is to create Docker images of each system, run tests on these images and when these tests are passed, the code can be pushed to the main branch and the docker image can be deployed on the cloud service Heroku.

The first step is to run tests. To do so a Golang environment is set up in an Ubuntu container, after setting up the environment the tests can be run, and if these tests are passed, the code can be pushed to the main branch. However, to deploy the system on Heroku two different docker images are created, and these two images are deployed using a github library called [heroku-deploy](#).

When it comes to testing the code is tested by using [the unit testing package for Golang](#). This means only the server code is unit tested and the client is in general manually tested while developing. However the unit tests for the server are continuously created to cover all database methods. These tests are created at the same time when the developers started planning database structure and updated throughout the project. Running the command `<go test -cover fmt>` in `taskflow-bachelor/service1` gives the percentage tested code. When delivering the product **95.1%** of the code on the server side is covered by tests.

The deployed versions of the code are also tested on a platform called [GTmetrix](#). [The website deployed on Heroku](#) gets 91% reate on code structure and 70% on performance. The performance results showed that the website needs some improvements, mainly serving static assets with an efficient cache policy.



While the website hosted on App Engine gets 100% rate on code structure and 97% on performance.



Performance Report for: <https://taskflow-303114.ew.r.appspot.com/>

Report generated: Wed, May 19, 2021 7:59 AM -0700
 Test Server Location: 🇨🇦 Vancouver, Canada
 Using: 🌐 Chrome (Desktop) 86.0.4240.193, Lighthouse 6.3.0

| | | | | | |
|----------|-------------|------------|---------------------|------------------|--------------|
| A | Performance | Structure | L. Contentful Paint | T. Blocking Time | C. Layout Sh |
| | 100% | 97% | 686ms | 0ms | 0 |

Top Issues

| IMPACT | AUDIT | |
|---------|---|----------------------------|
| Med-Low | Serve static assets with an efficient cache policy | Potential savings of 287KB |
| Low | Use a Content Delivery Network (CDN) | 6 resources found |
| Low | Avoid an excessive DOM size | 52 elements |
| Low | Avoid enormous network payloads | Total size was 357KB |
| Low | Avoid long main-thread tasks | 1 long task found |

8.3 Process Documentation

Taskflow

version 1.0

8.3.1 Labor Agreement

versjon 26.08.2020

Avtale mellom partene

Bedrift/virksomhet: Favn Software AS

Student(ene): Alexander Carlsen, Gaute Wierød Rønning & Mohammad Al Nayef
og NTNU, IDI AIT

Oppg.nr: 79 Studentprosjekt Strategi & koorinderingsverktøy

Studieprogram: Dataingeniør

Gjennomføring

Studenten skal gjennomføre et studentprosjekt i samarbeid med bedriften/virksomheten. IDI AIT veileder arbeidet faglig. Det er utarbeidet retningslinjer for gjennomføring av studentprosjekt som beskriver oppgavefordeling og hvordan studentprosjekter gjennomføres. Retningslinjene tar også opp ansvarsfraskrivelse, opphavsrettigheter og tilgjengelighet med muligheter for individuelle avtaler.

Ansvarsfraskrivelse

Instituttet er ikke ansvarlig for eventuelle ødeleggelser som studenten måtte påføre oppgavestillers utstyr direkte eller som følge av programvare studenten lager og/eller bruker, eller som studenten på annen måte medvirker til.

Opphavsrett og tilgjengelighet

Når ikke annet er avtalt, eier studenter selv den IPR (immaterielle rettigheter) de skaper som en del av studier/studieopphold ved IDI AIT. Alle resultater er åpent tilgjengelig. Opphavsretten reguleres av Åndsverksloven. Avtaler som inngås mellom IDI AIT og studenter skal som minimum sikre instituttet rett til å bruke generert IPR til utdannings- og forskningsformål. IDI AIT skal også motta en vurderingskopi av arbeidet inkludert eventuell kildekode. NTNU oppfordrer til publisering av alle bachelor- og masteroppgaver.

Marker med kryss det som gjelder denne oppgaven:

- Normalsituasjonen: Studentene har selv alle rettigheter knyttet til resultatet fra bacheloroppgaven, med de unntak som er beskrevet over.
- Oppdragsgiveren har rettighetene og kan utnytte produktet kommersielt og videreutvikle produktet/metoden. Instituttet vil ikke utnytte produktet kommersielt, men vil kunne arbeide videre med den grunnlagskompetansen som er vunnet gjennom prosjektet, som beskrevet over.
- Resultatene fra arbeidet legges ut som OpenSource iht lisens _____ (Se <http://creativecommons.no/lisenser>).
- Bacheloroppgaven (det skriftlige arbeidet) skal være undergitt utsatt offentliggjøring i __ år (maks 3).

Oppdragsgiver er selv ansvarlig for å avtale håndtering av eventuelle konfidensielle opplysninger med veileder/sensor og studenten(e).

Denne avtalen er underskrevet i 3 – tre - eksemplarer hvor partene skal ha hver sin.
Studentene må sende inn kopi av avtalen til admin ved NTNU IDI for arkivering og sette evt. utsatt publisering!

07.05.2021, Trondheim

(dato, sted)

Sveinung Øverland
Bedrift/virksomhet

[Signature]
Veileder ved IDI AIT

Alexander Carlsen
Student(ene)

8.3.2 Gantt Chart

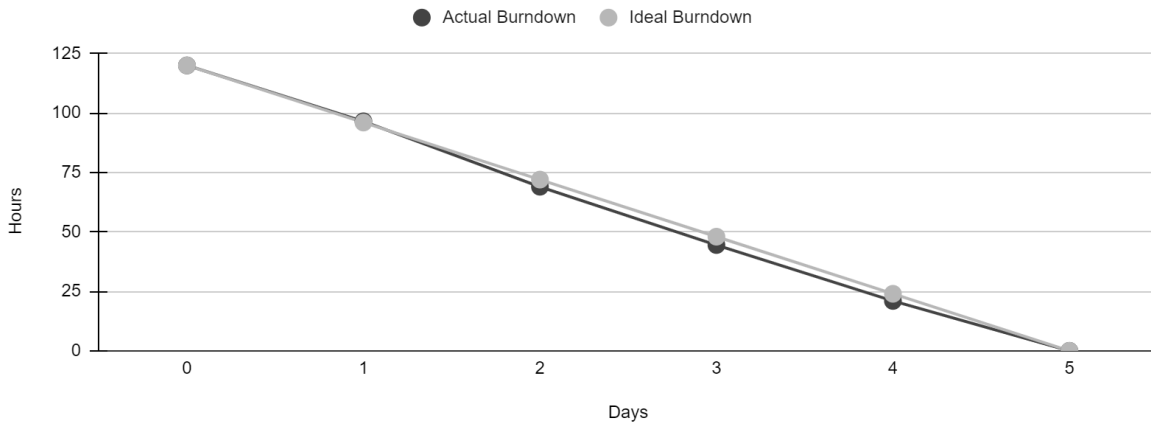
| | Start Date | End Date | Timeline | Status |
|-------------------------------------|--------------|--------------|----------|---------------|
| Taskflow | Jan 13, 2021 | May 20, 2021 | | |
| Plan the work and meeting with Favn | Jan 13, 2021 | Jan 16, 2021 | | Complete ▾ |
| Set up stack | Jan 14, 2021 | Jan 27, 2021 | | Complete ▾ |
| Create tests and set up CI | Jan 17, 2021 | Jan 22, 2021 | | Complete ▾ |
| Set up general frontend structure | Jan 15, 2021 | Jan 28, 2021 | | Complete ▾ |
| Users | Jan 25, 2021 | Feb 12, 2021 | | Complete ▾ |
| Workplace | Feb 4, 2021 | Feb 12, 2021 | | Complete ▾ |
| Project & Phases | Feb 10, 2021 | Mar 11, 2021 | | Complete ▾ |
| Task | Feb 14, 2021 | Mar 14, 2021 | | Complete ▾ |
| Graphical roadmap | Feb 19, 2021 | Mar 25, 2021 | | Complete ▾ |
| Subtasks | Mar 1, 2021 | Mar 19, 2021 | | Complete ▾ |
| Fix user roles and access to data | Mar 25, 2021 | Apr 8, 2021 | | Complete ▾ |
| Fix UI/UX and add feedback | Mar 17, 2021 | Apr 30, 2021 | | Complete ▾ |
| Refactor backend, tech debt | Apr 21, 2021 | Apr 24, 2021 | | Complete ▾ |
| User testing 1, in Favn | Apr 16, 2021 | Apr 24, 2021 | | Complete ▾ |
| Implement feedback | Apr 21, 2021 | Apr 30, 2021 | | Complete ▾ |
| User testing design, a/b? | Apr 28, 2021 | May 1, 2021 | | Complete ▾ |
| User testing 2, wider test group | May 1, 2021 | May 12, 2021 | | Complete ▾ |
| Implement feedback 2 | May 21, 2021 | Jun 26, 2021 | | Future work ▾ |
| Bachelor report | May 6, 2021 | May 20, 2021 | | Complete ▾ |
| Documentaion | Jan 14, 2021 | May 19, 2021 | | Complete ▾ |
| Kanban | May 21, 2021 | Jun 21, 2021 | | Future work ▾ |
| Strategic goals, management view | May 22, 2021 | Jun 22, 2021 | | Future work ▾ |
| Reminders | May 23, 2021 | Jun 23, 2021 | | Future work ▾ |
| | | Burndown | | |

8.3.3 Scrum Burndown Charts

8.3.3.1 Sprint 1

Burndown Chart

Total hours remaining after each day



Total time estimated for the sprint: 120 Hours

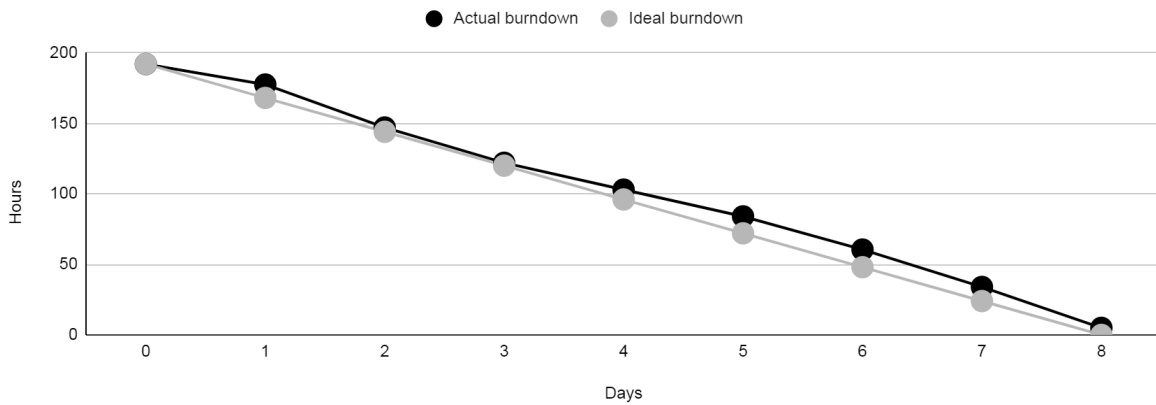
| | | | | | | |
|------------------------|-----|------|----|------|----|---|
| Actual Burndown | 120 | 96,5 | 69 | 44,5 | 21 | 0 |
| Ideal Burndown | 120 | 96 | 72 | 48 | 24 | 0 |

| Sprint Task | Responsible | Estimate | Remaining hours on task as of day n | | | | |
|----------------------------------|------------------|----------|-------------------------------------|----|-----|----|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 |
| Make react project w/CleanUp | All (Main Alex) | 4 | 1 | 0 | 0 | 0 | 0 |
| Set up MongoDB | Alexander, Gaute | 8 | 5 | 0 | 0 | 0 | 0 |
| Make testing Go server | Gaute, Mohammad | 36 | 33 | 26 | 10 | 0 | 0 |
| Connect the stack | All | 10 | 10 | 0 | 0 | 0 | 0 |
| Scrum Rituals | All | 3 | 2,5 | 2 | 1,5 | 1 | 0 |
| Set up Git/CI | Gaute, Mohammad | 5 | 4 | 4 | 4 | 0 | 0 |
| Frontend basics | Alexander | 12 | 9 | 5 | 2 | 0 | 0 |
| Any Documentation | All | 22 | 16 | 16 | 13 | 6 | 0 |
| Make backend types/Testing types | Gaute, Mohammad | 20 | 16 | 16 | 14 | 14 | 0 |

8.3.3.2 Sprint 2

Burndown Chart

Total hours remaining after each day



| | | | | | | | | | |
|------------------------|-----|-------|-----|-----|-----|----|------|----|---|
| Actual Burndown | 192 | 177,5 | 147 | 122 | 103 | 84 | 60,5 | 34 | 5 |
| Ideal Burndown | 192 | 168 | 144 | 120 | 96 | 72 | 48 | 24 | 0 |

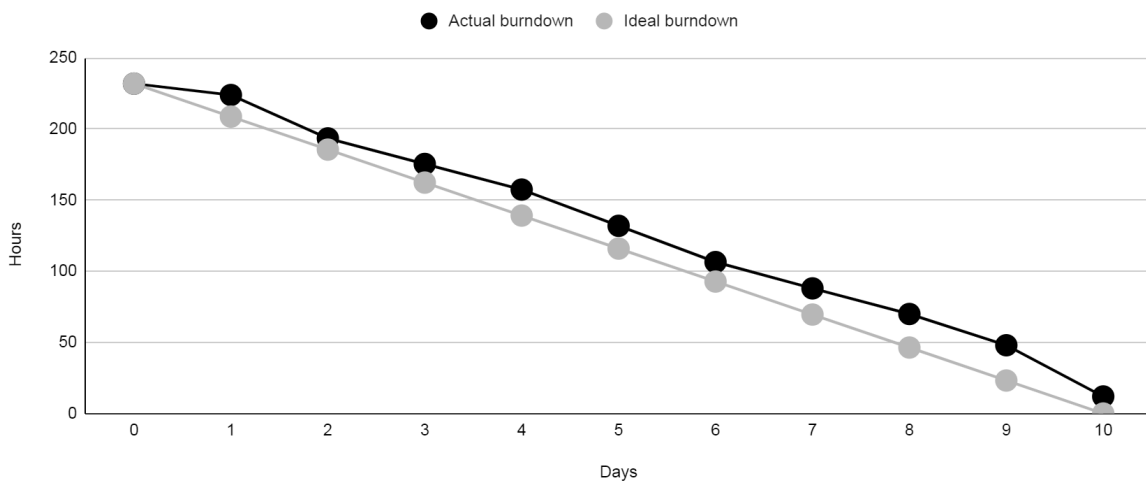
| Time est. total: | 192 | Estimate | Remaining hours on task as of day n | | | | | | | |
|--------------------------------------|--------------------|-----------------|--|----------|----------|----------|----------|----------|----------|----------|
| Sprint Task | Responsible | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Create a user | Moha/Alex | 10 | 7 | 4 | 3 | 1 | 1 | 1 | 0 | 0 |
| Login with OAuth 2.0 | Moha/Alex | 25 | 23 | 18 | 10 | 7 | 5 | 4 | 0 | 0 |
| Log out | Moha/Alex | 10 | 10 | 10 | 10 | 10 | 10 | 7 | 5 | 0 |
| Other token functionality | Moha/Alex | 15 | 13 | 13 | 13 | 7 | 3 | 3 | 2 | 2 |
| userAccess in resolvers and frontend | All | 15 | 15 | 15 | 15 | 15 | 15 | 8 | 1 | 0 |
| Any documentation | All | 5 | 5 | 5 | 5 | 5 | 2 | 0 | 0 | 0 |
| Scrum rituals | All | 2 | 2 | 1,5 | 1 | 0,5 | 0 | 0 | 0 | 0 |
| Create & edit organization | Gaute | 18 | 13 | 11 | 8 | 7 | 5 | 3 | 0 | 0 |
| Organization overview | Gaute | 15 | 15 | 9 | 7 | 6 | 4 | 2 | 2 | 0 |
| Invite / Manage members in org | Gaute | 10 | 10 | 7 | 3 | 2 | 2 | 2 | 2 | 2 |
| Make necessary endpoints | Gaute | 10 | 9 | 7 | 4 | 2 | 1 | 1 | 1 | 1 |

| | | | | | | | | | | |
|----------------------------------|-----|----|-----|-----|----|-----|----|-----|---|---|
| Resolvers backend (With testing) | All | 20 | 19 | 10 | 7 | 5 | 3 | 1 | 1 | 0 |
| Create project | All | 7 | 7 | 7 | 7 | 7 | 7 | 6 | 4 | 0 |
| Project overview | All | 10 | 10 | 10 | 10 | 10 | 9 | 8 | 6 | 0 |
| Connect Users/Projects/Orgs | All | 10 | 10 | 10 | 10 | 10 | 10 | 8 | 5 | 0 |
| Demo Styling / Testing (Orgs) | All | 10 | 9,5 | 9,5 | 9 | 8,5 | 7 | 6,5 | 5 | 0 |

8.3.3.3 Sprint 3

Burndown Chart

Total hours remaining after each day



| | | | | | | | | | | | |
|-------------------------|-----|-------|-------|-------|-------|-----|-------|------|------|------|----|
| Actual Burndown: | 232 | 224 | 193,5 | 175,5 | 157,5 | 132 | 106,5 | 88 | 70 | 48 | 12 |
| Ideal Burndown: | 232 | 208,8 | 185,6 | 162,4 | 139,2 | 116 | 92,8 | 69,6 | 46,4 | 23,2 | 0 |

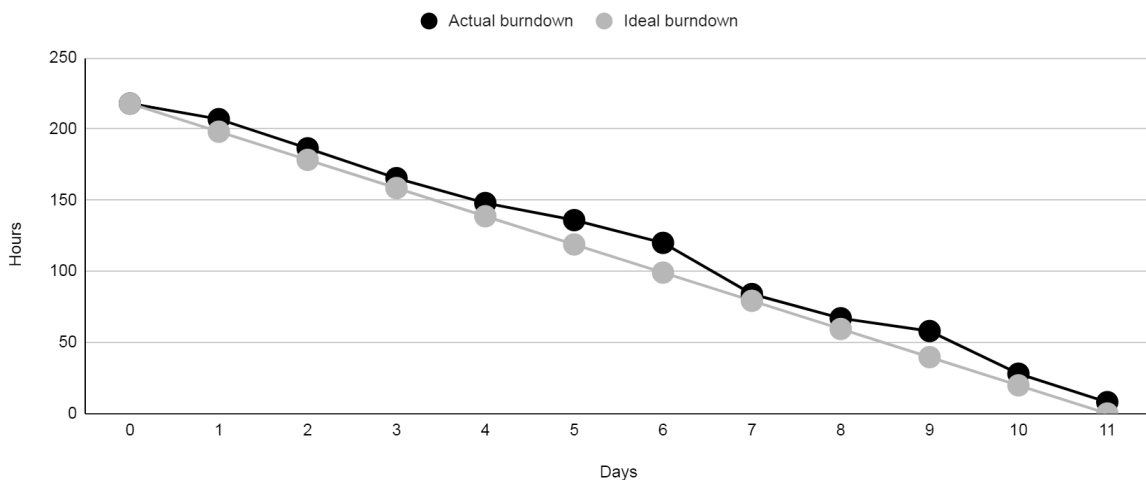
| Time est. total: | 232 | Estimate | Remaining hours on task as of day n | | | | | | | | | | | |
|--------------------------|--------------------|----------|-------------------------------------|----|----|----|----|----|----|----|----|---|----|--|
| | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| Sprint Task | Responsible | | | | | | | | | | | | | |
| Create graphical roadmap | Moha / Gaute | 80 | 80 | 70 | 66 | 60 | 52 | 40 | 32 | 22 | 16 | 3 | | |
| Edit the roadmap | Moha / Gaute | 45 | 45 | 40 | 36 | 34 | 30 | 28 | 26 | 18 | 12 | 3 | | |

| | | | | | | | | | | | | |
|--|-------------------|----|------|------|-----|-----|-----|-----|-----|-----|---|---|
| Add project members / lead / monitor | Alexander | 10 | 10 | 10 | 9 | 8 | 4 | 0 | 0 | 0 | 0 | 0 |
| Assign tasks and subtasks | Gaute | 10 | 9 | 8 | 8 | 7,5 | 7,5 | 7,5 | 5 | 5 | 3 | 0 |
| Create a project with at least one phase | Alexander | 10 | 10 | 5 | 5 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| Add a phase to a project | Alexander | 10 | 10 | 10 | 9 | 9 | 5 | 2 | 2 | 2 | 0 | 0 |
| Create a task | Gaute | 12 | 10,5 | 10,5 | 7 | 4 | 2,5 | 2 | 2 | 2 | 1 | 0 |
| View assigned tasks | Gaute | 13 | 11 | 8 | 7 | 7 | 6,5 | 6 | 5,5 | 5,5 | 4 | 0 |
| Input progress or mark as done (task) | Gaute / Alexander | 10 | 9 | 5 | 4 | 4 | 4 | 4 | 3,5 | 3,5 | 3 | 0 |
| Edit my own tasks | Gaute | 5 | 4,5 | 4,5 | 2 | 1 | 0,5 | 0,5 | 0,5 | 0,5 | 0 | 0 |
| Allocate tasks to a phase | Alexander | 5 | 4,5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 0 |
| Change related task phase | Alexander | 5 | 4,5 | 4,5 | 4,5 | 4,5 | 4,5 | 4 | 4 | 4 | 4 | 4 |
| Create tasks in a project / project tag | Alexander | 4 | 3,5 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 |
| Archive a task | Gaute | 4 | 3,5 | 3 | 3 | 0,5 | 0,5 | 0,5 | 0,5 | 0,5 | 0 | 0 |
| Edit phase-name | Alexander | 4 | 4 | 3 | 3 | 3 | 3 | 1 | 0 | 0 | 0 | 0 |
| Create subtask | Gaute | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 1 | 1 | 0 | 0 |

8.3.3.4 Sprint 4

Burndown Chart

Total hours remaining after each day



| | | | | | | | | | | | | |
|-------------------------|-----|-----|-------|-------|-----|-----|-----|----|----|----|----|---|
| Actual Burndown: | 218 | 207 | 186,5 | 165,5 | 148 | 136 | 120 | 84 | 67 | 58 | 28 | 8 |
| Ideal Burndown: | 218 | 198 | 178 | 159 | 139 | 119 | 99 | 79 | 59 | 40 | 20 | 0 |

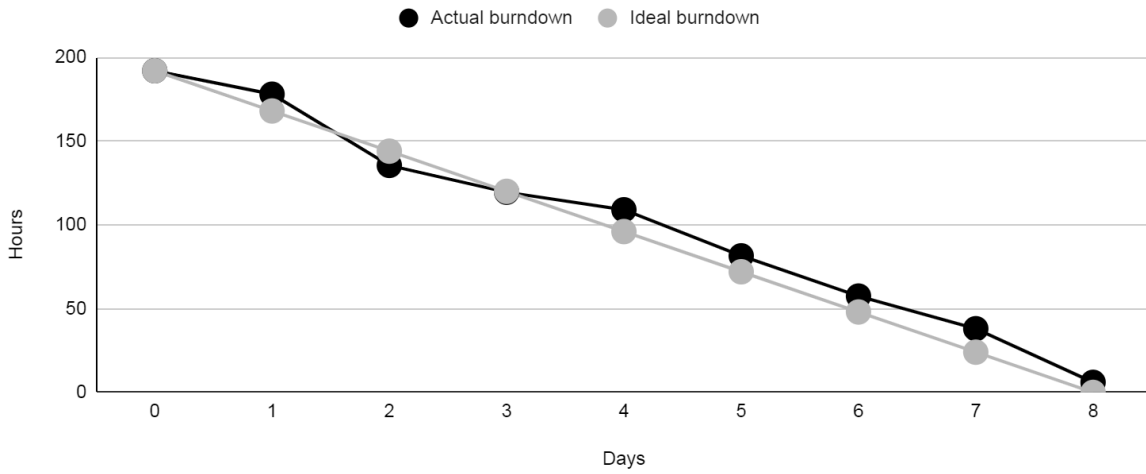
| Time est. total: | 218 | Estimate | Remaining hours on task as of | | | | | | | | | | |
|---|-------------------|----------|-------------------------------|-----|-----|----|----|----|---|---|---|----|----|
| | | | day n | | | | | | | | | | |
| Sprint Task | Responsible | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Project: Error handling & logic rules feedback when creating and editing | Alexander | 13 | 12 | 12 | 10 | 4 | 3 | 1 | 1 | 1 | 1 | 0 | 0 |
| Phase: Error handling & logic rules feedback when creating and editing | Alexander | 9 | 9 | 9 | 9 | 8 | 5 | 2 | 1 | 1 | 1 | 0 | 0 |
| Task: Error handling & logic rules feedback when creating and editing | Gaute / Alexander | 9 | 9 | 9 | 9 | 9 | 9 | 8 | 4 | 4 | 4 | 3 | 1 |
| Subtask: Error handling & logic rules feedback when creating and editing | Gaute / Alexander | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 4 | 4 | 4 | 3 | 1 |
| Fix styling of nodes in roadmap | Mohammad / Gaute | 8 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Convert roadmap to flowchart (series -> parallel -> series) | Mohammad / Gaute | 16 | 14 | 9,5 | 6 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Finalize logic in roadmap and add algorithm to check for loops | Mohammad / Gaute | 16 | 14 | 6 | 2,5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Dropdown for multiple states in tasks | All | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| Optimize tasklist UI (change to component, instant update when state changed, remove buttons the user cannot click) | Alexander | 15 | 15 | 15 | 15 | 14 | 14 | 13 | 2 | 2 | 2 | 0 | 0 |
| Optimize create-task and edit-task UI | All | 10 | 10 | 10 | 3 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 0 |

| | | | | | | | | | | | | | | |
|--|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|---|
| (search-dropdown instead of current menu) | | | | | | | | | | | | | | |
| Finalize archive task, phase & project logic (all subelements are also archived & parent's progress is modified) | Alexander & Gaute | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 25 | 22 | 10 | 0 |
| Add subtasks to archived | Gaute | 10 | 10 | 3 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Refactor roles resolvers for organization and project roles | Gaute / Moha | 20 | 20 | 20 | 20 | 20 | 20 | 18 | 15 | 10 | 8 | 2 | 0 | 0 |
| Add user-authorization as claims in Token (Not project) | Gaute / Moha | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 5 | 0 | 0 | 0 | 0 | 0 |
| Set up refetch-token & automatic refetching of new access- and refresh-token | Mohammad | 15 | 15 | 15 | 11 | 9 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Optimize user-authorization helpers | Moha / Gaute | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 10 | 10 | 8 | 3 | 0 | 0 |
| Add new & optimize current refetches in Project-view | Alexander | 5 | 5 | 5 | 5 | 4 | 3 | 3 | 1 | 1 | 1 | 0 | 0 | 0 |

8.3.3.5 Sprint 5

Burndown Chart

Total hours remaining after each day



| | | | | | | | | | |
|-------------------------|-----|-----|-------|-------|-----|------|------|----|---|
| Actual Burndown: | 192 | 178 | 135,5 | 119,5 | 109 | 81,5 | 57,5 | 38 | 6 |
| Ideal Burndown: | 192 | 168 | 144 | 120 | 96 | 72 | 48 | 24 | 0 |

| Time est. total: | 192 | Estimate | Remaining hours on task as of day n | | | | | | | | |
|--|--------------------|----------|-------------------------------------|-----|-----|------|-----|-----|----|---|---|
| | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Sprint Task | Responsible | | | | | | | | | | |
| Deploy/host Website | Moha | 27 | 23 | 12 | 7 | 3 | 0 | 0 | 0 | 0 | |
| User Testing within Favn / Others | All | 20 | 20 | 19 | 17 | 16,5 | 12 | 12 | 8 | 0 | |
| Catch up with documentation | All | 30 | 25 | 22 | 19 | 16 | 13 | 13 | 11 | 3 | |
| Implement functionality based on feedback | All | 28 | 25 | 14 | 13 | 12 | 7 | 5 | 3 | 1 | |
| Implement design changes based on feedback | All | 20 | 18 | 14 | 12 | 11 | 7 | 5 | 3 | 1 | |
| Make form for user testing | All | 10 | 10 | 10 | 10 | 10 | 8 | 0 | 0 | 0 | |
| Document user testing | All | 5 | 5 | 4,5 | 4,5 | 4,5 | 4,5 | 3,5 | 2 | 0 | |
| Make datasets from user testing data | All | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 2 | 0 | |
| Bug Fixes before first deployment | All | 37 | 37 | 25 | 22 | 21 | 18 | 10 | 4 | 0 | |
| Microservices (?) | All | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 1 | |
| Continuous deployment | Moha | 5 | 5 | 5 | 5 | 5 | 2 | 0 | 0 | 0 | |

8.3.4 Scrum retrospective & review

8.3.4.1 Sprint 1

Review

- What have we done:
 - Frontend basics are set up
 - Apollo client is set up
 - Back-end basics are set up
 - Database is set up
 - Stack is connected
 - Continuous integration with mock database is set up
 - Created basic tests for server
 - Planning documentation such as domain model and sequence diagrams
 - standard frontend components are created, such as landing page and test page
- What was planned but not implemented
 - U06 (navigation between several organization) is not completed

Retrospective

- What worked
 - Good communication among the team members and with supervisors
 - Good collaboration in Git
 - Prioritized well what was to be done, and we got done the most of the planned work
 - Good division of labor, everyone always knew what they were going to do
 - Good planning
 - Good at attendance
 - Learning to use new technologies worked well
- What could have been better
 - Use something else than the Meet.idi.ntnu solution to meet with supervisors
 - Slightly limited sprint, could last longer and thus spend less time on scrum artifacts

8.3.4.2 Sprint 2

Review

- What have we done:
 - Created the functionality of workplace
 - Login functionality using OAuth 2.0
 - Functionality of creating a project
 - Tested the created functionalities
 - Created user system based on OAuth 2.0
- What was planned but not implemented
 - invite users to an organization using email
 - Refresh token for the user logged in

- Not prioritized function when the goal is reaching MVP
- Integrate user functionality with project
 - Assign roles

Retrospective

- What worked
 - Good feedback within the team on functions created
 - Good Attendance
 - Good collaboration on Git branches
 - Maintained the motivation to work despite the workshop at NTNU
- What could have been better
 - Some use issues in Git. We have to make sure that the pushed code is really pushed before making changes on other branches.
 - measures: when branch-merging: thoroughly test that the functionality still works and that the content is correct
 - Somewhat better communication throughout the sprint, especially considering design and prototypes
 - measures: Create and review clear wireframes before development begins
 - Better accessibility between team members.
 - measures: every two hours join discord meeting and take a quick round of progression

8.3.4.3 Sprint 3

Review

- What have we done:
 - Roadmap
 - Visualizes the tasks in one phase
 - Set up dependencies
 - Edit tasks and task states
 - create new tasks
 - Project & Phases
 - project overview
 - Phase overview
 - Phase task list
 - Changing states in underlying tasks and subtasks changes the statistics for phase & project
 - Preview of Roadmap
 - Edit project & edit phase, add phases, add states in a phase
 - Tasks & tasklist
 - A list of tasks user has created
 - A list of tasks user has been assigned
 - Overview of archived tasks

- Subtasks
 - States of a task
 - Archiving of tasks and subtasks
- What was planned but not implemented
 - Fix add dependencies logic in roadmap
 - Change related tasks in phase (Frontend)
 - Create a task in a project (can be done in a phase, team decided that it is not needed in a project)

Retrospective

- What worked
 - Very good collaboration up to sprint review
 - Good demo testing, well planned and no errors
 - Very well done demo with a Favn
 - Communication was good this sprint
 - Breakfast together
 - Good cooperation in the office
- What could have been better
 - Small changes in the same file across different branches led to a lot of unnecessary merging through Git
 - Time from merge-request to merge-completion could have been shorter
 - Smaller tasks in burndown chart
 - The sprint could have been shorter, caused a burnout towards the end

8.3.4.4 Sprint 4

Review

- What have we done:
 - Roadmap
 - Styling of the nodes
 - logic of tasks dependencies
 - Project & Phases
 - Error handling and logic rules feedback both in project, phases, subtasks and tasks
 - logic of archive phases and projects
 - optimize refetch for automatic reload
 - Tasks & task lists
 - Optimize task lists UI
 - optimize create task and edit task UI
 - logic of archive tasks and subtasks
 - workplace
 - Refactor roles in backend

- Users
 - add user authorization claims in token
 - set up refresh token
 - optimize user authorization
- What was planned but not implemented
 - Dropdown for multiple states of phase and tasks

Retrospective

- What worked
 - Very good collaboration up to sprint review
 - Collaboration on Git merges
 - Meetings several times per work day
 - planning
- What could have been better
 - More documented code so others can use the same code for development
 - Kanban board is not updated frequently

8.3.4.5 Sprint 5

Review

- What have we done:
 - deploy and host the website
 - user testing
 - catch up documentation
 - implement functionality and design based on feedback
 - Document user testing and results
 - Bug fixes
 - continuous deployment
- What was planned but not implemented
 - Microservices
 - It was planned to implement this tasks if we get time

Retrospective

- What worked
 - effective implementation of user testing
 - Meetings during the day
 - Demo of the results
 - Use of Kanban board
 - sprint retrospective
- What could have been better

- Documentation one place could be more clear

8.3.5 Notices of meetings

8.3.5.1 Notice 1

Notice of meeting: Bachelor thesis 79 Micro Services

Date and time/room: Thursday 14.01.2021 kl 13:00 – 14.00, room: meet.idi.ntnu.no

The following persons are summoned:

Sveinung Øverland
Anders Hallem Iversen
Gaute Wierød Rønning
Mohammad Al Nayef
Alexander Carlsen
Donn Morrison

Agenda:

Case 01/2021 Introduction
Case 02/2021 Comments on the notice of meeting
Case 03/2021 Any general Questions
Case 04/2021 Comments on the vision document
Case 05/2021 Comments on the product backlog
Case 06/2021 Any possible questions

The meeting is scheduled to end at approx. 14:00. Contact the undersigned if you do not have the opportunity to come.

Mvh
Gaute

Trondheim 13.01.2021

8.3.5.1 Notice 2

Notice of meeting: Bachelor thesis 79 Micro Services

Date and time/room: Thursday 28.01.2021 kl 13:00 – 14.00,

Place: meet.idi.ntnu.no

The following persons are summoned:

Sveinung Øverland

Anders Hallem Iversen

Gaute Wierød Rønning

Mohammad Al Nayef

Alexander Carlsen

Donn Morrison

Agenda:

07/2021 Introduction

08/2021 Sprint 1 review / feedback

09/2021 Planning sprint 2 feedback

10/2021 Other questions from the team

11/2021 Any other questions

The meeting is scheduled to end at approx. 14:00. Contact the undersigned if you do not have the opportunity to come.

Mvh
Gaute

Trondheim 20.01.2021

8.3.5.1 Notice 3

Notice of meeting: Bachelor thesis 79 Micro Services

Date and time/room: Thursday 18.02.2021 kl 13:00 – 14.00, room: Google meet

The following persons are summoned:

Sveinung Øverland
Anders Hallem Iversen
Gaute Wierød Rønning
Mohammad Al Nayef
Alexander Carlsen
Donn Morrison

Agenda:

12/2021 Introduction
13/2021 Sprint 2 review + feedback
14/2021 Planning sprint 3
15/2021 Other questions from the team
16/2021 Any other questions

The meeting is scheduled to end at approx. 14:00. Contact the undersigned if you do not have the opportunity to come.

Mvh
Gaute

Trondheim 18.02.2021

8.3.5.1 Notice 4

Notice of meeting: Bachelor thesis 79 Micro Services

Date and time/room: Thursday 11.03.2021 kl 13:00 – 14.00, room: Google meet

The following persons are summoned:

Sveinung Øverland
Anders Hallem Iversen
Gaute Wierød Rønning
Mohammad Al Nayef
Alexander Carlsen
Donn Morrison

Agenda:

17/2021 Introduction
18/2021 Sprint 3 demo and feedback
19/2021 Discussion about sprint 4
20/2021 Other questions from the team
21/2021 Any other questions

The meeting is scheduled to end at approximately 14:00. Contact the undersigned if you do not have the opportunity to come.

Mvh
Gaute

Trondheim 10.03.2021

8.3.5.1 Notice 5

Notice of meeting: Bachelor thesis 79 Micro Services

Date and time/room: Thursday 13.04.2021 kl 13:00 – 14.00, room: Google meet

The following persons are summoned:

Sveinung Øverland
Anders Hallem Iversen
Gaute Wierød Rønning
Mohammad Al Nayef
Alexander Carlsen
Donn Morrison

Agenda:

22/2021 Introduction
23/2021 Sprint 4 demo and feedback
24/2021 Discussion about user testing
25/2021 Other questions from the team
26/2021 Any other questions

The meeting is scheduled to end at approximately 14:00. Contact the undersigned if you do not have the opportunity to come.

Mvh
Gaute

Trondheim 07.03.2021

8.3.6 Meeting minutes

8.3.6.1 Minute 1

Minutes of meeting bachelor thesis (Taskflow) 79

Date and time: 14.01.21 kl 13:00-14:00

Place: meet.idi.ntnu.no

Present at meeting:

Sveinung Øverland (Favn)
Anders Hallem Iversen (Favn)

Gaute Wierød Rønning
Mohammad Al Nayef
Alexander Carlsen
Donn Morrison (Supervisor)

Absent: None

Chair: Alexander

Case 1/2021.

Introduction of team members, supervisor and product owner.

Case 2/2021.

Meeting notice looks good, we will keep the same format for future meetings.

Case 3/2021.

Answers to some general questions the team members had about documentation templates or scrum artefacts. A burn-up chart that shows progress over all sprints is not required by the product owner. The vision document is to be submitted for review by the supervisor when the first draft is done. User needs must describe overall problems that must be solved by the system and not functional features or detailed situations like user stories.

Case 4/2021.s

The product owner comments on the vision document, what has been done so far looks good.

Case 5/2021.

The product owner comments on the product backlog, overall, it looks good. They will read through the backlog more closely later and give some kind of feedback either via mail or slack.

Case 6/2021.

No further questions asked.

14.01.2021, Gaute

8.3.6.1 Minute 2

Minutes of meeting bachelor thesis (Taskflow) 79

Date and time: 28.01.21 kl 13:00-14:00

Place: meet.idi.ntnu.no

Present at meeting:

Sveinung Øverland (Favn)
Anders Hallem Iversen (Favn)
Gaute Wierød Rønning
Mohammad Al Nayef
Alexander Carlsen
Donn Morrison (Supervisor)

Absent: None

Chair: Alexander

Case 7/2021.

Introduction

Case 8/2021.

Sprint 1 review

- Comment (Favn): domain model looks good and contain the most important parts

Case 9/2021.

Planning sprint 2 / feedback

priority list (Favn)

- Log in function
- Gantt-diagram
- Create an organization function
- Invite member (If we have time)

Case 10/2021.

Questions from the team:

- Should we create users using Google login and email/password?
 - (Anders): It is enough with Google login for now.
- Do you have any tips for the use of oAuth
 - (Donn) recommended us to read well through oAuth2 because there are several ways to make mistakes.

Case 11/2021.

Any other questions:

- (Anders) wants us to send the domain model and the Gantt chart when we finish them

Tips:

(Sveinung) Tips about microservices:

1. do not create too many services
2. testing locally can be difficult if we have too many services

No further questions asked.

28.01.2021, Mohammad

8.3.6.1 Minute 3

Minutes of meeting bachelor thesis (Taskflow) 79

Date and time: 18.02.21 kl 13:00-14:00

Place: Google Meet

Present at meeting:

Sveinung Øverland (Favn)
Anders Hallem Iversen (Favn)
Gaute Wierød Rønning
Mohammad Al Nayef
Alexander Carlsen

Absent: Donn Morrison (Could not come, because he needs to focus on a research paper)

Chair: Alexander

Case 12/2021.

Introduction

Case 13/2021.

Sprint 2 review / feedback

- Comment (Favn): Google login is implemented well but it needs more testing.
- members in an organization should not have access settings page, admin and owner of an organization do not need to be members.

Case 14/2021.

Planning sprint 3 / feedback

The goal for this sprint is to get tasks and projects connected together, and show them in a graphical roadmap.

Case 15/2021.

Questions from the team:

- Is it important to get the user functionality implemented perfectly now, for example implementing refresh token?
- (Favn): It is not necessary now, the main focus should be the roadmap.

Case 16/2021.

Any other questions:

- (Anders) Is it possible to host the website soon?
 - (Team members) We prefer to wait until next sprint, so we have more functionable website

No further questions asked.

18.02.2021, Mohammad

8.3.6.1 Minute 4

Minutes of meeting bachelor thesis (Taskflow) 79

Date and time: 11.03.21 kl 13:00-14:00

Place: Google Meet

Present at meeting:

Sveinung Øverland (Favn)
Anders Hallem Iversen (Favn)
Gaute Wierød Rønning
Mohammad Al Nayef
Alexander Carlsen

Absent: Donn Morrison (Could not come, he had a conference, and he suggested to have meeting without him if we don't need input from him)

Chair: Alexander

Case 17/2021.

Introduction

Case 18/2021.

Sprint 3 review / feedback

- Comments (Favn):
 - The roadmap should have more flexible logic, so the users can add some parallel tasks.
 - All tasks do not need to have deadline date

Case 19/2021.

Planning sprint 4 / feedback

The goal for sprint 4 is to fix the UX/UI for the website and make roadmap more functionable, and fix refresh tokens.

Case 20/2021.

Questions from the team:

- Is it preferable to start testing before UX/UI is finished?
 - (Favn): we can wait until the design and user feedback are implemented

Case 21/2021.

Any other questions:

- (Sveinung) Is it possible to add repeating tasks (tasks which happen in a period)?
 - (Team members): We do not have this function today, but we can add it in sprint 4
- (Anders) Can the team members add a roadmap for phases in a project?
 - (Team members): We think the idea is a good idea, but we are not sure if we have time for this. We can check this when we plan sprint 4

No further questions asked.

12.03.2021, Mohammad

8.3.6.1 Minute 5

Minutes of meeting bachelor thesis (Taskflow) 79

Date and time: 13.04.21 kl 13:00-14:00

Place: Google Meet

Present at meeting:

Sveinung Øverland (Favn)
Anders Hallem Iversen (Favn)
Gaute Wierød Rønning
Mohammad Al Nayef
Alexander Carlsen
Donn Morrison

Chair: Alexander

Case 22/2021.

Introduction

Case 23/2021.

Sprint 4 review / feedback

- Comments (Donn):
 - Generally happy with design and functionality.

- Some bugs need to be fixed, some 422 errors in demo. Most likely refetches failing.
- Happy with the security measures in place, both in server and in client.

Case 24/2021.

Planning User testing

- Next work period will be a mix between implementing feedback from user testing, running user testing and updating the documentation
- Briefly starting to document the big parts and attachments of the main report.
- The work period will be designated as spring 5, however it will not be a feature based sprint.

Case 25/2021.

Questions from the team:

- (Team members): “How is our current main issue related to the context of the thesis?”
- (Donn): “I think that the main issue describes the thesis well.”

Case 26/2021.

Any other questions:

- (Donn): “How is the team planning to do user testing?”
- (Team members): We plan to firstly test internally in FavN AS. After this we will try to implement some changes based on this data. Then we plan to test with some bigger contacts who seemed interested to test our product. All the test data will be compiled to usable data and reflected over in the main report.

No further questions asked.

13.04.2021, Gaute

8.3.7 Time sheets w/ daily status

8.3.7.1 Alexander:

| Week | Date | Hours | Category | Tasks |
|------|------------|-------|----------|---|
| | 2021-01-13 | 7,5 | Planning | Started making product backlog, spring backlog templates and vision the document |
| | 2021-01-14 | 8 | Planning | Start-up meeting w/Donn & FavN, worked further on product backlog and vision document |

| | | | | |
|------------|------------|---------|--|--|
| | 2021-01-15 | 9 | Backend, DB, Frontend | Git-repo, backend skeleton, frontend skeleton, connecting backend to DB, basic frontend components |
| | 2021-01-16 | | | |
| | 2021-01-17 | | | |
| | 2021-01-18 | | | |
| | 2021-01-19 | | | |
| 3 | 2021-01-20 | 8 | Frontend, Backend | Connecting frontend to server through Apollo, server configuration, testing queries from frontend to db |
| | 2021-01-21 | 7,5 | Front End, Planning | Enabling mutations from frontend to server, domain models, displaying test data in frontend |
| | 2021-01-22 | 8,5 | Frontend, Backend | |
| | 2021-01-23 | | | |
| | 2021-01-24 | | | |
| | 2021-01-25 | | | |
| | 2021-01-26 | 2 | NTNU Lecture | Lecture in science theory with Kristin, 2 hours on Zoom |
| 4 | 2021-01-27 | 6 | NTNU Lecture | Lecture in science theory with Kristin, 4 hours on Zoom. Sidebar in frontend |
| | 2021-01-28 | 4 | Backend | Update backend to work with new gql types, fixed test-task update resolvers |
| | 2021-01-29 | 5 | Planning, Meeting, Backend, Documentation | Sprint 2 planning, Meeting with Favn & NTNU, Made Gant diagram/roadmap & started on Oauth-2 |
| | 2021-01-30 | 8 | Backend, Frontend | Implementing O-auth2 in frontend of service 1, attempting to create service 2 as an authentication service |
| | 2021-01-31 | 8 | Backend | Service 2, user authentication & interservice communication |
| | 2021-02-01 | | | |
| | 2021-02-02 | | | |
| | 2021-02-03 | 7,5 | Backend | O-auth2 authentication & login, validation of google access-tokens server side |
| 2021-02-04 | 8 | Backend | O-auth2: Login, refresh tokens and token renewal | |
| | 2021-02-05 | | | |
| | 2021-02-06 | | | |
| | 2021-02-07 | 2 | Frontend | General UI components |

| | | | | |
|---|------------|-----|--|--|
| | 2021-02-08 | | | |
| | 2021-02-09 | 2 | Backend | Project endpoints |
| 6 | 2021-02-10 | 6 | Backend, frontend | Project endpoints, project list UI |
| | 2021-02-11 | 7 | Frontend, backend, testing | Project view, endpoints, testing of project endpoints |
| | 2021-02-12 | 6,5 | Frontend, documentation | Project view, general UI components, documentation & design |
| | 2021-02-13 | 2 | NTNU Lecture | "Vårt BA-project", main issue and vision/scope presented |
| | 2021-02-14 | | | |
| | 2021-02-15 | 8 | NTNU BA-Workshop | Workshop kickoff, meeting with students from BIO-ING, writing work-agreement |
| | 2021-02-16 | 8 | NTNU BA-Workshop | Product owners presentation, working with our product idea and main issue |
| | 2021-02-17 | 8 | NTNU BA-Workshop / Frontend / Docs / Backend | Our product idea presentation, watching other groups ideas. |
| | 2021-02-18 | 7 | Frontend | |
| 7 | 2021-02-19 | 5 | Backend | Endpoints for phases & projects |
| | 2021-02-20 | 8 | Backend, Testing | Worked on endpoints & tests for projects, phases |
| | 2021-02-21 | | | |
| | 2021-02-22 | | | |
| | 2021-02-23 | | | |
| | 2021-02-24 | 7 | Backend, Frontend | Project |
| | 2021-02-25 | 7 | Backend, Frontend | Edit project |
| 8 | 2021-02-26 | 5 | Frontend, Documentation | UI for project |
| | 2021-02-27 | | | |
| | 2021-02-28 | 4 | Backend, Frontend | UI & Endpoints for phases & edit projects |
| | 2021-03-01 | 2 | Frontend, Backend | Edit phase |

| | | | | |
|------------|------------|------------|----------------------|---|
| | 2021-03-02 | 2,5 | Backend, Frontend | Allocate project members, lead & monitor |
| 9 | 2021-03-03 | 7 | Backend, Frontend | Allocate project members, lead & monitor. Made dropdown-menus for choosing project lead, addUser UI for projects, remove user & uselist |
| | 2021-03-04 | 3 | Frontend | |
| | 2021-03-05 | | | |
| | 2021-03-06 | 8 | Frontend | Improved phase-edit, project overview, project creation and general components in frontend |
| | 2021-03-07 | 8 | Frontend, Backend | UI-optimization |
| | 2021-03-08 | | | |
| | 2021-03-09 | 3 | Frontend | |
| | 10 | 2021-03-10 | 8 | Frontend, Backend |
| 2021-03-11 | | 9 | Frontend, Backend | Post-merge bug-fixing on TaskList in phaseOverview and editSubtasks, improved on how progress & weight is updated in parent task, phase & project |
| 2021-03-12 | | | | |
| 2021-03-13 | | | | |
| 2021-03-14 | | | | |
| 2021-03-15 | | 2 | Frontend | Refactored frontend design, tested alternative darkmode designs |
| 2021-03-16 | | 5 | Frontend | Made all colors in frontend based on css-variables |
| 11 | | 2021-03-17 | 7 | Frontend |
| | 2021-03-18 | | | |
| | 2021-03-19 | | | |
| | 2021-03-20 | | | |
| | 2021-03-21 | | | |
| | 2021-03-22 | | | |
| | 2021-03-23 | | | |
| 12 | 2021-03-24 | | | |
| | 2021-03-25 | | | |
| | 2021-03-26 | | | |

| | | | | |
|----|------------|------------|----------------------------------|--|
| | 2021-03-27 | | | |
| | 2021-03-28 | 7 | Frontend, Backend, Documentation | Added input & logic-rules for project-creation in frontend & backend, UI indicators, and a list of all planned input-rules |
| | 2021-03-29 | 7 | Frontend, Backend | Added input-rules and UI-indicators for these for EditProject, EditPhase, and AddPhase. Performed minor UI-fixes in the afore-mentioned components |
| | 2021-03-30 | 8,5 | Frontend | Created a general error-message component, implemented it in all forms with input-rules |
| 13 | 2021-03-31 | 9 | Frontend | Redesigned CreateTask, made it inline & less space-consuming. Created several custom dropdowns for CreateTask & CreateSubtask component |
| | 2021-04-01 | 8 | Frontend | Refactored tasklist, made task & subtask into components. Redesigned subtasks |
| | 2021-04-02 | | | |
| | 2021-04-03 | 7,5 | Frontend, Backend | Added keyboard-shortcuts to CreateTask & CreateSubtask. Refactored how task-states are updated & eliminated bugs with negative progress. |
| | 2021-04-04 | | | |
| | 2021-04-05 | 8 | Frontend, Backend | Refactored Tasklist & its endpoints, added filtering of tasks. Implemented tasklist in PhaseOverview. Made input-rules in CreateTask/CreateSubtask |
| | 2021-04-06 | 2 | Front End, Planning | Minor UI fixes, planned future minor UI- & functionality improvements |
| | 2021-04-07 | 8,5 | Frontend, Backend | Refactored EditTask, Archive subtask/task/phase/project removes weight & progress from parents. Updated assignedTasks UI. |
| 14 | 2021-04-08 | 2 | Frontend | Fixed refetches in Project/PhaseOverview |
| | 2021-04-09 | 6,5 | Frontend | Added EditSubtask, fixed bugs in EditTask/EditSubtask regarding assignments, duration & refetches |
| | 2021-04-10 | | | |
| | 2021-04-11 | | | |
| | 2021-04-12 | 8 | Frontend | Refactored Workspace-Settings, implemented new design, made a new member list that merged admin- and member lists. Added dropdown to modify member-authorization |
| | 2021-04-13 | 7,5 | Frontend, Meeting | Meeting with Favn. Redesigned TaskList, added date-display & filtering based on if tasks are completed & hide/show for subtasks. Restructured CreateTask/CreateSubtask/EditTask-components |
| | 15 | 2021-04-14 | 8,5 | Frontend |

| | | | | |
|----|------------|-----|-------------------------|--|
| | | | | weight & progress when subtask is edited. Restructured PhaseOverview |
| | 2021-04-15 | 10 | Backend, Frontend | Fixed progress & weight being updated when Tasks & Subtasks are archived/edited. Bug-fixing regarding adding users to workspaces & refetching in project/phases. Added limitations for date-input in Tasks/Subtasks & Phase |
| | 2021-04-16 | | | |
| | 2021-04-17 | | | |
| | 2021-04-18 | 3 | Frontend | Bug-fixing & restructuring of refetches upon task/subtask creation, editing & archiving |
| | 2021-04-19 | | | |
| | 2021-04-20 | | | |
| | 2021-04-21 | 8 | Frontend | Redesigned Create- & Edit-Task components in roadmap. Restructured EditRoadmap-page. Improved date-limits on tasks. Attempted to make a kanban-board, but it was scrapped since the team deemed it too time-consuming. |
| | 2021-04-22 | 8 | Frontend | General polish of UI components. Fixed z-index issues on dropdowns, improved UI & search in assignee selector |
| | 2021-04-23 | 8 | Frontend | Renamed links & navbar-elements, renamed buttons, restructured tasklist & phaseOverview |
| | 2021-04-24 | | | |
| | 2021-04-25 | | | |
| | 2021-04-26 | 8 | Frontend | Polishing & Quality of life improvements |
| 16 | 2021-04-27 | 7,5 | Backend, Frontend | Worked on filtering of tasks in tasklist, as well as implementing multiple assignees to a task |
| | 2021-04-28 | 7 | Frontend, Documentation | Polished UI of roadmap-page, added explanations of the roadmap-guide & redesigned it. Worked on sequence-diagrams |
| | 2021-04-29 | | | |
| | 2021-04-30 | | | |
| | 2021-05-01 | 11 | Frontend, Backend | Worked on filtering of assigned-tasks & created-tasks. Added date-limitations to missing inputs. Creating a subtask for the first time now resets its parent's progress & weight. Removed unused endpoints. |
| | 2021-05-02 | 3 | Backend | Migrated project hosting to Google Cloud's App Engine. |
| | 2021-05-03 | 13 | Backend, Frontend | Improved & fixed filtering for my-tasks & created-tasks, tried to display subtasks based on their deadline-date, not their parents. Failed because of Apollo cache not merging data correctly, and the reverted to- and improved upon the old tasklist |

| | | | | |
|-------------|------------|-------|----------------------------------|---|
| | 2021-05-04 | 6 | Testing, Documentation, Frontend | Fixed bug with AddMember to workspace, performed usertest with Equinor & Tryg forsikring and updated the domain-model |
| 18 | 2021-05-05 | 2 | Documentation | Updated domain-models |
| | 2021-05-06 | 2 | Documentation | Worked on personal time-list |
| | 2021-05-07 | 4 | Documentation | Wrote on the bachelor report (Overview) |
| | 2021-05-08 | 5 | Documentation | Wrote on the bachelor report (Theory) |
| | 2021-05-09 | 6 | Documentation | Planned experiment |
| | 2021-05-10 | 5 | Testing | Performed experiments and data gathering |
| | 2021-05-11 | 6 | Testing | Performed experiments and data gathering |
| 19 | 2021-05-12 | 6 | Testing, Documentation | Performed experiments and data gathering, as well as statistical analysis of the results |
| | 2021-05-13 | 6 | Documentation | Wrote on the bachelor report (Methodology) |
| | 2021-05-14 | 6 | Documentation | Wrote on the bachelor report (Methodology) |
| | 2021-05-15 | 6 | Documentation | Wrote on the bachelor report (Results) |
| | 2021-05-16 | 5 | Documentation | Wrote on the bachelor report (Results) |
| | 2021-05-17 | 6 | Documentation | Wrote on the bachelor report (Discussion) |
| | 2021-05-18 | 6 | Documentation | Wrote on the bachelor report (Discussion & Conclusion) |
| 20 | 2021-05-19 | 6 | Documentation | Wrote on the bachelor report (Conclusion, Further work, Introduction, Design decisions) |
| | 2021-05-20 | 6 | Documentation | Proof-reading of the bachelor report |
| Total hours | | 527,5 | | |

8.3.7.2 Mohammad:

| Week | Date | Hours | Category | Task |
|------|------------|-------|---------------------------|---|
| | 2021-01-13 | 8 | planning | Product backlog, Project Vision |
| | 2021-01-14 | 8 | planning | Meeting with Favn and Donn, Product backlog, user stories and System requirements |
| | 2021-01-15 | 8 | Documentation/ Backend | Contact, Server setup |
| | 2021-01-16 | | | |
| | 2021-01-17 | | | |
| | 2021-01-18 | | | |

| | | | | |
|---|------------|-----|--------------------|--|
| | 2021-01-19 | | | |
| 3 | 2021-01-20 | 8 | Fullstack | Debugging some methods in resolver, Connect front end with back end using Apollo |
| | 2021-01-21 | 6,5 | Testing | Backend testing, some docker testing and small fixes in backend structure |
| | 2021-01-22 | 8 | Testing, planning | Sat up the CI pipeline and created some tests, go through requirements documentation |
| | 2021-01-23 | 6 | Testing | Created tests to backend |
| | 2021-01-24 | | | |
| | 2021-01-25 | | | |
| | 2021-01-26 | 3 | NTNU Lecture | Lecture in science theory with Kristin, 2 hours on Zoom, saw the 1 hour long video needed for the next lecture |
| 4 | 2021-01-27 | 4 | NTNU Lecture | Lecture in science theory with Kristin, 4 hours on Zoom |
| | 2021-01-28 | 6 | Testing, planning | Planned the sprint 2 and it's documentation, some debugging for the entire stack |
| | 2021-01-29 | 7 | User, planning | Meeting with Favn and Donn, create user oAuth, sprint backlog |
| | 2021-01-30 | 7,5 | Backend, frontend | created google oAuth front end for login and send data to the server, organization in server and some debugging |
| | 2021-01-31 | 4 | Backend | Micro services communication |
| | 2021-02-01 | | | |
| | 2021-02-02 | | | |
| 5 | 2021-02-03 | 7,5 | Backend | User authentication and JWT token |
| | 2021-02-04 | 8 | Backend | Server middleware, and setting cookies |
| | 2021-02-05 | 7,5 | Backend | Login function |
| | 2021-02-06 | | | |
| | 2021-02-07 | | | |
| | 2021-02-08 | | | |
| | 2021-02-09 | 1 | documentation | Presentation of the main issue |
| 6 | 2021-02-10 | 9 | Backend / frontend | Fix token expiration, and parse token. test user in context and fix access to organization resolvers, meeting about main issue |
| | 2021-02-11 | 5,5 | Frontend | Fix some user friendly functions in frontend, beside merging user branch in main and some cookie functionality |
| | 2021-02-12 | 7 | Frontend | Fixed a bug in login render, added log out, added org context to cookie and read about Gomail package |
| | 2021-02-13 | 2 | NTNU Lecture | "Vårt BA-project", main issue and vision/scope presented |
| | 2021-02-14 | | | |
| | 2021-02-15 | 8 | NTNU BA-Workshop | Workshop kickoff, meeting with students from BIO-ING, writing work-agreement |
| | 2021-02-16 | 8 | NTNU BA-Workshop | Product owners presentation, working with our product idea and main issue |

| | | | | |
|----|------------|---|-----------------------------|--|
| 7 | 2021-02-17 | 6 | NTNU BA-Workshop | Our product idea presentation, watching other groups ideas. Testing before demo. |
| | 2021-02-18 | 7 | Planning/meeting | Planning sprint 3, meeting with Favn and some styling for demo |
| | 2021-02-19 | | | |
| | 2021-02-20 | 8 | Fullstack | worked with React flow library to create the roadmap in front end based on data from backend |
| | 2021-02-21 | | | |
| | 2021-02-22 | | | |
| | 2021-02-23 | | | |
| 8 | 2021-02-24 | 8 | Fullstack | Added possibility to edit dependencies in tasks, show sorted tasks in roadmap |
| | 2021-02-25 | 8 | Frontend | Show task details, and sort tasks in roadmap |
| | 2021-02-26 | 8 | Frontend/testing | Connecting Roadmap and tasks together, add edit task in roadmap and test it |
| | 2021-02-27 | | | |
| | 2021-02-28 | | | |
| | 2021-03-01 | | | |
| | 2021-03-02 | | | |
| 9 | 2021-03-03 | 7 | Frontend | Roadmap, update the roadmap when tasks updated, fixed some bugs |
| | 2021-03-04 | 8 | fullstack | rules for adding edges in roadmap, added show tasks in phase |
| | 2021-03-05 | 6 | fullstack | remove edges + next tasks and fix tasks dependencies |
| | 2021-03-06 | 8 | frontend | added roadmap to phase page, dynamic variables in roadmap, show phases styling and testing |
| | 2021-03-07 | | | |
| | 2021-03-08 | | | |
| | 2021-03-09 | | | |
| 10 | 2021-03-10 | 8 | Frontend | navigation to roadmap from phase page, added add task to roadmap and fixed a bug with remove task dependencies |
| | 2021-03-11 | 8 | Planning, testing, frontend | Do some demo tests, meeting with Favn, working on roadmap dependencies logic |
| | 2021-03-12 | 8 | Planning, frontend | Planning sprint 4, documentation, tasks check (before sett done) in roadmap, research on algorithm for adding dependencies |
| | 2021-03-13 | | | |
| | 2021-03-14 | | | |
| | 2021-03-15 | | | |
| | 2021-03-16 | | | |
| 11 | 2021-03-17 | | | |

| | | | | |
|----|------------|---|--------------------|---|
| | 2021-03-18 | | | |
| | 2021-03-19 | 6 | Frontend / backend | Convert roadmap to flowchart (serie -> parallel -> serie), add endpoint to backend to fetch tasks in phase |
| | 2021-03-20 | | | |
| | 2021-03-21 | | | |
| | 2021-03-22 | | | |
| | 2021-03-23 | | | |
| 12 | 2021-03-24 | 8 | Frontend | Fixed adding dependencies in roadmap (Added BFS algorithm) fixed som styling in roadmap |
| | 2021-03-25 | 8 | frontend | Fixed mark phase as done, added some error handling to roadmap, finalize logic in roadmap, started reading about refresh token |
| | 2021-03-26 | 6 | backend/frontend | UI/UX in frontend, and some feedback for the user from backend when something is wrong |
| | 2021-03-27 | 8 | backend/frontend | UI/UX in front end |
| | 2021-03-28 | | | |
| | 2021-03-29 | | | |
| | 2021-03-30 | | | |
| 13 | 2021-03-31 | 7 | Fullstack | Tested the flow of user authentication and changed the token functionality to fit the refresh token function, fixed some styling in roadmap |
| | 2021-04-01 | 8 | Fullstack | added refresh token endpoint in backend, fixed the refresh token functionality |
| | 2021-04-02 | 5 | Backend | added claims to token in backend, and fixed the context in backend so it contains orgId |
| | 2021-04-03 | 8 | Frontend | Fixed claims in token to be used in frontend, fixed refreshing tokens too many times |
| | 2021-04-04 | 4 | Fullstack | setting claims in frontend and using these claims |
| | 2021-04-05 | 7 | fullstack | fixed user roles in frontend, added logic methods in roadmap, fixed some styling and a backend bug in create task, |
| | 2021-04-06 | | | |
| 14 | 2021-04-07 | 8 | fullstack | User roles both frontend and backend, demo testing and bug fixing |
| | 2021-04-08 | 6 | frontend | Helped with logical errors and popups in frontend, read about hosting and deployment |
| | 2021-04-09 | 7 | fullstack | bug fixing, and testing. tried to deploy the server but it does not work for now |
| | 2021-04-10 | | | |
| | 2021-04-11 | | | |
| | 2021-04-12 | | | |

| | | | | |
|----|------------|---|-----------------------|---|
| | 2021-04-13 | | | |
| 15 | 2021-04-14 | 8 | Fullstack | bug fixing after testing with colleagues, created docker image for the server, added shortcuts guide to the roadmap |
| | 2021-04-15 | 8 | Fullstack | Bug fixes, user test, deployment(Not finished) |
| | 2021-04-16 | 7 | Fullstack | Hosting, changes in code (set cookies moved to server, requests to refresh token, get claims, and token changes) |
| | 2021-04-17 | 6 | Fullstack | create .env files for frontend and backend and create testing environment |
| | 2021-04-18 | | | |
| | 2021-04-19 | | | |
| | 2021-04-20 | | | |
| 16 | 2021-04-21 | 7 | Testing/Backend | Continuous deployment, finishing up hosting |
| | 2021-04-22 | 7 | Testing/Fix | CI/CD testing, fix bugs, and user testing |
| | 2021-04-23 | 7 | Testing/Fix | Fixed a bug in logout, make changes on .yml files after a request from Favn, |
| | 2021-04-24 | | | |
| | 2021-04-25 | | | |
| | 2021-04-26 | | | |
| | 2021-04-27 | | | |
| 17 | 2021-04-28 | 7 | Fullstack | added setCurrentPhase (backend + UI + local storage), fixed archive phase, bug fixes |
| | 2021-04-29 | | -- | |
| | 2021-04-30 | 8 | Documentation | Go doc, systemdokumentasjon |
| | 2021-05-01 | 7 | Testing | Bug fixes (landing page / show name /UI for login and landing page), user testing |
| | 2021-05-02 | | | |
| | 2021-05-03 | | | |
| | 2021-05-04 | 4 | Testing | Bug fixes (task dependencies "add and remove"), testing |
| 18 | 2021-05-05 | 8 | Testing/Documentation | User testing, write testing documentation, demo testing, small bug fixes. |
| | 2021-05-06 | 7 | Documentation | System documentation, Go doc (Created html files/pages for offline docs) |
| | 2021-05-07 | 8 | Documentation | System documentation (security, server, source documentation and installation) |
| | 2021-05-08 | | | |
| | 2021-05-09 | 6 | Documentation | Main report (planning, OAuth, reading about API) |
| | 2021-05-10 | | | |
| | 2021-05-11 | 5 | Documentation/report | Main report(Search algorithm, APIs) |

| | | | | |
|-------------|------------|-------|---------------------------|---|
| 19 | 2021-05-12 | 8 | Documentation/ rapport | User testing with Tryg, writing in Main report |
| | 2021-05-13 | 6 | Documentation/ report | Main report (choices of technologies), updated Gantt-diagram small fixes on the deployed server |
| | 2021-05-14 | 6 | Documentation/ report | Main report (engineering results) reading what is written and review of work |
| | 2021-05-15 | 6 | Documentation | EER diagram, testing results, review what the others wrote |
| | 2021-05-16 | 8 | Documentation | main report discussion, system documentation class diagram, translate documents written in Norwegian |
| | 2021-05-17 | 6 | Documentation | system documentation project structure, system architecture |
| | 2021-05-18 | 4 | Documentation | Report, user testing report, CI/CD system documentation, documentation of code |
| 20 | 2021-05-19 | 8 | Documentation | Main report product and system design, reading through the report |
| | 2021-05-20 | 6 | Documentation | set up files together, read through all documentation, reflection notes. presentation |
| | | | | |
| | | | | |
| Total hours | | 509,5 | | |

8.3.7.3 Gaute:

| Week | Date | Hours | Category | Task |
|------|------------|-------|--------------|---|
| | 2021-01-13 | 9 | Planning | Vision doc, Product Backlog |
| | 2021-01-14 | 9 | Planning/Doc | Planning of sprint 1, Vision document, Meeting documents, meeting with Favn and supervisor |
| | 2021-01-15 | 9 | Doc/Backend | Planning of contracts and other documents. Made general methods in Go server with graphql, connected to db |
| | 2021-01-16 | | | |
| | 2021-01-17 | | | |
| | 2021-01-18 | | | |

| | | | | |
|---|------------|-----|--------------------------|---|
| | 2021-01-19 | | | |
| 3 | 2021-01-20 | 8 | Backend | Making update method in resolver, connecting the backend with the frontend, reading about testing env for gqlgen |
| | 2021-01-21 | 8 | Backend/Testing | Cleaning up and modifying the methods in db.go and resolver, starting with unit tests and creating of mock db |
| | 2021-01-22 | 8 | Doc/Backend/Testing | Making/reviewing domain model, finish making mock db with some testing, started looking at CI workflow |
| | 2021-01-23 | 7 | Backend/Review | Adding user type to backend, making methods for user, made schema based on domain model, sprint review |
| | 2021-01-24 | | | |
| | 2021-01-25 | | | |
| | 2021-01-26 | 2 | NTNU Lecture | Lecture in science theory with Kristin, 2 hours on Zoom |
| 4 | 2021-01-27 | 4 | NTNU Lecture | Lecture in science theory with Kristin, 4 hours on Zoom |
| | 2021-01-28 | 8 | Frontend/Backend | Stack debugging and demoing the product so far |
| | 2021-01-29 | 7,5 | Planning/Backend | Meeting with favn, starting on 2nd sprint, making organization branch and starting with endpoints and tests |
| | 2021-01-30 | 8 | Backend/Frontend/Testing | Finishing some endpoints, connecting some to frontend, continue with testing locally |
| | 2021-01-31 | 7 | Backend | Debugging service communication, continue adding more specified endpoints for orgs and tests for these |
| | 2021-02-01 | | | |
| | 2021-02-02 | | | |
| 5 | 2021-02-03 | 8 | Backend/Frontend | Finishing endpoints needed for now with tests and resolvers, adding queries to frontend and testing on client |
| | 2021-02-04 | 8 | Backend/Frontend | Working more with organizations, whole stack |
| | 2021-02-05 | 8 | Frontend | Reworking some organization UI, more frontend testing |
| | 2021-02-06 | | | |
| | 2021-02-07 | | | |
| | 2021-02-08 | | | |
| | 2021-02-09 | | | |
| 6 | 2021-02-10 | 8 | Frontend/Backend | Started on Projects backend, helped on the access token, other token functionality, merged user into orgs, problem presentation |
| | 2021-02-11 | 8 | Frontend/Backend | Worked with demo styling, login redirect, backend debugging, user-org testing, had to fix errors made in git |
| | 2021-02-12 | 8 | Frontend/Backend | Finished demo styling, fixed some bugs that appeared in user-org testing, nullUrl fix, fixed tests and resolvers based on new models from merge |
| | 2021-02-13 | 2 | NTNU Lecture | "Vårt BA-project", main issue and vision/scope presented |
| | 2021-02-14 | | | |

| | | | | |
|---|------------|-----|---|--|
| | 2021-02-15 | 8 | NTNU BA-Workshop | Workshop kickoff, meeting with students from BIO-ING, writing work-agreement |
| | 2021-02-16 | 8 | NTNU BA-Workshop | Product owners presentation, working with our product idea and main issue |
| 7 | 2021-02-17 | 5 | NTNU BA-Workshop / Frontend testing | Our product idea presentation, watching other groups ideas. Demo testing on the system before sprint 2 meeting. |
| | 2021-02-18 | 4,5 | Demo (Front/backend) | Demo testing on the system, fixing some general bugs, added some minor details in frontend components and some backend resolvers |
| | 2021-02-19 | 7 | Planning/Doc | Meeting with favn, sprint 2 review and retrospective, demo testing, some logic bug fixing in resolvers, sprint 3 planning |
| | 2021-02-20 | 8 | Backend | Made tests and resolvers for all task endpoints, tested some endpoints in playground, connected assigned and author tasks to frontend pages |
| | 2021-02-21 | 7 | Frontend / Backend / Testing | Fixed tasks cards, worked on tasks resolvers and tests |
| | 2021-02-22 | | | |
| | 2021-02-23 | | | |
| 8 | 2021-02-24 | 7 | Frontend / Backend | Made create task and update tasks, fixed some checks in tasks resolvers |
| | 2021-02-25 | 8 | Frontend / Backend | Worked on archive tasks, edit tasks, assigning tasks, started on subtasks, merged roadmap with new task functionality |
| | 2021-02-26 | 7,5 | Frontend testing / Backend | Demo testing and bug fixing the task functionality, updated and made new task tests, helped on roadmap logic and frontend structure |
| | 2021-02-27 | | | |
| | 2021-02-28 | | | |
| | 2021-03-01 | | | |
| | 2021-03-02 | | | |
| | 2021-03-03 | 7 | Frontend | Worked on the roadmap, creating custom nodes. Subtasks needed to be reworked, started on this today |
| | 2021-03-04 | 7 | Fullstack | Worked on subtasks, made tests and create / update / get resolvers, got subtasks into roadmap |
| | 2021-03-05 | | | |
| | 2021-03-06 | 7 | Fullstack | Helped other team members with bugs and logical errors in backend, worked on subtasks, started working on nested queries and reworking queries |
| | 2021-03-07 | | | |
| | 2021-03-08 | | | |

| | | | | |
|------------|------------|------------|-------------------------|--|
| | 2021-03-09 | 6,5 | Frontend | Worked on UI on myTasks and myAssignments. Nested queries now work, saving the number of queries from client to server and back |
| 10 | 2021-03-10 | 7 | Frontend/Backend | Started with demo testing, finished up the task lists, merging on git |
| | 2021-03-11 | 6,5 | Testing + Demo | Ran demo tests, worked on some bugs, then had a meeting with Favn |
| | 2021-03-12 | 7 | Docs/Planning/ Frontend | Planning Sprint 4, making the scrum artifacts, started working on logic in roadmap |
| | 2021-03-13 | | | |
| | 2021-03-14 | | | |
| | 2021-03-15 | | | |
| | 2021-03-16 | | | |
| | 11 | 2021-03-17 | | |
| 2021-03-18 | | | | |
| 2021-03-19 | | 7 | Frontend | Worked on archiving tasks, adding needed components to the list, worked on a pop-up in roadmap, spring based |
| 2021-03-20 | | | | |
| 2021-03-21 | | | | |
| 2021-03-22 | | | | |
| 2021-03-23 | | | | |
| 12 | 2021-03-24 | 7 | Frontend/MMI/ Backend | Worked on roadmap styling and logic, reworked some pages and changed some forms, added some spring. Single subtasks can be found archived now. |
| | 2021-03-25 | 7 | Frontend/MMI | Worked on feedback in roadmap, fixed styling bug in roadmap, createTask UI and editTask UI, added some more springs |
| | 2021-03-26 | 7 | Frontend | Worked on optimizing the editTask UI and other UI/UX |
| | 2021-03-27 | 7 | Frontend | Worked on UI/UX on frontend, testing new implemented logic in roadmap and tasks |
| | 2021-03-28 | | | |
| | 2021-03-29 | | | |
| | 2021-03-30 | | | |
| 13 | 2021-03-31 | | | |
| | 2021-04-01 | | | |
| | 2021-04-02 | | | |
| | 2021-04-03 | 8 | Frontend/Backend | Worked on claims, refactoring org resolvers and more UI/UX |
| | 2021-04-04 | | | |
| | 2021-04-05 | 8 | Frontend/Backend | Worked on task ready, soonReady and firstTask logic in backend and frontend, more on UX/UI, merge with projects branch, fix styling to match the dark mode |

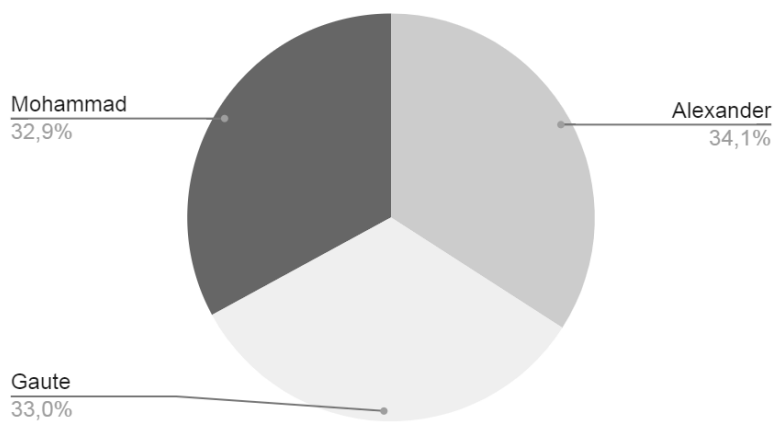
| | | | | |
|----|------------|---|--------------------------------|---|
| | 2021-04-06 | 6 | Frontend/Backend | Made resolvers for archive project and phases, made db methods to resolvers, worked on UX/UI in roadmap, fixed tests |
| 14 | 2021-04-07 | 8 | Frontend/Backend | Worked on minimap for roadmap, and started on keybinding for roadmap |
| | 2021-04-08 | 7 | Frontend | Keybind in roadmap now has right logic, added popup in roadmap |
| | 2021-04-09 | 6 | Frontend | Lots of small fixes in roadmap and on popup |
| | 2021-04-10 | | | |
| | 2021-04-11 | | | |
| | 2021-04-12 | 3 | User testing | Testing main features inc. roadmap, tasklist, project and organizations. Got some useful information in a user test, found some bugs that needs to be fixed |
| | 2021-04-13 | 5 | Meeting/Testing | Meeting with Favn and supervisor. Working with bug fixes and design flaws. |
| 15 | 2021-04-14 | 7 | Frontend | Lots of frontend fixes. Keyboard Shortcuts and other logic fixes in roadmap. Tried some hosting services. |
| | 2021-04-15 | 8 | Frontend/Backend | Fixing bugs before first deployment, helping bug fixing deployment, small UX/UX fixes, user testing with a Favn intern |
| | 2021-04-16 | 7 | Frontend/Backend | Implemented some functionality based on feedback from user testing. Added some keybindings, fixed the endNode quick connection, started working with .env files for the server deployment. More small UI/UX changes |
| | 2021-04-17 | | | |
| | 2021-04-18 | | | |
| | 2021-04-19 | 2 | Documentation | Started working with the end report, sys doc and progress doc |
| | 2021-04-20 | 3 | Documentation | Clean up in notion, all acceptance criteria are done for the main features |
| | 2021-04-21 | 7 | Frontend/Documentation | Added template progress docs, made custom reusable react spring animations, added this to all needed places, fixed a key bindings bug. (Found a bug in reAssigntask, wrong refetch) |
| | 2021-04-22 | 8 | Frontend/Backend/User Testing | Fixed some animation bugs, fixed some other state related bugs, helped debug tests to fix them, merge and deploy a testing version of the product, user testing |
| | 2021-04-23 | 7 | Frontend/Backend/Documentation | Looked at user testing results, started looking at designs and functionality changes that we could realistically implement after Favn testing. Drafting an excel file for user testing results and data. Fixed error feedback in adding members to workplace |
| | 2021-04-24 | | | |
| | 2021-04-25 | | | |
| | 2021-04-26 | 2 | Planning | Meeting with people from Trygg and scheduling a user testing session (may 5th, 14:00). They have agreed to do a short starter demo user test and a longer test involving 2-3 people where they are going to be working project based. Meeting to discuss results will take place may 12th.. |

| | | | | |
|----|------------|-----|--------------------------------|---|
| | 2021-04-27 | 6 | Planning/Frontend/Backend | Some small frontend fixes, adding sorting functions where they are needed, fixes in the backend. Planning user testing with Equinor, meeting with them 5. may kl 11:00. |
| 17 | 2021-04-28 | 7 | Documentation/Frontend/Backend | Writing on the main report and requirements docs. Worked on fixing assigning tasks and getting the right tasks in the users assignments, both tasks and subtasks |
| | 2021-04-29 | | | |
| | 2021-04-30 | 5 | Documentation | Writing on the system documentation, starting on some models that should be added to the system docs. Small bug fixes after merge in tasks |
| | 2021-05-01 | 8 | Backend | Worked on filters in resolvers and db methods for more advanced filtering of the tasks connected to a user |
| | 2021-05-02 | | | |
| | 2021-05-03 | | | |
| | 2021-05-04 | 8,5 | Backend / Frontend | Bug fixes, getting ready to test with Equinor and Tryg. Fixed resolvers and other bugs. |
| 18 | 2021-05-05 | 5 | User testing | Testing with Equinor and Tryg AS. Recap meeting with Tryg AS on 12.05.2021, kl: 14-15 |
| | 2021-05-06 | | | |
| | 2021-05-07 | 3 | Documentation | Writing on the main report. Fetching sources for theory. |
| | 2021-05-08 | | | |
| | 2021-05-09 | 8 | Documentation | Planning and setting the main structure in the main report. Key words define the contents for each chapter. Writing on the theory chapter. |
| | 2021-05-10 | 3 | Documentation | Rewriting and approving / considering changes in theory. Finishing up some sub chapters in the theory chapter. Researching some more on the different relevant technical topics |
| | 2021-05-11 | 7 | Documentation | Main report. Fetching and reading more sources. Preparing a meeting with Favn AS. |
| 19 | 2021-05-12 | 6 | Meeting / Documentation | User testing recap meeting with Favn AS.Organized remaining documentation work in Taskflow, Created division of labor sheets, finished up the theory chapter, revision of other chapters. |
| | 2021-05-13 | 6 | Documentation | Main report, revising and writing on chapter 3. |
| | 2021-05-14 | 5 | Documentation | Main report, revising and writing on chapter 4.3 |
| | 2021-05-15 | 3 | Documentation | Main report, revising chapter 4.2 and 4.1, starting on chapter 5.3, discussing some contents |
| | 2021-05-16 | 6 | Documentation | Main report, rewriting and restructuring chapter 4.3, now chapter 4.4, writing on chapter 5.3 |
| | 2021-05-17 | 3 | Documentation | Main report, revising 5.3, writing on 5.3 |
| | 2021-05-18 | 4,5 | Documentation | Main report, writing on 5.3 |

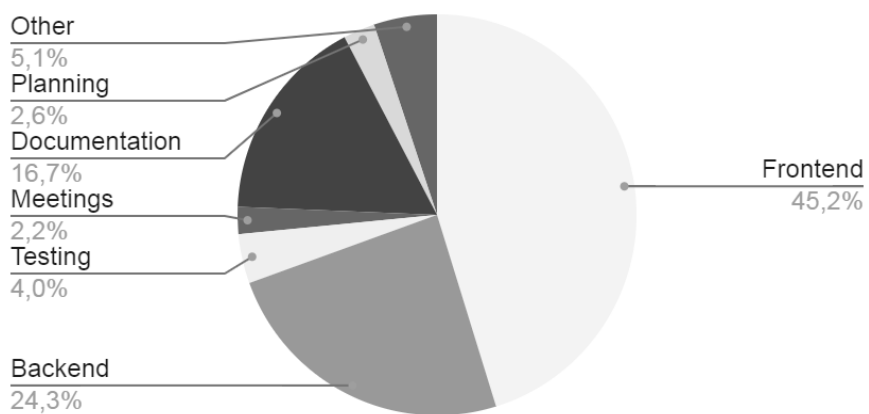
| | | | | |
|-------------|------------|------|---------------|---|
| 20 | 2021-05-19 | 10,5 | Documentation | Main report, revising and writing. Process Docs, writing and structuring. Reviewing other attachments. |
| | 2021-05-20 | 6 | Documentation | Last revising on main report, structuring attachments in main report, finishing up any documentation needed for attachments |
| Total hours | | 510 | | |

8.3.8 Division of labor

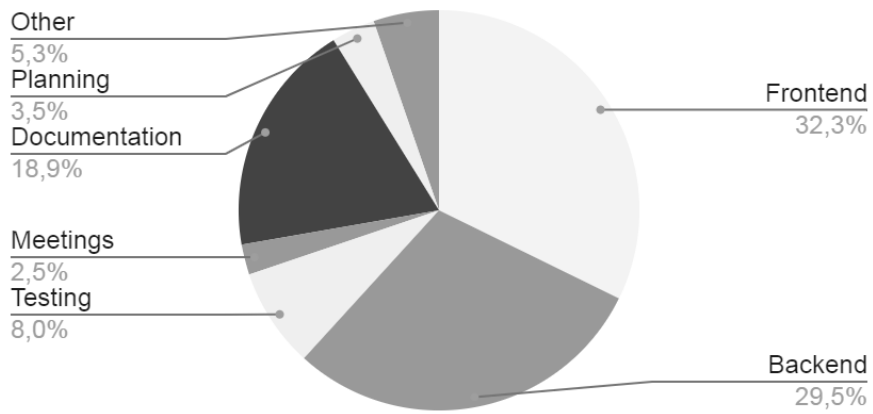
Total Hours Worked Distribution



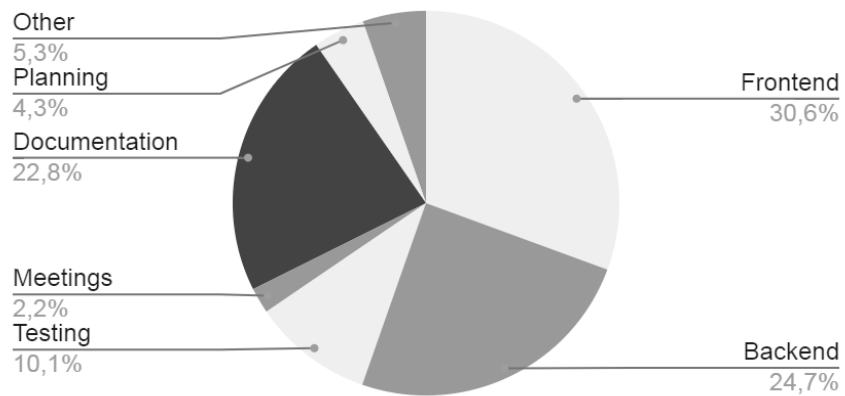
Alexander



Gaute



Mohammad



| Categories | Alexander | Gaute | Mohammad |
|---------------------------|-----------|-------|----------|
| Frontend | 238,5 | 164,5 | 156 |
| Backend | 128 | 150,5 | 126 |
| Testing | 21 | 41 | 51,5 |
| Meetings / Reviews | 11,5 | 12,5 | 11 |
| Documentation | 88 | 96,5 | 116 |
| Planning | 13,5 | 18 | 22 |
| Other (NTNU Related work) | 27 | 27 | 27 |
| Total Hours Worked | 527,5 | 510 | 509,5 |

8.4 Equinor User Testing Report

- Introduction
 - Question:
 - “Does the market exist for this niche product? Is there demand for Taskflow?”
 - This is a system that Favn thinks there definitely is demand for.
- Login:
 - Easy to login, “Very seamless and quick”
 - Launch button needs some work
- Workspace:
 - Changed name and description worked well
 - “Is it possible to change the rights of members here?”
 - The team had to describe the rights of the roles, an explanation could be helpful
 - “Is there any autosave feature here? Could be useful if the user navigates out of workspace settings and dont save the changes”
 - We will take this into consideration for future development.
- Project:
 - Create project worked well
 - Update project worked well
 - The definition of a phase is a little too ambiguous, could need an explanation.
 - LOGIC ERROR: Update project: The user got to make a phase with a start date before the project start date.
 - It is not intuitive to find the phase just created.
 - Create tasks worked well, both with and without keybindings.
- Roadmap:
 - Connection of tasks worked well, intuitive system
 - The color codes needs further explaining
 - How to delete connections of tasks was easy to find and intuitive

- Switch workspace
 - If a user does not have more than 1 workspace this button should be hidden

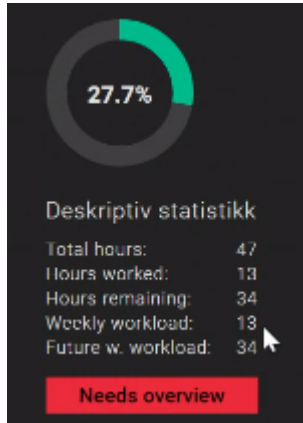
- Created tasks:
 - If there is a possibility to create tasks in this page, this page should not be called created tasks but something else.
 - “What if there are multiple people working on the same task?”
 - The subtasks are made to tackle this problem, however multi assignments in tasks should be implemented in the future.
 - If a task is assigned to a user that task should be locked from editing, by doing this other users cannot edit the task while it's being worked on, ensuring the minimal confusion. There should also be an edit log, to see all recent changes and who made them.
 - Archive task:
 - Archive task worked well
 - “But what of deletion?”
 - At the moment there is only archiving.
 - A bit more visual system could help effectivise the workflow.(Use of drag and drop cards, less text)

- General feedback:
 - The system was intuitive
 - A solid foundation to an easy to use project management tool.
 - There were some main functionalities that Equinor could get use of, where they had no similar solution on our product scale.
 - Drag and drop rather than text.

8.5 Tryg Forsikring User Testing Report

- Introduction
 - The testers has used planning tools in the form of Microsoft planner
 - Never done a agile project before
 - “What is the economical plan for the project”?
 - The team has not been overly invested in the economics of the project.
 - 3 members shall test Taskflow over the period of 1 week.
- Login:
 - No problems with login. However, 1 of the members did not have a google account.
- Workspace:
 - Updating the information worked well
 - Adding members worked well
 - “Is it possible to invite members with their mail”?
 - At this moment no, but it was planned functionality.
 - “Does all the users have the same rights within a workspace”?
 - No, they have different rights.
- prosjekt:
 - Very good UX implementation with the button colors, specifically the green ones.
 - Add states is the first they see in add phases, and it's not intuitive what this is
 - add phases “good”
 - “Is there a phase's roadmap”?
 - No, however this functionality was discussed with Favn
 - add tasks in prosjekt “good”
 - change phases “good”, however a overview of all the phases is missing
 - deadline date (thought to be start date) is good that its limiting the phase start and end based on the project
 - edit task “good”
 - add subtask “good”
 - Task start date can start before project

- When setting your own user as the assignee on tasks, there is no indicating that you have been selected.
- Dropdown overlaps with “hide completed” button
- The status codes of the progress in phases and projects should be customizable, or have some more explaining behind them.



- Phase name in review roadmap needs some css work, a long phase name affects the components structure.
- If a user regrets archiving a phase or project, a undo button or unarchive button is good.
- Roadmap
 - Connection tasks via the mice worked well
 - Connection of tasks via the keybinding worked well, however there is need for some marking feedback when selecting the tasks
 - Roadmap Guide is intuitive and clear
 - “Is the roadmap based on the dates of the tasks?”
 - An explanation of the color codes of the tasks in the roadmap could have been helpful.
 - An undo button for the connection and removing of connections is helpful here
- My tasks:
 - Easy to view the information and intuitive
- General Feedback:
 - Missing overview of what the different resources are allocated to “This could help the project lead of allocation resources between projects and phases, if perhaps one phase or project finished early”

- Overall good UX

8.6 Tryg 2nd Testing Report

Feedback after several days with testing:

- Create a placeholder user, and decide the assignee afterwards when creating a task
- We found out that we need to add some phases without start and end dates, because we often need to plan the project first and what to do, and after that decide the deadlines and dates.
- Duration, if we add Decimal numbers, it fails
- Edit and add subtasks icons are intuitive and easy to spot
- The phase drop down need more obvious design
- Some feedback when a user added to a project or workplace is needed

Experiment:

- “Try to find roadmap in a project”:
 - The user found the roadmap,
 - The minus here is that we can't see the progress of the tasks and if a task has deadline soon
 - Idea: not all subtasks need to be done before doing the next task. Example if a task has 5 subtasks, 3 of these can be marked as “crucial” and when these are done progression on the next task can start.

8.7 Requirements documentation

Version 1.0

Audit History

| Date | Version | Description | Author(s) |
|-------------|----------------|--------------------|------------------------------|
| 14.01.2021 | 1.0 | First Draft | Alexander, Gaute og Mohammad |
| 19.05.2021 | 1.1 | Ready for approval | Gaute og Mohammad |
| | | | |
| | | | |

8.7.1 Introduction

This document's purpose is to address the main problem domains and the system's functionality specifications. First are the system's functionality and in what way they should be operated by a user. Next we describe the main problem domain and present the models tied to this domain problem, in context to Domain Driven Design. Lastly we look at the earliest prototypes of the user interface, also called wireframes. These models set a base for ideas to describe possible functionality and design solutions to the domain problem. The document is connected to the taskFlow thesis main report.

8.7.2 User Stories

These are the main user stories for the product taskFlow. User stories are a way to model a system's functionality in a non-technical and universal language, and to level the information so it's useful and meaningful for all stakeholders.

| ID | User need | As a... | I want to... | So that... | Importance | Sprint | Criteria |
|------|-----------|--------------|---|--|------------|--------|--|
| GR01 | GR | Project Lead | Create a graphical roadmap of tasks with dependencies | Most of the tasks in the projects are hidden until they are relevant and ready to be worked on, and the team has an overview of the entire work process of the project | High | 3 | GR01-01 GR01-02 GR01-03 |
| GR02 | GR | Project Lead | Edit the roadmap | Update the roadmap and the tasks in the project | High | 3 | GR02-01 GR02-02 GR02-03 GR02-04 GR02-05 GR02-06 GR02-07 GR02-08 |
| GR03 | GR | Project Lead | Divide a roadmap into phases and assign tasks to a phase | Easily organize tasks in longer projects into work phases and sprints | Medium | 4 | GR03-01 GR03-03 GR03-02 |
| GR04 | GR | Project Lead | Quickly create and edit the roadmap with keybindings and UI shortcuts | I can more efficiently plan projects and update tasks when changes occur | Low | 4 | GR04-01 GR04-03 GR04-02 GR04-04 |
| GR05 | GR | Project Lead | Create templates from earlier graphical roadmaps and reuse these in future projects | I can easier follow organizational procedures and work methods as well as plan projects faster | Low | 5 | GR05-01 GR05-02 |
| GR06 | GR | Project Lead | Be able to quick-assign tasks in the roadmap by first selecting a member | I can easier assign tasks to project members | Low | 5 | GR06-01 GR06-02 GR06-05 GR06-03 |

| ID | User need | As a... | I want to... | So that... | Importance | Sprint | Criteria |
|------|-----------|---------------|---|--|------------|--------|-------------------------|
| GR07 | GR | Project Lead | Make a task with subtasks into a kanban board by making custom states | Team members and I can easily view the team's progress in the current sprint | Low | 4 | GR07-01 GR07-02 GR07-03 |
| OR01 | OR | User | Create an organization | Administrating and organizing your projects | High | 2 | OR01-01 OR01-02 OR01-03 |
| OR02 | OR | Administrator | Invite new members | Add members and decide their roles | High | 2 | OR02-01 OR02-02 |
| OR03 | OR | Administrator | Change a member's role | I can manage my organizations roles | Medium | 2 | OR03-01 OR03-02 OR03-03 |
| OR04 | OR | Administrator | Remove a member | I can manage my organizations members | Medium | 2 | OR04-01 OR04-02 |
| OR05 | OR | Administrator | Edit organization | I can update my organization | Low | 2 | OR05-01 |
| OR06 | OR | User | Navigate between my organizations | If I work in more than 1 organization I would like to switch | Low | 2 | OR06-01 OR06-02 |
| P02 | P | Project Lead | Add project monitor and project members | I can manage my own projects | High | 3 | P02-01 P02-02 P02-03 |
| P06 | P | Project Lead | Assign tasks and subtasks to project members | I can organize who does what tasks in the project | High | 3 | P06-01 P06-02 |
| P04 | P | Project Lead | Remove project members | I can manage my own projects | Medium | 4 | P04-01 P04-02 |
| P03 | P | Project Lead | Edit project details | I can manage my own projects | Low | 4 | P03-01 P03-02 |
| P05 | P | Project Lead | Designate a project as inactive | Projects that are inactive are not visible | Low | 4 | P05-01 P05-02 |



















| ID | User need | As a... | I want to... | So that... | Importance | Sprint | Criteria |
|-----|-----------|---------|--|---|------------|--------|-----------------------------|
| R01 | R | User | Create general reminder to self | I can easily see what I should check before starting my day | Low | 5 | |
| R02 | R | User | Create a reminder to check on a task while creating the task | I can easily follow up tasks I have assigned to other people | Low | 5 | |
| T01 | T | User | create a task | I can save important information about the task such as: title, description, assigned project members, estimated time and progress. | High | 3 | T01-01 T01-02 T01-03 |
| T02 | T | User | View assigned tasks | I can easily get a overview of which tasks I am assigned | High | 3 | T02-01 T02-02 T02-03 T02-04 |
| T03 | T | User | Input progress or mark as done | I can update my todo-list when a task is done or update its progress when I am finished working on it for the day | High | 3 | T03-01 T03-02 T03-03 T03-04 |
| T04 | T | User | Edit my own tasks | I can update the description or the estimated time of the task, and notify the people who are involved with the new changes | High | 3 | T04-01 T04-02 |
| T05 | T | User | Create tasks in a project or give an existing task a project tag | I can easily distinguish which tasks belongs to which projects | Medium | 3 | T05-01 T05-02 T06-01 T06-02 |
| T07 | T | User | Add subtasks to a task | I can easily see all of the components of larger tasks | Medium | 3 | T07-01 T07-02 |


















| ID | User need | As a... | I want to... | So that... | Importance | Sprint | Criteria |
|------|-----------|--------------|---|---|------------|--------|--|
| T08 | T | User | Hide a task (archived) | I can hide a task that is almost done, but could not be completed | Medium | 3 | T08-01 T08-02 T08-03 T08-04 |
| T06 | T | User | Send a mail to members who are involved in a task | I can easily reach out to the members when I need that. | Low | 5 | |
| U01 | U | User | create a user | I can save my user information in the system and use it to log in. | High | 2 | U01-01 U01-02 U01-03 U01-04 |
| U02 | U | User | log in | I can get access to my profile and the projects which I'm involved in. | High | 2 | U02-01 U02-02 U02-03 U02-04 U02-05 |
| U06 | U | User | Navigate the site | I can find the information I am looking for | High | 1 | U06-01 U06-02 |
| U03 | U | User | edit user profile | I can update my personal information. This includes changing my password. | Medium | 2 | U03-01 U03-02 U03-03 U03-04 |
| U04 | U | User | log out | I can be sure that no one other than me has access to my user | Medium | 2 | U04-01 U04-02 |
| U05 | U | User | Decide to stay logged in | I can save time when accessing Taskflow on my devices | Low | 2 | U05-01 U05-02 |
| PH01 | PH | Project Lead | Create a project with at least one phase | So i can filter between different parts of a projects lifeline | High | 3 | PH01-01 PH01-02 |
| PH02 | PH | Project Lead | Add a phase to project | So i easily know where in my project timeline the team is | High | 3 | PH02-01 PH02-02 |




















COUNT 41




















8.7.3 Acceptance Criteria

















| Aa ID | Description | ☑ Done | 🔗 User-story |
|---------|---|--------|--------------|
| GR01-01 | Can create a graphical flowchart map based on the current tasks in a given phase | ☑ | 📄 GR01 |
| GR01-02 | Only phase tasks is shown in a roadmap, organization tasks should be filtered out | ☑ | 📄 GR01 |
| GR01-03 | Has custom nodes that can show the information of a task in a detailed and ordered way | ☑ | 📄 GR01 |
| GR02-01 | Can edit the links between tasks in the graph | ☑ | 📄 GR02 |
| GR02-02 | Can add tasks directly into the roadmap | ☑ | 📄 GR02 |
| GR02-03 | Can edit tasks while editing the roadmap | ☑ | 📄 GR02 |
| GR02-04 | Can update task states while editing the roadmap | ☑ | 📄 GR02 |
| GR02-05 | Make the graph a flow chart, where connections can go from series to parallel to series | ☑ | 📄 GR02 |
| GR02-06 | Check for loops in the chart using a fitting algorithm | ☑ | 📄 GR02 |
| GR02-07 | Check for logically wrong connections, keeps the structure and connections in the flowchart clean. | ☑ | 📄 GR02 |
| GR02-08 | Database methods used should be tested properly | ☑ | 📄 GR02 |
| GR03-01 | Can add dependencies to the tasks by linking them up together in a node flow tree | ☑ | 📄 GR03 |
| GR03-02 | Have different information in the roadmap based on the current active phase | ☑ | 📄 GR03 |
| GR03-03 | Database methods used should be tested properly | ☑ | 📄 GR03 |
| GR04-01 | Can use keybindings in the roadmap and other button controls so that i can save time when organizing my tasks | ☑ | 📄 GR04 |

| | | | |
|---------|--|-------------------------------------|--|
| GR04-02 | Keybindings should have their own page with proper explanations on how to use them | <input checked="" type="checkbox"/> |  GR04 |
| GR04-03 | Keybindings should be intuitive and should not interfere with other default keybindings | <input checked="" type="checkbox"/> |  GR04 |
| GR04-04 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  GR04 |
| GR05-01 | Choose between set designs to give my roadmap a personal look, or customize my own design by tweeking on the current designs | <input type="checkbox"/> |  GR05 |
| GR05-02 | Save my current design on the roadmap so i can reuse it later | <input type="checkbox"/> |  GR05 |
| GR06-01 | Can organize and lead my project easier and faster by assigning 1 or more tasks straight from the roadmap | <input checked="" type="checkbox"/> |  GR06 |
| GR06-02 | Intuitive and simple design, should be easy to use | <input checked="" type="checkbox"/> |  GR06 |
| GR06-03 | Can only assign people within a given project | <input checked="" type="checkbox"/> |  GR06 |
| GR06-04 | Only project lead or organization admin can use the quick assign function | <input checked="" type="checkbox"/> |  GR06 |
| GR06-05 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  GR06 |
| GR07-01 | Can edit the tasks and information within the kanban board | <input type="checkbox"/> |  GR07 |
| GR07-02 | Kanban board updates when a update happens either in your projects tasks or in the projects roadmaps | <input type="checkbox"/> |  GR07 |
| GR07-03 | Database methods used should be tested properly | <input type="checkbox"/> |  GR07 |
| OR01-01 | Gets prompt when created organization | <input type="checkbox"/> |  OR01 |
| OR01-02 | Write test for endpoint(s) creating and saving the organization to db | <input checked="" type="checkbox"/> |  OR01 |
| OR01-03 | Create a organization through a UI | <input checked="" type="checkbox"/> |  OR01 |
| OR02-01 | Can invite new members through a UI, either via username or mail, or with google account | <input checked="" type="checkbox"/> |  OR02 |
| OR02-02 | Write test for endpoint adding user ID's to the given organization | <input checked="" type="checkbox"/> |  OR02 |

| | | | |
|--|--|-------------------------------------|---|
| OR03-01 | Get a overview of all members currently in the organization | <input checked="" type="checkbox"/> |  OR03 |
| OR03-02 | Can set a members role to fit their job within the organization | <input checked="" type="checkbox"/> |  OR03 |
| OR03-03 | Write test for endpoints adding the users ID's to the different role arrays | <input checked="" type="checkbox"/> |  OR03 |
| OR04-01 | The administrator can delete, add and change rolls of the members | <input checked="" type="checkbox"/> |  OR04 |
| OR04-02 | The endpoint (change roll, add member and delete member) should be tested | <input checked="" type="checkbox"/> |  OR04 |
| OR05-01 | Test before editing and after editing to check that the updates were done | <input checked="" type="checkbox"/> |  OR05 |
| OR06-01 | Can view organizations I am a member of and switch between these | <input checked="" type="checkbox"/> |  OR06 |
| OR06-02 | Can preview key information such as num. of members, description and my role in that organization | <input checked="" type="checkbox"/> |  OR06 |
| P02-01 | Can add a member / admin in the organization as a project monitor to a project | <input checked="" type="checkbox"/> |  P02 |
| P02  OPEN | Can add a member / admin in the organization as a project member to a project | <input checked="" type="checkbox"/> |  P02 |
| P02-03 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  P02 |
| P03-01 | Authentication for editing the project. should not be allowed if user is not admin or project lead | <input checked="" type="checkbox"/> |  P03 |
| P03-02 | Can view all information about the project and edit key points | <input checked="" type="checkbox"/> |  P03 |
| P04-01 | Authentication for removing members, should not be allowed if user is not admin or project lead | <input checked="" type="checkbox"/> |  P04 |
| P04-02 | Can view all current project members in a list | <input checked="" type="checkbox"/> |  P04 |
| P05-01 | Can archive a project, with all its current artifacts then also archived | <input checked="" type="checkbox"/> |  P05 |

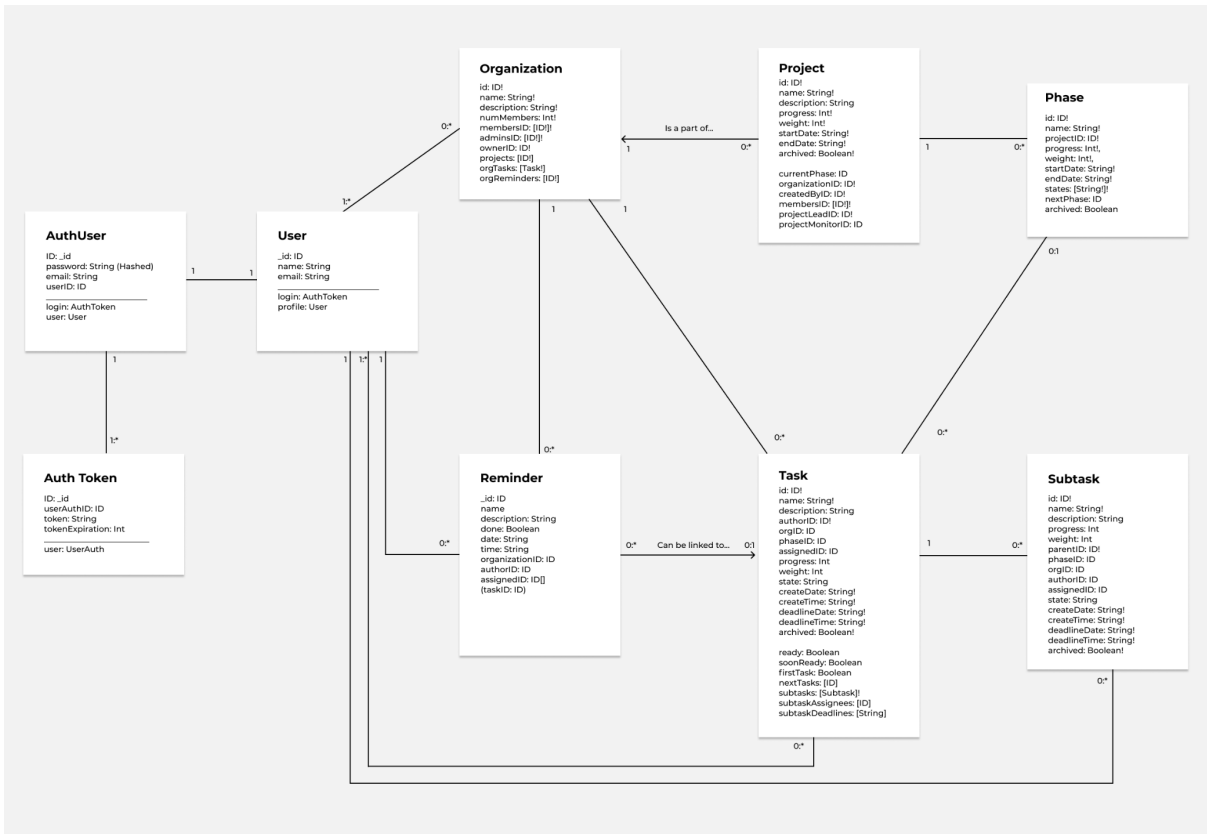
| | | | |
|--|---|-------------------------------------|---|
| P05  OPEN | Can see the total progress within a project, when done it should display for user to see | <input checked="" type="checkbox"/> |  P05 |
| P06-01 | Can assign tasks and subtasks in a project to project developers | <input checked="" type="checkbox"/> |  P06 |
| P06-02 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  P06 |
| PH01-01 | Can make a new project, 1 phase added by default to the new project | <input checked="" type="checkbox"/> |  PH01 |
| PH01-02 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  PH01 |
| PH02-01 | Can add more phases to my project | <input checked="" type="checkbox"/> |  PH02 |
| PH02-02 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  PH02 |
| PH03-01 | Can edit my current phases | <input checked="" type="checkbox"/> |  PH03 |
| PH03-02 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  PH03 |
| PH04-01 | Can allocate tasks to a phase or create a task in a current phase | <input checked="" type="checkbox"/> |  PH04 |
| PH04-02 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  PH04 |
| T01-01 | User can easily create a task by only just filling in a taskName, or more detailed setting and filling in all the other data fields | <input checked="" type="checkbox"/> |  T01 |
| T01-02 | Creating a task should be doable in both the users organization, and depending on the users role, in a phase | <input checked="" type="checkbox"/> |  T01 |
| T01-03 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  T01 |
| T02-01 | Can go to a page where tasks assigned to the user can be found | <input checked="" type="checkbox"/> |  T02 |
| T02-02 | The page filters out tasks that are more than a week ahead | <input checked="" type="checkbox"/> |  T02 |
| T02-03 | Easily view what tasks are for "today", "tomorrow" and "upcoming" | <input checked="" type="checkbox"/> |  T02 |
| T02-04 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  T02 |

| | | | |
|------------------------|--|-------------------------------------|---|
| T03-01 | Can mark my own tasks or my assigned tasks as done | <input checked="" type="checkbox"/> |  T03 |
| T03-02 | Can mark subtasks to my tasks as done, the parent task should be updated aswell | <input checked="" type="checkbox"/> |  T03 |
| T03-03 | Marking a task as done updates the progress in a current phase, which updates the progress in a current project | <input checked="" type="checkbox"/> |  T03 |
| T03-04 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  T03 |
| T04-01 | Can edit the tasks i have made, or the tasks in a project as the project lead | <input checked="" type="checkbox"/> |  T04 |
| T04-02 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  T04 |
| T05-01 | Can create a task in a phase within a project, tasks are not directly linked to a project. | <input checked="" type="checkbox"/> |  T05 |
| T05-02 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  T05 |
| T06-01 | Can use a system in taskflow to send mail to other organization users | <input type="checkbox"/> |  T05 |
| T06-02 | Users should be notified when they have been sent a mail via taskflow | <input type="checkbox"/> |  T05 |
| T07-01 | Can create a subtask to a task, this lets me spilt tasks into more manageable tasks and gives me more control over my workflow | <input checked="" type="checkbox"/> |  T07 |
| T07-02 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  T07 |
| T08-01 | Can archive my tasks and subtasks | <input checked="" type="checkbox"/> |  T08 |
| T08-02 | Archived tasks can be viewed in a seperate list | <input type="checkbox"/> |  T08 |
| T08-03 | Archived tasks should be filtered out of the active task lists | <input checked="" type="checkbox"/> |  T08 |
| T08-04 | Database methods used should be tested properly | <input checked="" type="checkbox"/> |  T08 |
| U01-01 | Create a user through a UI | <input type="checkbox"/> |  U01 |
| U01-02 | Create test for user creation | <input checked="" type="checkbox"/> |  U01 |
| U01-03 | User info is saved to DB and password is hashed + salted (if OAuth2 see U01-04) | <input checked="" type="checkbox"/> |  U01 |

| | | | |
|------------------------|---|-------------------------------------|---|
| U01-04 | User creation follows Oauth-2 protocols | <input checked="" type="checkbox"/> |  U01 |
| U02-01 | Can log in to an existing user through a UI | <input checked="" type="checkbox"/> |  U02 |
| U02-02 | When logging in, authorization token in Auth-microservice is created and sent to user | <input checked="" type="checkbox"/> |  U02 |
| U02-03 | Store authorization token locally and attach it to header of GraphQL requests | <input checked="" type="checkbox"/> |  U02 |
| U02-04 | Log in requests follows Oauth-2 protocols | <input checked="" type="checkbox"/> |  U02 |
| U02-05 | Write test for log-in and authorization-check endpoints | <input checked="" type="checkbox"/> |  U02 |
| U03-01 | Can view user-information | <input type="checkbox"/> |  U03 |
| U03-02 | Can edit user-information | <input type="checkbox"/> |  U03 |
| U03-03 | Can delete user, gets an "are you sure" prompt before executing | <input type="checkbox"/> |  U03 |
| U03-04 | Server endpoints are tested | <input checked="" type="checkbox"/> |  U03 |
| U04-01 | Gets a log out prompt "are you sure?" | <input checked="" type="checkbox"/> |  U04 |
| U04-02 | Remove tokens and other storage info | <input checked="" type="checkbox"/> |  U04 |
| U05-01 | A checkbox is displayed in the login UI | <input type="checkbox"/> |  U05 |
| U05-02 | A token with longer expiration time is generated and returned to the user | <input checked="" type="checkbox"/> |  U05 |
| U06-01 | Can navigate to all the pages on the site from the navbar | <input checked="" type="checkbox"/> |  U06 |
| U06-02 | 3 Clicks rule is applied | <input checked="" type="checkbox"/> |  U06 |

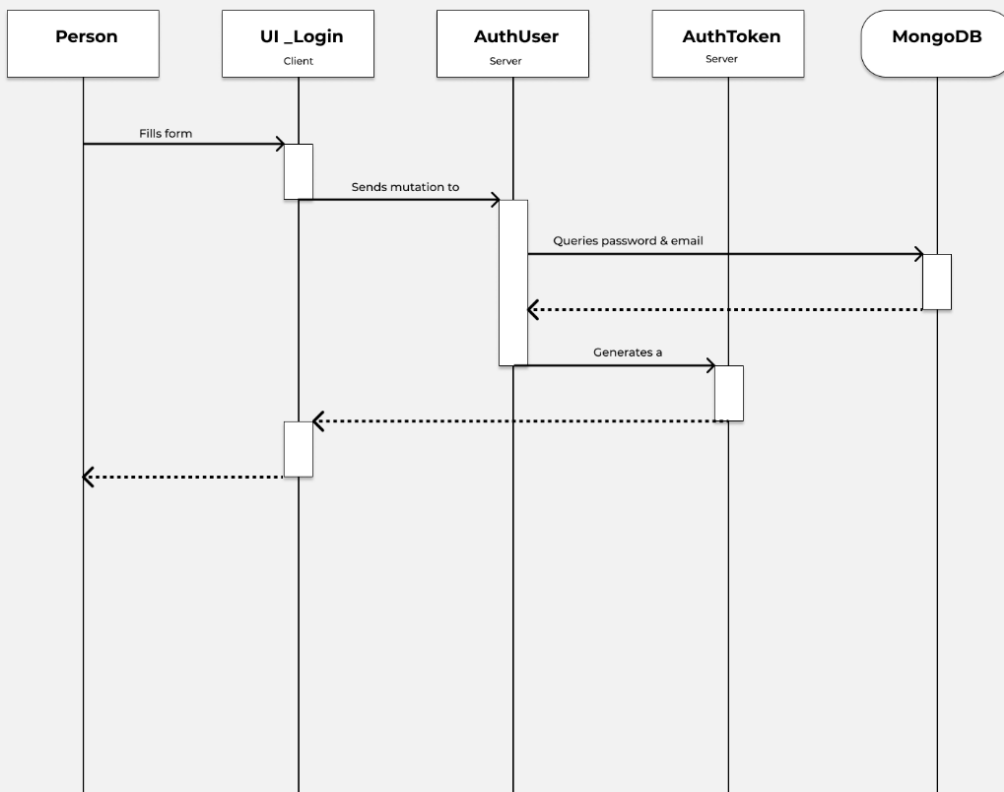
8.7.4 Domain model

8.7.4.1 Domain Model

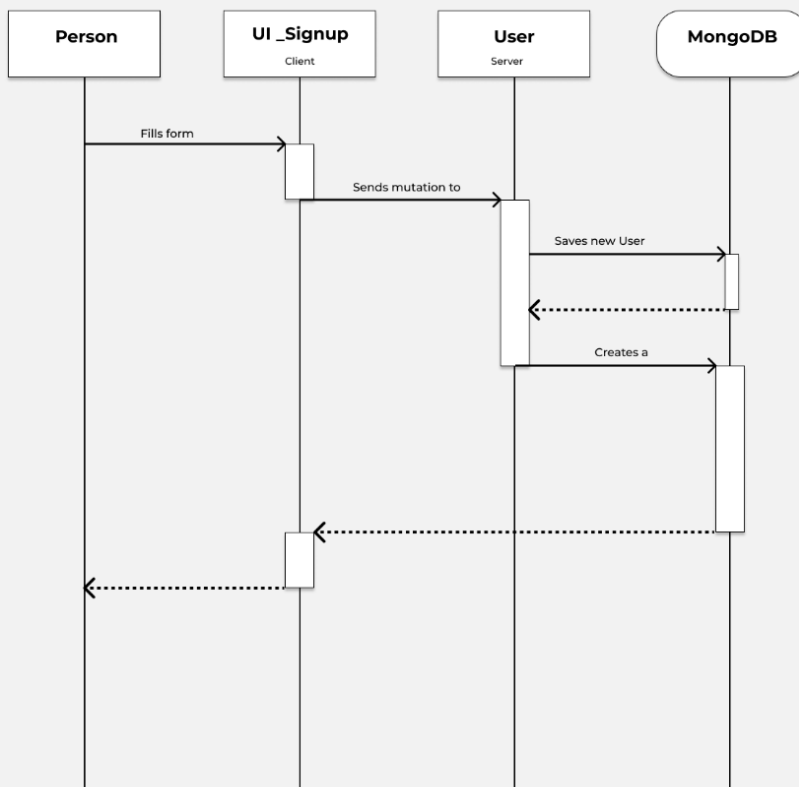


8.7.4.2 Sequence diagrams

Logging into an existing user



Creating a user



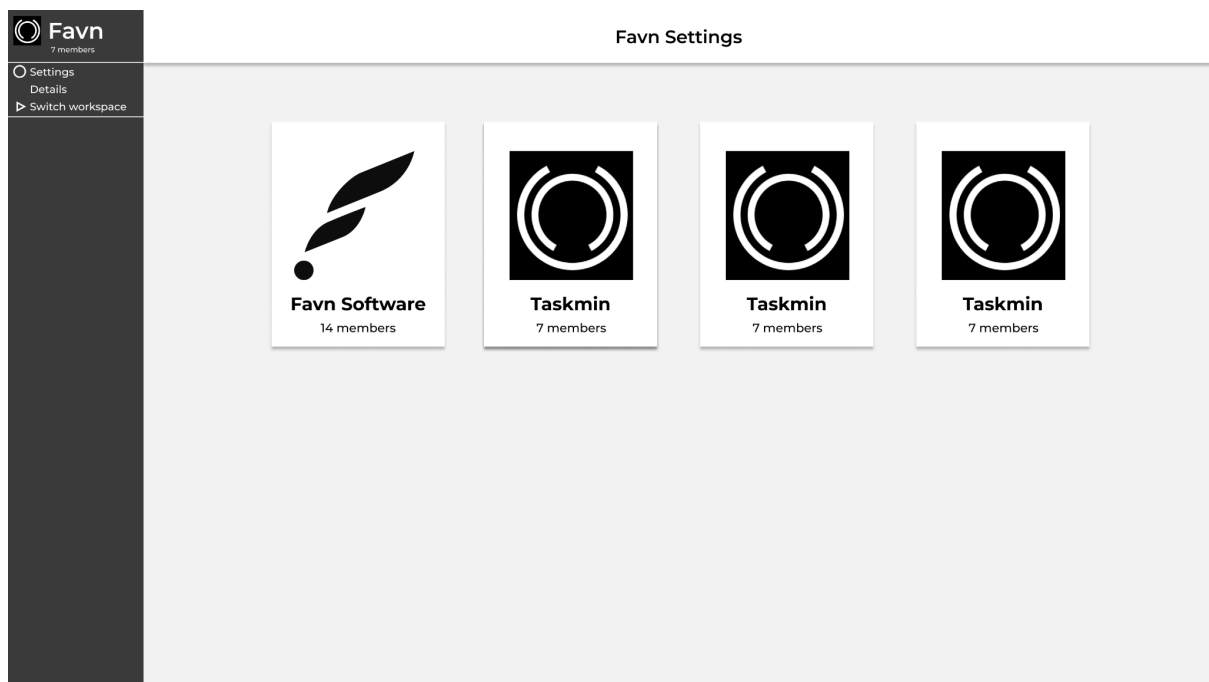
8.7.5. Prototypes

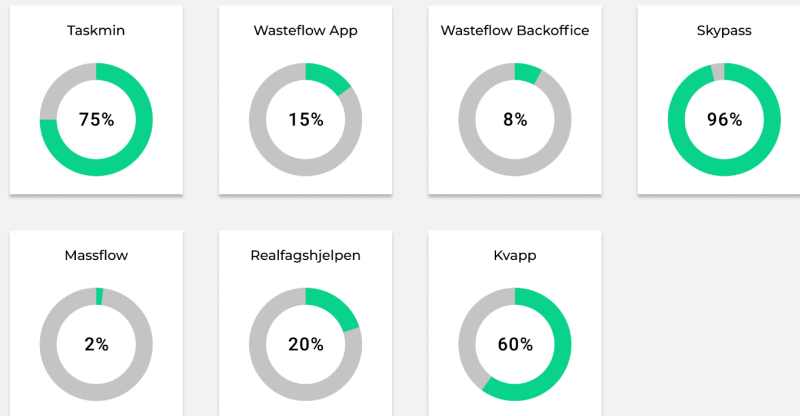
8.7.5.1 Wireframes

The wireframes are created by using [Figma](#).

The following is a link to all wireframes and diagrams for Taskflow application:

<https://www.figma.com/file/dDURwJqN3ec32cIynMF74h/Taskmin?node-id=0%3A1>

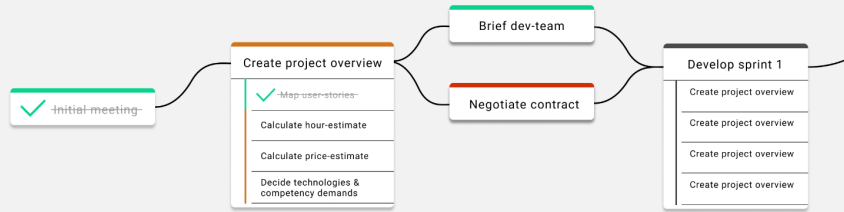




Today

- ✓ Org: Create new seller job listing
 - ✓ Taskmin: Add 'my tasks' example to figma
 - ✓ Taskmin: Plan task workshop
- Rebranding:** check status
- [Bonus] Taskmin:** Add 'create task' design to figma
- [Bonus] Org:** Plan event for next friday

This week



Taskflow

75%

Total progress

Statistics

| | |
|---------------------|------|
| Total hours: | 400 |
| Hours worked: | 300 |
| Hours remaining: | 100 |
| Weekly workload: | 25 |
| Future w. workload: | 20,1 |

On track

Team

- Alexander Carlsen
- Gaute Wierød Rønning
- Mohammad Al Nayef

Project lead
Developer
Designer

● Frontend
● Backend
● Design

44,3%

Sprint 2 progress

Sprint 2

[Dropdown for andre sprints]

Due today

- ✓ @org: Create new seller job listing
- ✓ Taskmin: Add 'my tasks' example to figma
- ✓ Taskmin: Plan task workshop

Rebranding: check status

[Bonus] Taskmin: Add 'create task' design to figma

[Bonus] Org: Plan event for next friday

Due this week

- ✓ @org: Create new seller job listing
- ✓ Taskmin: Add 'my tasks' example to figma
- ✓ Taskmin: Plan task workshop

Rebranding: check status

[Bonus] Taskmin: Add 'create task' design to figma

[Bonus] Org: Plan event for next friday

Statistics

| | |
|---------------------|----|
| Total hours: | 50 |
| Hours worked: | 22 |
| Hours remaining: | 28 |
| Weekly workload: | 25 |
| Future w. workload: | 28 |

3 hours behind



75%

Total progress

On track

Statistics

| | |
|---------------------|------|
| Total hours: | 400 |
| Hours worked: | 300 |
| Hours remaining: | 100 |
| Weekly workload: | 25 |
| Future w. workload: | 20,1 |

Team

- Alexander Carlsen
- Gaute Wierød Renning
- Mohammad Al Nayef

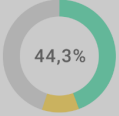
Project lead
Developer
Designer



- Frontend
- Backend
- Design

Sprint 2

[Dropdown for andre sprints]



44,3%

Sprint 2 progress

Statistics

| | |
|---------------------|----|
| Total hours: | 50 |
| Hours worked: | 22 |
| Hours remaining: | 28 |
| Weekly workload: | 25 |
| Future w. workload: | 28 |

3 hours behind

Due today

- ✓ Org: Create new seller job listing
- ✓ Taskmin: Add 'my tasks' example to figma
- ✓ Taskmin: Plan task workshop

Rebranding: check status

[Bonus] Taskmin: Add 'create task' design to figma

[Bonus] Org: Plan event for next friday

Due this week

- ✓ Org: Create new seller job listing
- ✓ Taskmin: Add 'my tasks' example to figma
- ✓ Taskmin: Plan task workshop

Rebranding: check status

[Bonus] Taskmin: Add 'create task' design to figma

[Bonus] Org: Plan event for next friday