Digital Havn | Norkart

TDT4290|Customer Driven Project

Department of Computer Science

Group 10

Gard Drag-Erlandsen
Jesper Elverum
Eli Fjellbirkeland Johannesen
Johan Otto Munkeby
Erik Salvesen
Sara Sveen

Supervisor
Letizia Jaccheri

Trondheim | Autumn 2023

NTNU
Norwegian University of Science and Technology

Executive Summary

Several Norwegian ports and companies have created a mutual project called "Digital Havn" which aims to digitalize the infrastructure related to port management. This is due to outdated systems and little progress over the last decades. The team became part of this project through the course *TDT4290 - Customer Driven Project* at NTNU, with the customer Norkart.

The objective for our project was to create a web based map client for port management, and make it as user friendly as possible. This was due to the existence of similar systems that were too complicated for the average user.

We conducted an agile development process inspired by Scrum and Extreme Programming, with five sprints over 12 weeks. The creation of issues and criteria was mainly based on three user interviews held during sprint 2 and 3. The team focused on implementing sustainable software, and reflected upon diversity and the use of artificial intelligence in the project.

The team created a prototype of a web based map client that fulfilled the customers criteria. The final product can be viewed (here) and the source code for the prototype can be accessed (here). The report describes the development process of the web based map client, and grounds the decisions made.

In general our team dynamic was strong throughout the project, and both the team and the customer were very satisfied with the final product. For future work the inclusion of a 3D-based map client using depth data would increase the functional usability of the system.

Trondheim - Autumn 2023

Table of Contents

List of Figures

List of Tables

1 Introduction

In our project, we addressed the problem of digitalizing Norwegian ports. Our specific focus was on creating a more user-friendly product than existing solutions, to better cater to a wider and more diverse range of end-users. To address this, we developed a comprehensive prototype of a web application. This prototype was refined through multiple stages, including three usability tests and in-depth user interviews, and ongoing communication with end-users from various Norwegian ports.

The culmination of our efforts is a final product that leverages Leaflet.js to provide an interactive map. This map integrates geographical port data from NGIS-OpenAPI, offering an intuitive and user-centric interface for accessing and editing critical port information.

The project was undertaken as part of the TDT4290 - Customer Driven Project course at NTNU Trondheim. The course challenges students to leverage their accumulated knowledge within software development, methodologies, and architecture to carry out a project for a professional customer, and thereby offering a practical insight into the software development industry.

1.1 Overall context

The project's customer, Norkart, is at the forefront of municipal engineering, mapping, and property information in Norway. With access to the country's expansive geographic information data warehouse, Norkart plays a pivotal role in the ongoing digitalization efforts of Norwegian ports (Norkart, 2023). Key examples are their engagement with the Port of Oslo and Kartverket in initiatives like "Norsk digital havneinfrastruktur" and "Digital Havn." This included a central role in the work on port data standardization, which is crucial for the infrastructure required to digitalize Norwegian ports. The goals of these projects are twofold: to enhance efficiency in port operations' and to guide these operations towards greater environmental sustainability. This is achieved through comprehensive mapping, digitalization, and the development of a shared digital framework across the Ports of Norway (Oslo Havn, 2023).

In this context, Norkart has partnered with us, students enrolled in the TDT4290 course. This collaboration presents a significant opportunity: if we, as part of Norkart's team, can successfully demonstrate a prototype that not only improves user-friendliness but also appeals to a wider range of end-users, it could potentially unlock additional funding. This funding would enable the further development of a more comprehensive and sophisticated product, advancing Norkart's digital transformation efforts in the Norwegian port sector.

1.2 Motivation

The project offered our team a unique opportunity to gain practical experience in addressing a technical challenge with tangible real-world implications for a significant number of end-users. Every team member felt a personal connection to the project, appreciating the significance and practicality of digitizing Norwegian ports. This venture also allowed us to enhance our knowledge of geospatial technologies and tools — areas previously unfamiliar to us. Such experiences could be beneficial for each team member in future professional endeavors as it serves as an introduction to professional project management. Additionally, Norkart's intention to not only use, but potentially improve our product further motivated us to create a solution of substantial value.

1.3 Demands

At the onset of the project, the customer outlined several key requirements. The primary feature was to make the product user-friendly, with an emphasis on creating an interface that was intuitive and easy to navigate. The project's goal was to develop a prototype, which meant that the initial stages did not heavily prioritize aspects like quality or security. Despite this, the customer specifically requested thorough documentation of the product. This was to ensure that the product would be easily comprehensible and could be effectively enhanced and expanded in the future.

1.4 Results

The team effectively developed a fully functional web application that met the initial customer requirements, and surpassed expectations for the prototype in some aspects. This achievement was accompanied by comprehensive documentation of the application.

In addition, the team helped highlight the importance and relevance of this application to the geospatial community in Norway. This was achieved by delivering a presentation and actively participating at FOSS4G, a seminar focused on open-source geospatial software. The impact of our team's work was further magnified during a technical demonstration at a quarterly board meeting. This demonstration played a crucial role in the decision-making process regarding additional funding for the project. The application is now operational and accessible via the link provided in the subsequent section.

1.5 Resources

- **Link to Final Presentation:** Slides
- **Link to FOSS4G Presentation:** Slides
- **Link to source code:** Gtihub
- **Link to product:** `https://folk.ntnu.no/eriksalv/TDT4290-leaflet-client/`
- **Link to design prototype:** Figma

2 Planning

Planning plays a pivotal role in the project's inception, ensuring not only customer satisfaction but also the smooth progression of product development. Effective planning enhances efficiency, mitigates risks, and establishes robust routines (Arend et al., 2017). As elaborated in Section 4.2.1 Why Scrum?, our team adheres to the Scrum methodology, underscoring the significance of well-defined routines. In our Scrum implementation, we delineated sprint timelines, with the specifics of each sprint mapped out during dedicated sprint planning meetings.

This section summarizes the project schedule, roles, tools, and communication infrastructure. It also outlines our approach to quality assurance and risk management. All agreements on organizational aspects and guiding principles from the planning phase are formally documented in the team contract, accessible in Appendix G.

2.1 Project Schedule

During the initial planning stage, our team segmented the project into five distinct sprint phases, aligning with the Scrum development methodology outlined in Section 4. These phases have

been graphically represented in Figure 1. Our efforts were directed toward achieving specific sprint goals within each sprint, as articulated in Section 8. We organized the tasks and activities, and clarified the goal for each sprint during dedicated sprint planning meetings. To enhance the project's structure and coordination, the team established regular meeting times, to keep up with the Project Schedule and goals.

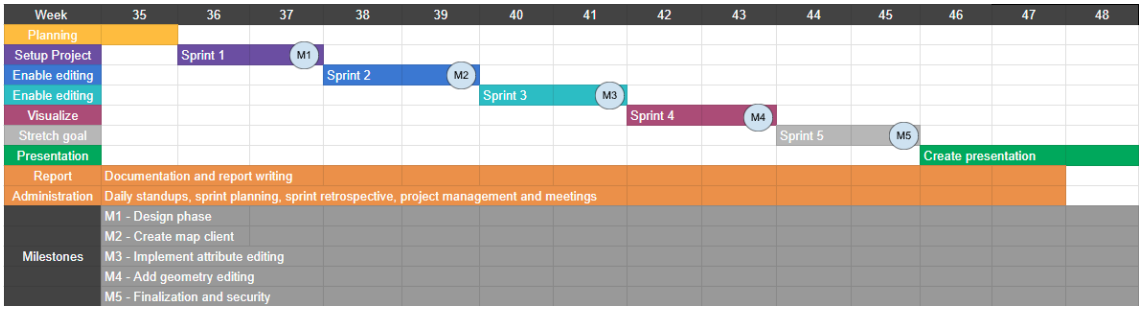


Figure 1: Gantt diagram illustrating the project schedule

Monday team meetings, from 09:15 to 14:00, prioritized full attendance in the first hour. Members had flexibility to leave for other commitments, but it was encouraged to stay until 14:00 to maximize collaboration, and this was generally the practice. This protocol also applied to Friday meetings, encouraging group engagement despite members' enrollment in other courses.

Wednesday supervisor meetings were mandatory for all team members while biweekly group leader meetings were exclusively attended by the project manager. In the team lead's absence, another member would attend.

The customer meetings were initially slotted for Thursdays from 10:00 to 11:00, but were usually rescheduled to accommodate the customer. Figure 2 outlines the team's meetings during a regular week of the project.

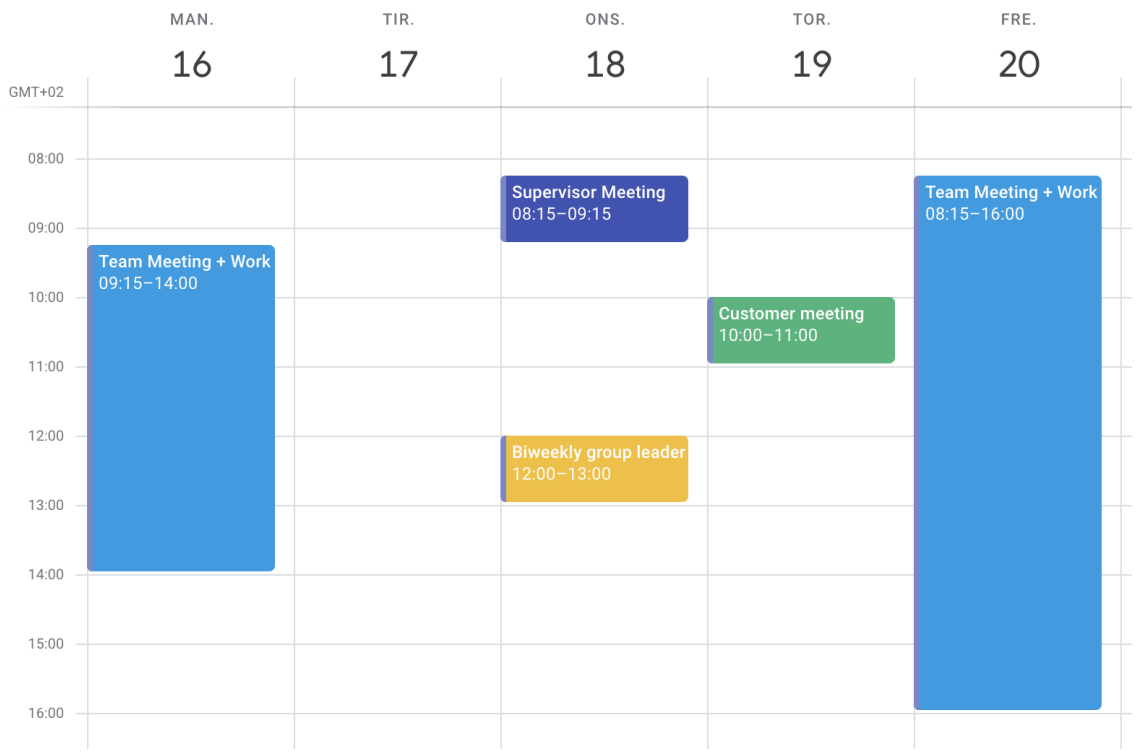


Figure 2: The weekly schedule displaying meeting times

2.2 Team Organization

Table 1 illustrates the team organization and assigned roles. In addition to the roles outlined in the compendium (Jaccheri, 2023), we introduced some additional roles we saw fit. We assigned the roles according to prior expertise and individual preferences, but also had diversity in mind (see Section 5.2.1).

It's important to stress that even though specific team members were assigned to different tasks, the group collectively could contribute when needed. While an individual is assigned to a task, they are not solely responsible for its execution. For example, the report manager ensures the report is effectively written but does not write it independently. Effective delegation and attention to deadlines are crucial in this context.

Roles	Responsibilities	Assignee
Project Manager	- Organize and lead meetings. - Attend group leader meetings, and inform about decisions and updates made at the meeting. - Assign and monitor tasks.	Sara
Scrum Master	- Uphold Scrum methodology. - Lead scrums, planning, retrospectives. - Update backlog and Kanban board.	Erik
Tech Lead	- Distribute coding tasks. - Make technology related decisions. - Coaching other members on technologies.	Jesper
Quality Manager	- Oversee product and report quality. - Ensure pre-defined standards are followed.	Gard
Report Manager	- Update and track report progress. - Align with reporting deadlines.	Johan
Lead architect	- Make and document architectural decisions. - Direct design and usability test preparations.	Eli
UX manager	- Main responsibility for Figma prototype and mockups. - Direct usability test preparations.	Eli
Documentation Manager	- Main responsibility for code documentation and README on Github. - Ensure document quality and organization (e.g. for meeting documents).	Erik
Developer	- Coding, testing, and code review. - Log time for tasks. - Update progress on user story tasks.	Everyone
AI and Sustainability Manager	- Run AI and sustainability workshops. - Contribute to report sections on AI and sustainability.	Johan
Diversity Manager	- Lead diversity workshops. - Manage diversity content in the report.	Sara
Security Manager	- Focus on security aspects. - Facilitate security-related discussions.	Gard

Table 1: Distribution of roles on the team

2.3 Tools and Infrastructure

A well-organized product relies on a streamlined workflow, where tools refer to digital resources facilitating software development (Vaughan-Nichols, 2003). These tools form the team's infrastructure, promoting a healthy development environment. Emphasizing efficiency, we carefully selected a concise set of tools during the planning phase. This was based on our familiarity with the tools and their unique qualities. Each tool served a specific purpose, contributing to overall organization and maintaining a well-defined structure throughout development.

2.3.1 Communication

The team primarily used Slack for communication, creating specific channels for organized messaging and information retrieval. For quick, everyday messages, we used Messenger, reserving it

for brief notifications and less critical topics. In contrast, Slack was designated for more significant and lasting information.

For customer interactions, we chose Microsoft Teams, aligning with the customer's existing usage. Teams also facilitated screen sharing during meetings, accommodating the absence of monitors or widescreens in our meeting spaces. Additionally, it served as the platform for virtual meetings to include any team members unable to attend in person.

2.3.2 Code Repository and Coordination

GitHub was our platform of choice for software development, as suggested by the customer. It was used for repository management, issue tracking, code review, and workflow monitoring. The customer had already set up a GitHub repository and recommended using GitHub Projects for workflow tracking through a Kanban board. Github streamlined task delegation and automatically deleted feature branches on task completion.

2.3.3 Coding

Visual Studio Code (VSCode) was the unanimous choice for code editing across the team. Familiarity with VSCode among all members facilitated debugging, knowledge sharing, and pair programming. Additionally, VSCode is well suited for development using JavaScript/TypeScript, and has an extensive plugin library for specific functionalities that may become useful.

2.3.4 Common workspace

Google Drive was our central hub for document storage. We used Google Docs for notes and agendas, and Google Sheets for time tracking. Both the supervisor and customer had access to these documents for review and feedback. For the final report, we opted for LaTeX using Overleaf due to its capabilities in managing complex documents, allowing easy section tracking and collaborative editing.

2.4 Quality Assurance

Quality Assurance (QA) is a proactive strategy that plays a pivotal role in upholding product quality during development. It ensures alignment with customer expectations and adheres to international standards such as ISO 9126 (*ISO 9001:2015*, 2015). A thorough grasp of QA practices from both the developer's and customer's viewpoints is key to fostering collaborative success, paving the way for successful project outcomes (Westland, 2023). To guarantee the quality of our product, we implemented several measures throughout our project, as outlined below.

2.4.1 Response time

Timely responses in QA are crucial for project success. They maintain project momentum, prevent delays, and ensure deadlines are met. Swift responses enhance customer satisfaction and help identify and resolve issues early, reducing project risks. They foster effective collaboration, reduce uncertainty, and support quality control. After discussing with the customer we agreed on the response times presented in Table 2.

Task	Timeframe
Approval of agenda, including questions for customer meeting	24 hours
Approval of minutes of customer meeting	24 hours
Answer to a question in Teams	24 hours
Other inquiries via Teams or email	24 hours
Feedback on phase documents the customer would like for review	48 hours
Approval of phase documents	48 hours

Table 2: Task Timeframes

2.4.2 Routines

Routines in QA for producing high-quality work is vital for maintaining consistency, meet customer expectations, complying with legal requirements, providing thorough documentation, and ensuring effective code review and version control processes. These practices collectively contribute to the overall success and quality of a software development project (Dönmez et al., 2016). Our core routines and pragmatic standards are detailed below, and are also available in the project's CONTRIBUTING.md.

Coding Style:

- Use lowerCamelCase for variables, functions, and filenames.
- Write global constants in CONSTANT_CASE.
- In TypeScript, prefer 'let' or 'const' over 'var'.
- Favor string templates over concatenation (e.g., `const greeting = ‘Hello {name}!’;`).
- Install Prettier and ESLint extensions in Visual Studio Code.
- Adhere to ESLint and Prettier rules (see `.eslintrc.json` and `.prettierrc`).
- Follow TypeScript compiler rules for program correctness (see `tsconfig.json`).

Code review:

Our team's code quality is maintained through a code review process. Every pull request with code changes must be reviewed by at least one other team member. The reviewer is required to provide detailed, actionable feedback directly on the pull request, highlighting specific changes or improvements needed for approval. Additionally, referencing specific code snippets in the pull request is encouraged to ensure clear and precise feedback. This approach ensures that every code modification is carefully vetted and meets our quality standards.

Commits: Our code management process emphasizes clarity and organization. We utilize conventional commits, with 'feat' denoting feature additions, 'docs' for documentation updates, and 'fix' for bug fixes. This standardized approach helps in categorizing changes effectively. Commit messages should be short and concise, capturing the essence of the change without unnecessary details. Additionally, frequent commits are encouraged to maintain a clear and granular history, aiding in easier tracking and understanding of the project's evolution.

Workflow:

1. Create a new feature branch for every issue, based on the 'develop' branch, with the branch name id-issue-title. You can use the "create branch" button on each issue to do this easily. Strive to keep branches small, ideally with fewer than 200 lines changed.
2. Always create a pull request before merging a branch into the 'develop' branch for code review (CR).
3. Ensure that the pull request is only merged when it is approved by at least one reviewer. Avoid creating pull requests directly from a feature branch to the 'main' branch.
4. At the end of a sprint, merge the 'develop' branch into the 'main' branch.
5. Create a release after merging to 'main'.

Issues:

- Ensure detailed descriptions for each issue for clarity and future reference.
- Use relevant labels to categorize issues: 'feature', 'documentation', 'bug', 'research'.
- Assign priority and size to issues with existing labels.
- Assign issues to developers and move to 'In progress' on the Kanban board before starting.
- For review-ready issues, transition to 'In review' and assign reviewers.
- Move issues to 'Done' once the corresponding pull request is merged into 'develop'.
- Place dependent issues in 'Blocked', specifying the blocking issue.
- Use 'Ready' for planned but not yet started issues in the current sprint.

2.4.3 Templates and Standards

Templates and standards in quality assurance streamline processes, reduce errors, improve clarity, and enhance overall project efficiency and quality. They are essential tools for maintaining consistency and reducing stress within the project team. All the templates used are available in Appendix E.

2.5 Risk Management

Risk assessment is a proactive approach that plays a vital role in identifying, addressing, and mitigating potential risks within a project. It entails systematically evaluating potential risks, gauging their likely impact and severity, and developing strategies to either reduce their likelihood or mitigate their consequences (Anthony Jnr et al., 2016, 31). Early detection of these risks is key to minimizing long-term costs and is essential in ensuring the project meets the specified requirements outlined in Section 6 Requirements Specification (Han & Huang, 2007, 32).

In software development, unexpected issues can arise from internal team actions and external factors. To address these, we conducted a comprehensive risk analysis, identifying challenges related to team dynamics and technical hurdles. Relying on past experiences and the challenge of client engagement, we documented potential risks and mitigation strategies in Table 3, assessing each based on likelihood, potential impact, and overall risk.

Our risk assessment underscored the crucial role of effective team communication, emphasizing the importance of alignment on project progress. Additionally, maintaining communication with the client was identified as a vital measure in preventing many of the identified risks.

Table 3: Risk Assessment Table
(I = Impact, P = Probability, R = Total Risk, H = High, M = Medium, L = Low)

ID	Area	Risk Factor	Consequence	I	P	R	Strategy	Responsible
R1	Team	Inadequate internal communication	Redundant work, delays, technical debt	H	M	H	Slack updates, daily scrums, GitHub project tracking	Scrum master
R2	Development	Neglecting code quality/testing	Reduced maintainability, technical debt	H	M	H	Enforce testing standards, quality oversight	Quality manager
R3	Report	Overemphasis on functionality	Rushed, low-quality reporting	H	M	H	Parallel report and development work	Quality manager
R4	Team	Inadequate member contribution	Insufficient progress, team friction	M	M	M	Daily check-ins	Team leader, all members
R5	Customer	Ineffective customer communication	Misaligned expectations, project discrepancies	H	L	M	Weekly meetings, customer inclusion in development	Team leader
R6	Development	Unclear project scope	Incomplete functionality	M	M	M	Detailed sprint planning, clear customer demand	All
R7	Team	Member absences	Increased workload for others	M	H	M	Remote work support, early leave communication	All
R8	Customer	Customer indecision during development	Wasted work, increased workload	M	L	L	Ongoing customer dialogue during development	Team leader, Scrum master
R9	Team	Permanent team member departure	Higher workload, knowledge loss, role redistribution	M	L	L	Comprehensive documentation, shared responsibilities	All, Scrum master

2.6 Effort Registration

Effort registration is crucial in project management for tracking progress, managing resources, controlling costs, and meeting goals and deadlines. It aids in informed decision-making throughout the project life-cycle Heijstek & Chaudron, 2008.

We used an Excel spreadsheet for recording work, categorizing activities into Meeting, Research, Coding, Administrative, Report, Design, and Lectures. After Sprint 5, we created a pie chart

(Figure 3) to visualize time distribution across these activities, excluding time outside sprints 1 to 5. This analysis revealed a significant allocation of time to coding, report writing, and meetings. Despite a desire for more coding time, the quality of code benefited from extensive meeting discussions, resulting in a satisfactory balance of time allocation.

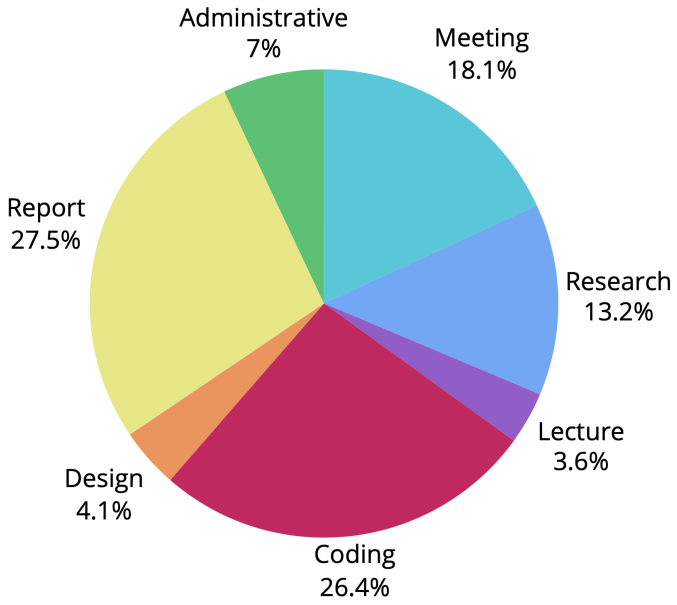


Figure 3: Illustrating total hours spent on each activity

Further, we analyzed time spent per activity for each sprint using a bar chart (Figure 4). This showed a decline in Research and a consistent increase in Coding from sprint 2 to 5, reflecting our learning curve and growing efficiency. The time dedicated to the report also increased progressively, indicating a focused advancement in our project work.

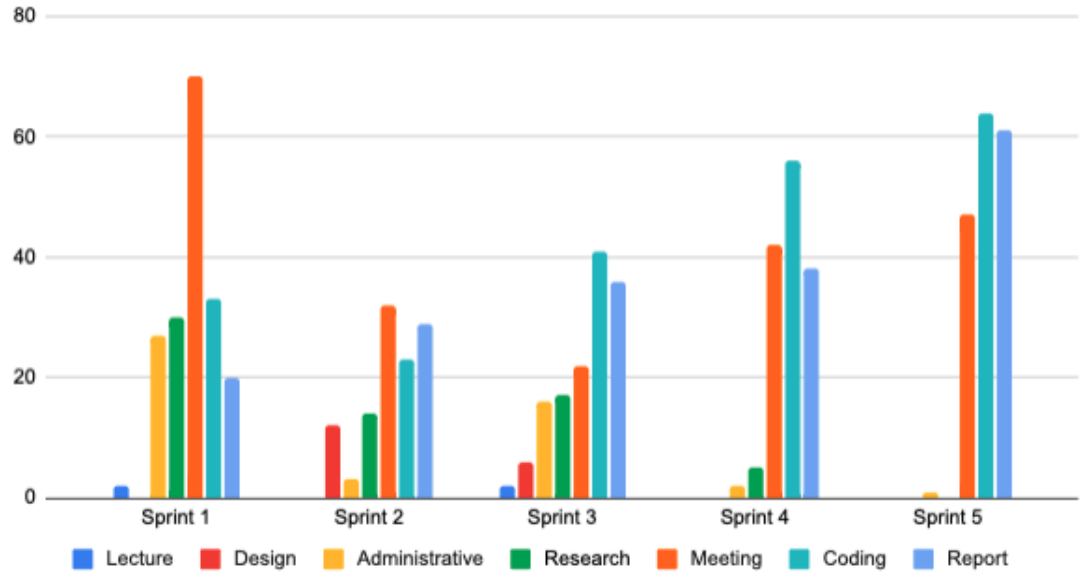


Figure 4: Shows the total hours spent on each activity for each sprint

3 Problem Space and Solution Space

This section describes the problem and solution space discovered through preliminary research, the introductory presentation by Norkart and subsequent user interviews with industry experts.

3.1 Problem space and business goals

Our customer, Norkart, plays a significant role in the "Digital havn" project, which involves creating an infrastructure and standard for port data. This laid the groundwork for our task to develop a web-based map client for visualizing and updating this data. The customer aims for reusability across all Norwegian ports, easy integration into port websites, and user-friendliness without requiring GIS expertise. These goals were outlined in the initial customer meeting and are summarized in Figure 10 and Table 4.

Table 4: Business Goals

Business Goals	
ID	**Description**
BG1	User friendly interface
BG2	Visualize collected geographical port data on a web based map
BG3	Users should be able to edit properties and geometry of port data in an easy way
BG4	System should require minimal technical skills and GIS-background
BG5	The map client should be embeddable in a port's website
BG6	It should be easy to continue development on the project in the future
BG7	Reliable service
BG8	Accessible service
BG9	Trustworthy and secure source of data

3.2 Data source, ownership, and IPR

The work on digitalizing Norwegian ports started by creating a standardization of different types of port objects, known as the Havnedata standard. This allowed ports to manage their data in a system called NGIS. NGIS is a platform maintained by Kartverket that allows management of geographical data across organizations and software in a central database. The NGIS platform can be interacted with through an API such as the REST-based NGIS-OpenAPI, which is what we used for our project.

In NGIS, data ownership is specific to each dataset, and external access requires the data owner's permission. Although our project's code is open-source, we lack rights to the data. To overcome this, Norkart and Kartverket provided us test database credentials for development and external use. Additionally, NGIS data can be viewed as map images via WMS services, such as the Havnedata WMS, without authorization.

3.3 Goals and limitations for the student project

During the initial presentation, the customer set specific subgoals, detailed in Figure 11 in Appendix A. The primary focus was on a 2D web-based map client, covered in the first three subgoals. The

last three goals, involving implementing geometry editing, a 3D map client, and NGIS-OpenAPI security review, were optional. Norkart requested omitting user authentication to save time. Additionally, editing polygon shapes was deemed challenging due to "shared geometry" issues and was removed from the project's scope.

3.4 Existing Solutions

There are some existing solutions, such as plugins for QGIS - a free and open source desktop application for GIS - and Grieg Connect, that partially meet the outlined business requirements. However, user interviews revealed, as detailed in Section 3.7, that these systems are often too complex for many end users in the industry. Consequently, administrators in Norwegian ports resort to more primitive methods like Excel spreadsheets, and some larger ports have developed their own systems. This complexity prevents information sharing between ports and potential clients.

3.5 Desired Solution

The goal of the proposed solution is to simplify data display and updates in ports. It aims to provide port workers with an intuitive map client tailored to their needs. The system is primarily designed for port management professionals, but since it is open source it is free to use by private users. The key objective has been to lower the usage barrier, ensuring the system is accessible and user-friendly, thereby attracting a wider user base.

3.6 Evaluation criteria

The customer provided evaluation criteria in the form of the main business goals listed in Table 4. For the technological solution the customer strongly recommended using Leaflet.js for the frontend part of the 2D map client due to its simple and editable nature. The customer conveyed that further criteria could be developed in compliance with user interviews and during the exploration of necessary features for the system.

3.7 Market investigations

Following the project's initial presentation the team members did research related to the project and tried to comprehend how the end product should function. From this research the team recognized the need to understand real-world use cases and user priorities. To get this information we conducted three hour long user interviews, one during sprint 2 and two during sprint 3, with representatives from Kartverket, Port of Kristiansand, and Port of Arendal, all men aged between 45-60. These participants, provided by the customer, also underwent usability testing, detailed in Section 10.4.

3.7.1 Interview 1: Kartverket

The first interview was with a representative from Kartverket on the 18th of September. At this point in the development process the team still lacked some general understanding about the project. The representative from Kartverket elaborated in more detail about the real-world issues

the project should solve, and the background for these issues. This clarification was extremely valuable for producing new requirements, and founded a solid base for further development.

The most valuable takeaway from the interview was that the system had to be easy to use. This was justified by the fact that most users of the system are older and have limited technical knowledge. The system needed more intuitive navigation and buttons to function well. Our focus therefore became more directed on creating a user friendly and easy to use system, rather than implementing a lot of complicated features.

3.7.2 Interview 2: Port of Kristiansand

The second user interview was held with a representative from Port of Kristiansand on the 13th of October. This interview was held during sprint 3, and the team had at this point developed a more general understanding of the problem and solution space.

The representative was highly experienced with sea- and ship-related topics, due to his 40 year long experience in the industry. He gave the team a historical view of the systems used in the industry, and emphasized the importance of capturing data to update the current systems. The data displayed in these systems have to be accurate, due to low margins related to keel-clearance and navigation. The representative also emphasized that properties and data in these systems must be quality checked before being added, but this should only be necessary for data that is highly important. Simpler data points like fenders or bollards should be easy to edit. This conversation uncovered the desire for a feature where the user can set the status of an object on the map. This could prove to be an important feature to visualize broken or out-of-order objects.

3.7.3 Interview 3: Port of Arendal

The third user interview was held with a representative from Port of Arendal on the 16th of October. The representative had 25 years of experience dealing with port management and data, and was very interested in the project. The focus of the interview was to identify use cases for the system. The representative imagined that updating the status of the objects in the port could be helpful not only to larger ships, but also to private parties looking for the same information. Another valuable request was to implement an edit-mode for the user. The idea was that the user could make several changes, compared to a single change, to the map and then save the changes when exiting edit-mode. The changes made to a map should also be added to a log, to identify which user have altered information in the system. A comment field was also suggested to have a dialogue with other coworkers.

The representative stressed the need for a toolbar list of individual map objects in the system, highlighting its importance for ease of navigation, especially when objects are closely positioned. Additionally, he underscored the significance of maintaining the system's simplicity to accommodate the limited technical skills of end users. Simplifying the system would also facilitate broader distribution, as it lowers the threshold for user qualifications, thereby enabling a wider user base.

3.8 Technological solution

In order to create a satisfying response to the needs of the desired solution, the team decided on some key technologies that would be valuable to the development of the product.

3.8.1 Leaflet

Leaflet is an open-source JavaScript library for interactive maps that was strongly recommended by Norkart. Its strengths lies in simplicity and performance, allowing developers to easily integrate dynamic maps on their website. The support for GeoJSON and its extensive documentation made it ideal for the scope of our project.

Since Leaflet has been at the center of the development, it is natural to mention how it could limit the application. As the application needs to display large amounts of data, and Leaflet renders these objects on the client-side, there is a need to limit the amount of data visible to the user. If not, the application would run poorly on weaker computers. The final product is also dependent on third-party packages that are plugins to the existing Leaflet functionality, meaning the project is relying on these packages being maintained to work with future versions of Leaflet.

3.8.2 NGIS-OpenAPI

The application communicates with NGIS-OpenAPI in order to load and store geographical data. The data retrieved from the API is given in GeoJSON format, making it compatible with Leaflet without much modification. The API is also well documented, with clear information on the available endpoints, and the parameters needed for a valid request. The validation schemas are also sourced from the API, and are used in conjuncture with Ajv - an open source JSON schema validator - to secure valid input from the user. This implementation makes the validation process itself dynamic in terms of changes made to the schemas.

3.8.3 Proxy

Due to the issue of data ownership discussed in Section 3.2 and the exclusion of user authentication from the project scope, a way was needed for users to access the data from NGIS without credentials being leaked. Our solution for this issue was to create a proxy that communicates directly with NGIS-OpenAPI through the provided credentials, and forwards the data to the frontend client, without the API credentials ever being sent between the frontend and proxy.

We decided to create this proxy using Node.js and TypeScript to have have the same programming language as the Leaflet map client. The express.js framework was also used, because it makes writing web servers with Node.js easier, and we wanted to spend minimal time on the proxy component.

3.8.4 TypeScript

TypeScript is a superset of JavaScript that adds static typing to the language, providing enhanced tooling for developers. The decision to use Typescript in our project was due to the improved code quality, maintainability, and the opportunity for early error detection over regular JavaScript. By explicitly defining variable types and interfaces, Typescript enhances the predictability and robustness of the codebase.

We decided not to use any framework such as React on top of TypeScript, as it was not seen as necessary at the time. The lack of framework could pose some trouble as the application grows in size however. Managing a large codebase without the organizational structures provided by a framework can become a complex task, and lead to more overhead. On the other hand, using no framework has its benefits in terms of better performance and more control.

3.8.5 Vite

Vite is a fast and efficient build tool for modern web development. Leveraging the power of ES modules, Vite significantly speeds up the development process by offering fast cold server starts. The efficient handling of dependencies and seamless integration with Typescript makes Vite a suitable choice for our development environment.

By utilizing Vite, we aim to improve the project's build speed, reduce development bottlenecks, and provide a smoother experience for developers working on the codebase. The decision to incorporate Vite aligns with our commitment to leveraging cutting-edge tools that enhance developer productivity

4 Development Methodology

Development methodologies are structured project management approaches for tasks like software development. They ensure efficient planning, product quality, risk management, and collaboration. The right development methodologies are chosen based on the project's nature, and choosing the right methodologies is a critical factor in software development (Sommerville, 2016). We adopted a hybrid approach by integrating elements from various development methodologies, such as Gitflow, Scrum, Kanban, and Extreme Programming.

4.1 Gitflow

In our project, we opted for the Gitflow branching model due to its suitability for our development process. Gitflow is a Git branching framework that organizes work into distinct branches for features, releases, and ongoing tasks (Atlassian, n.d.-a). This structure offers a well-organized approach to version control, aligning closely with our project's needs.

4.1.1 How We Implemented Gitflow and Its Benefits

At the core of Gitflow are two primary branches: **main** and **develop**. The **main** branch stores the official release history, while **develop** serves as the integration point for new features before their official release. This arrangement provides a clear and structured version control mechanism.

One of the standout features of Gitflow is its use of **feature branches** for the development of new features and non-emergency bug fixes. These branches keep work separate from the main codebase until it's ready for integration, promoting a clean and organized development process.

When it's time to prepare for a release, Gitflow facilitates this with **release branches**. These branches are created from the current state of the **develop** branch, and after final preparations, they are merged into both **main** and **develop**.

4.1.2 Choosing Gitflow

Gitflow, with its structured approach to version control and release management, was our choice after careful evaluation. While it introduces some complexity and overhead, it aligns well with our project's feature and release management needs. Despite a learning curve, we believe Gitflow's benefits outweigh the challenges.

It's worth noting an alternative approach: trunk-based development. This model streamlines merging directly into the main branch, fostering collaboration and enabling rapid iterations, crucial for continuous integration and delivery (Atlassian, n.d.-b). Unlike Gitflow's structured branches, trunk-based development assumes the main branch is always stable.

4.2 Scrum

Scrum is an agile project management framework designed for flexible and iterative software development. It operates on principles of transparency, inspection, and adaptation, dividing projects into short sprints of one to four weeks. This approach prioritizes collaboration, adaptability, and continuous improvement, making it ideal for projects with evolving requirements and a need for frequent stakeholder feedback (Kniberg, 2007).

4.2.1 Why Scrum?

We selected Scrum for its industry recognition, adoption in software development, and team familiarity. The client's preference for Scrum, as mentioned in the project assignment appendix (section F), supported effective communication and teamwork. Scrum's adaptability was key to handling our project's evolving needs, offering a customer-centric approach that contrasts with the more rigid waterfall model (GeeksforGeeks, 2023). Regular customer engagement through Sprint Reviews helped align the product with client expectations. Scrum's structured meetings like daily stand-ups and retrospectives ensured efficient communication and problem-solving within the team.

4.2.2 Drawbacks of Scrum

Despite Scrum's benefits, it presents challenges such as the need for extensive team collaboration and time commitment, which can be difficult with other obligations. Customer involvement, a cornerstone of Scrum, can also be challenging but was advantageous in our case, as the customer was actively involved, enhancing communication and project alignment.

4.2.3 Product Backlog

The product backlog is a central part of Scrum, which serves as a list of requirements, stories, and features (Kniberg, 2007). We used Github's issue system to create backlog items with the following fields:

- **ID**: Unique identifier for each issue, added automatically by Github.
- **Name**: Short description explaining the issue's content.
- **Description**: Longer, detailed description for clarity on the issue.
- **Assignee**: Tracks who is working on each issue.
- **Labels**: Tags for classifying the issue type (e.g., new feature, documentation, bug fix). Full list available **here**.
- **Priority**: Assigns a level (Low, Medium, High, Urgent) to each issue for work prioritization.
- **Size**: Initial work estimate needed for issue resolution.

4.2.4 Sprints and Sprint Planning

During the project planning phase, we decided on five sprints, each lasting two weeks. This decision was influenced by various factors, including allocating one week for the course kickoff and initial planning, which set our timeline. We determined that one-week sprints were too short and anything beyond two weeks too long for our needs. Allowing two weeks at the end for report writing and presentation preparation, we had a ten-week window for development, fitting in five sprints. The project schedule is detailed in Figure 1.

Before each sprint, we held efficient sprint planning meetings using GitHub Projects. These meetings, kept under an hour, involved revisiting the sprint goal, creating new issues, and estimating their scope and complexity to determine the sprint workload. We carefully selected issues for the sprint backlog, considering the sprint's goal and our capacity, and set milestones for each sprint to track planned tasks. This structured approach ensured efficient teamwork and clear planning for each sprint.

4.2.5 Time Estimation

We made a deliberate choice to use relative sizing, utilizing the predefined sizes available in GitHub Projects. This was done for estimating the time required for tasks, as opposed to traditional numeric values like the Planning Poker sequence. This decision was rooted in both our team's needs and preferences.

The adoption of words or labels such as *Tiny*, *Small*, *Medium*, and so on, offers several benefits over specific numeric values. These labels are more accessible and less prone to confusion, as they allow team members to intuitively understand the size of the task without needing to remember precise numeric meanings.

However, relative sizing has drawbacks like potential imprecision, subjectivity, and challenges in producing burndown charts. Given the project's complexity and the need to grasp new concepts, we prioritized simplicity in our estimations. Relative sizing aligned with our team's learning curve and project objectives.

4.2.6 Daily Scrum

Acknowledging our roles as students balancing multiple courses, we understood that dedicating eight hours a day to project development was unfeasible. Consequently, daily stand-up meetings held each day seemed impractical. To optimize our time and efforts, we opted for a more flexible approach. We conducted daily stand-ups at the onset of our student and supervisor meetings, specifically on Mondays and Wednesdays. During these daily scrum sessions, we consistently addressed the three questions (Kniberg, 2007):

- What progress have you made since the previous meeting?
- What tasks are you currently working on?
- Are there any challenges or obstacles hindering your work?

To overcome the challenge of team members working on Tuesdays, Thursdays, and Fridays, and wanting to share progress without daily in-person meetings, we implemented a digital scrum solution. Using a dedicated Slack channel called "daily scrum" and a Slack bot with a daily reminder

message, team members could easily update their progress outside of scheduled meetings. This streamlined communication, reducing the need for unnecessary in-person meetings, especially on days with comprehensive updates during scheduled sessions.

4.2.7 Scrum Master

Initially, we considered a rotating Scrum Master role but realized the advantages of consistency. Maintaining a consistent Scrum Master throughout the project reduced the learning curve and eliminated the need to track role changes each sprint or week. Consequently, we decided to appoint a dedicated Scrum Master to ensure stability throughout the project.

The Scrum Master's role included facilitating and coaching to ensure effective implementation of the Scrum framework. They organized and led key Scrum events like sprint planning, daily stand-ups, sprint reviews, and retrospectives. Additionally, the Scrum Master addressed impediments, fostering a collaborative and productive work environment.

4.2.8 Sprint Review

The Sprint Review is crucial for the scrum team and stakeholders to inspect the product, gather feedback, and adapt (Kniberg, 2007). After each sprint, we conducted a review with the customer. We presented completed and incomplete product backlog items, explaining reasons behind incompletions or addressing additional items. Discussions included what went well, challenges faced, strategies employed, and a live demo of completed work. Customer feedback shaped plans for the next sprint, and we revisited our project timeline to align with the team's capacity.

4.2.9 Sprint Retrospective

In our Sprint Retrospectives, we assessed the sprint's performance in terms of contributions, interactions, processes, tools, and adherence to the definition of done (ScrumAlliance, n.d.), marking the sprint's conclusion. We structured our retrospectives for efficiency, preparing an agenda in advance and using RetroTool for taking notes. Discussions encompassed 'Liked', 'Learned', 'Lacked', and 'Longed for' aspects.

We began with individual contributions to the 'Liked' section, allowing three minutes per person. This format was repeated for the other sections. Following this, a 20-minute round-robin discussion allowed team members to elaborate on their points.

We grouped similar issues for clarity and then conducted an anonymous vote, with each member allocating three votes to prioritize improvements. The top-voted issues were then used to formulate action points in a 30-minute session. Finally, a 5-minute summary captured key takeaways for the upcoming sprint.

4.3 Kanban Board

To help visualize our workflow, we utilized Github Projects' Kanban board. This Kanban board serves as a visual project management tool, allowing us to track the progress of work items as they move through different stages in our workflow. It plays a vital role in helping our teams visualize, organize, and efficiently manage tasks, enhancing transparency in our project management (MiroBlog, 2023).

Within this Kanban board, we have categorized our issues based on their status, with each status representing a different stage in our workflow. These status categories include:

- **Backlog:** Holds product backlog items not in the current sprint.
- **Blocked:** Indicates current sprint issues awaiting completion of others.
- **Ready:** Issues prepared for work but not yet in progress.
- **In Progress:** Actively worked on issues.
- **In Review:** Issues undergoing code review for quality assurance.
- **Done:** Issues completed and merged into the develop branch.
- **Rejected:** Issues that cannot or will not be implemented.

4.4 Why Kanban?

Kanban offers valuable visibility into task status, aiding teams in identifying bottlenecks for enhanced efficiency (Rehkopf, n.d.). Unlike Scrum, a Kanban board does not reset for every sprint, providing greater flexibility for reprioritization and updates (Rehkopf, n.d.).

However, this flexibility can pose challenges. Projects with strict deadlines may find Kanban's lack of fixed timeframes and sprint commitments challenging. The absence of structured planning horizons and less emphasis on backlog prioritization might affect task sequencing. Without defined roles and Work in Progress (WIP) limits, there's a risk of inefficiency and burnout. The adaptable nature of Kanban may also dilute team accountability, a concern we address in the Extreme Programming section (MiroBlog, 2023).

The integration of Scrum with Kanban allowed us to leverage Scrum's strengths in structured project management, defined roles, and clear planning horizons. Simultaneously, the Kanban board's flexibility accommodated changes seamlessly, enhancing our adaptability and responsiveness to evolving project needs. This combined methodology optimized our team's performance and project outcomes.

4.5 Extreme Programming

We enhanced the Scrum methodology by incorporating Extreme Programming (XP) practices. XP is a unique agile software development framework with a dual mission: the creation of higher-quality software and an elevated quality of life for the development team. XP offers a highly specific set of guidelines for software development practices, setting it apart from other agile methodologies. It places a greater emphasis on development practices rather than project management, serving as a valuable supplement to Scrum (McDonald, 2023).

4.6 Extreme Programming Practices

Sit Together: Given that communication is a core value of XP and that face-to-face conversation is widely recognized as the most effective form of communication, it is recommended to have your team work closely together in a shared workspace (McDonald, 2023). To put this into practice, we opted for a work setup where we sit together and collaborate on Mondays and Fridays, as detailed in Section 2.1.

Pair programming: Pair Programming involves two programmers collaborating at the same computer to jointly develop software. It leverages the power of two minds and four eyes to facilitate continuous code review and resolve issues quickly, resulting in higher quality code without doubling the time spent (McDonald, 2023). Our team tried to embrace the idea of utilizing pair programming to the fullest extent, implementing it naturally whenever the opportunity arose.

Continuous Integration: Continuous Integration is a software development practice where code changes are immediately tested as they are added to a larger codebase. This approach helps catch and resolve integration issues sooner, making the development process more efficient and reducing the complexity of identifying problems during integration (McDonald, 2023). See section 10.2 for more information about our unit testing.

Collective Code Ownership: Collective Ownership in software development means that every team member is authorized and encouraged to make changes to any code file, promoting collaboration and shared responsibility (Agile Alliance, 2023). To put this into practice, we ensured that every team member had access to all project files, and promoted open communication, allowing team members to voice their preferences for tasks and change their assignments if needed.

Coding Standards: Coding Standards are a set of shared guidelines ensuring that all code within a system maintains a consistent and familiar appearance, promoting collective ownership (Altexsoft, 2021). Recognizing the importance of this, we established a set of coding standards at an early stage, employing tools like *ESLint* and *prettier* to enforce these rules. For a more comprehensive description of our coding standards, please refer to Section 2.4.

4.6.1 Why Extreme Programming

Integrating Extreme Programming (XP) with Scrum posed challenges. Adapting to XP practices, like pair programming, within Scrum's structure demanded extra resources and schedule adjustments. Team members, especially those less familiar with XP, faced a learning curve impacting productivity. Despite potential risks, we believed embracing XP practices would enhance our development processes and contribute to project success.

The integration of XP practices into the Scrum framework is motivated by the pursuit of a powerful synergy. XP's focus on software quality and team well-being complements Scrum's iterative progress and customer-centric approach. This integration strengthens team communication and cohesion through practices like sitting together, enhances issue resolution and code quality via pair programming, promotes early issue detection with continuous integration, emphasizes shared responsibility with collective code ownership, and maintains code consistency through coding standards. The result is higher-quality software delivered efficiently, aligning with the core principles of both methodologies.

5 Innovation

This section is dedicated to the more current topics in software development; Sustainability, Diversity, and Artificial Intelligence (AI). The team had several dedicated sessions were the sole function was to identify the related problems to these topics. This was done in consultation with both the customer and the supervisor. The direct consideration of these topics in a software development process was somewhat new to both the team and the customer, but very thoughtful and interesting.

5.1 Sustainability

Sustainability is defined in (Brundtland, 1987) as the utilization of existing resources without compromising future generations. Sustainability has gained focus over the past decades. Condori-Fernandez & Lago (2018) mentions four dimensions within sustainability: social, technical, economical and environmental. All of these four dimensions contribute in making software development processes more sustainable. It is important to discover and explore the challenges surrounding the project's sustainability early on, and thus make it easier to improve and consider the best solutions for the project.

5.1.1 Focus within the project

The sustainability aspect of the project was highlighted during the user interview with Kartverket. Kartverket's representative presented valid reasons for why sustainability is an important aspect of the project, and the team discussed further how this focus could be implemented in the best possible way. For example, it was specifically emphasized that the increased cargo capacity of a ship, due to more accurate depth data in Norwegian ports can allow ships to take fewer round trips. Another reason was that the current method to park a ship in the port often revolves around exchanging hundreds of e-mails between ports and ships. The inclusion of a system that handles this problem will increase social, environmental and economic sustainability.

Later in the development process the team held a dedicated sustainability meeting, identifying sustainability related aspects. These aspects are categorized into the four dimensions of sustainability. It was important for the group to address all four dimensions of sustainability, as earlier research has shown that projects often focus on social and technical sustainability, but lack consideration about economic and environmental sustainability (Capiluppi & Jaccheri, 2023).

5.1.2 Social sustainability

Social sustainability includes all factors that influence the social and cultural identity of people (e.g. health and comfort). It also addresses the balance between conflicting interests and how people assess their environment (Pham et al., 2020). The team emphasized the significance of security requirements such as confidentiality, authenticity, and accountability, ensuring equal and equitable data access while preventing unauthorized access. Trust, usability, and user satisfaction were also considered essential contributors to social sustainability. This relates to the feeling of comfort, where a safe system prevents any discomfort by being hacked or exposed to a security threat.

5.1.3 Technical sustainability

The technical dimension refers to maintenance, resilience, evolution and and the ease of transition of artificial systems (Pham et al., 2020). The team also emphasized the importance of functional correctness, availability, and interoperability, with the latter being crucial for software reuse and interaction with other systems. Maintaining code standards, using version control systems like Git, and emphasizing modularity and testability were identified as key practices for technical sustainability. The team also pointed out that in hindsight, the project would have been more technical sustainable with a frontend framework like React. The complexity of the project is not very high at the moment, but for further implementation a framework would streamline some of

the low level implementation details, such as imperatively manipulating HTML DOM-elements that is currently necessary.

5.1.4 Economical sustainability

The economical sustainability is reflected by the degree to which life cycle costs are minimized, economic efficiency is improved and capital and product value is maintained (Pham et al., 2020). The team recognized effectiveness, reusability, and resource utilization as vital contributing factors. By optimizing operations based on data depth, the user of the system can load cargo ships more efficiently, prevent accidents, and enhance planning, thereby maximizing resources and reducing waste.

5.1.5 Environmental sustainability

The environmental sustainability covers the protection of the global and local ecosystems, and saving natural resources. This includes issues ranging from immediate waste production to energy consumption (Pham et al., 2020). The team acknowledged resource utilization, maintainability, environmental risk mitigation, and time behavior as critical factors.

The aforementioned improvement to cargo loading will also reduce the amount of round trips needed, thereby reducing emissions and improving time efficiency. In addition, our solution will remove the need for using emails with PDF as the method for sharing port information. According to our customer, these email threads can be extensive and reach up to 300 emails. Replacing this with an up to date map client will reduce the amount of data transferred and save a lot of time.

5.2 Diversity

Diversity in the field of computing is crucial for fostering inclusion and representation across various dimensions, such as race, gender, beliefs, geographical origin, age, abilities, and more. This inclusive approach has gained traction in software development, as evidenced by the positive impact diversity can have on teams and the software development process (Google, 2022).

5.2.1 Diversity in the process: Team dynamics

In our group, we recognized the significance of diversity and strove to incorporate it into our team dynamics. The balanced representation of men and women in our team was predetermined to ensure gender diversity. This intentional composition aligns with industry trends, where companies actively seek diversity for its numerous advantages (Microsoft, 2023).

Software engineering is a male-dominated industry, which makes it important to not overshadow the female minority. Even in gender-balanced groups like ours there is a trend for women to be assigned more traditional, non technical work, such as project manager, secretary, or writer (Hirshfield & Koretsky, 2017). Therefore, we payed extra attention to diversifying the project roles (see Table 1) such that both male and female opinions would be heard equally in all aspects of the project. Notably, we assigned the role of software architect and UX manager to women, which have been significantly male-dominated roles for TDT4290 in previous years (Nguyen-Duc et al., 2019).

During Sprint 4, the team experienced major absences from two team members, one due to planned

surgery and the other due to unexpected health issues. Despite a temporarily reduced workforce, the team adapted by facilitating for digital participation and remote work, ensuring continued productivity.

The decision-making process for both the role assignments and adapting to health challenges was characterized by open discussions and mutual agreement. Our commitment to diversity extended beyond gender, encompassing a supportive work environment that valued individual contributions, regardless of background or perspective. Overall, we felt like our explicit focus on diversity contributed to an inclusive work environment, which promoted a wider range of perspectives, stronger innovation, and faster problem solving.

5.2.2 Diversity in the Product: Usability

It is important to consider the inclusiveness of software that aims to support diverse people in problem-solving situations. The users who tend to be best supported by problem-solving software are those who are best represented in software development teams, e.g. relatively young, able-bodied, males (Burnett et al., 2016). Since the potential users of our product is mostly the older generation, it was incredibly valuable to perform usability tests as described in Sections 3.7 and 10.4 on this demographic. We realized that diversifying the usability test subjects further, such as including women, different nationalities and diverse age groups, could have given us even more nuanced views of the product's usability. However, given the limited time and resources of the project, it was not prioritized to find more a more diverse group of usability test subjects.

5.3 Artificial Intelligence

AI has evolved from something people hardly know, to a tool used for various tasks in a normal working day. The recent introduction of ChatGPT and other large-language models (LLM) have changed the way people solve problems. The inclusion of these tools in software development have also expanded widely, since research has shown that productivity can increase when using these tools (Kalla & Smith, 2023).

5.3.1 Customer demands

The customer had a proposal regarding AI in our project, specifically machine learning, to help the user extract relevant data and make smart suggestions in the software system. This part of the project was not initiated, as it was clear that the customer wished us to prioritize other requirements first.

5.3.2 Usage of AI tools

On the other hand, the team has used AI in the development process, by using ChatGPT to troubleshoot problems and generating code snippets. The use of ChatGPT in the project was encouraged by the customer, to experience how useful this tool really is. The team has experienced that ChatGPT can increase productivity by providing solutions to problems related to coding. Making use of this approach enabled us to develop new skills and concentrate on tasks of higher relevance, moving away from more mundane, low-level activities. This is documented as one of the main beneficial aspects of integrating AI into software development (Satell, 2019).

However, some drawbacks were found as the application grew in size, since ChatGPT required more knowledge about the remaining code and dependencies in order to be effective. This led to long and complicated prompts, which created errors. The team resorted to mainly using ChatGPT as a debugging tool and search engine to maximize efficiency.

The team did not use Github Co-pilot, due to the fact that team members did not want to have code written for them. The team made a decision early on to try to do all the coding by themselves, and not utilize AI tools directly in an IDE for the production of code, only for troubleshooting and coming up with general ideas. This was done to increase the learning aspect from the project. In hindsight this could have boosted the teams productivity, but it was considered more valuable to learn the programming language and acquiring new knowledge than maximizing the productivity.

6 Requirements Specification

6.1 Functional Requirements

The functional requirements for the project are presented in Table 5. These were derived from the business goals presented in Section 3.1, as well as the user interviews described in Section 3.7.

Table 5: Functional requirements for the project

ID	Related BGs (4)	Requirement	Priority
FR1	BG2, BG3	NGIS-OpenAPI credentials must be hidden through a proxy	High
FR2	BG2	Collected geographical data must be visualized on a web based interactive map	High
FR3	BG2	Users should be able to view details about the data by clicking on their locations on the map	High
FR4	BG3, BG4	Users should be able to edit details of the data, such as their status	High
FR5	BG3, BG4	Users should be able to edit geometry (location) of points and lines on the map by dragging them interactively	Medium
FR6	BG1, BG2	Objects on the map should be displayed with official symbols based on the object type	Medium
FR7	BG2	Use the Havnedata WMS and depth WMS to display read only data without need for credentials	Medium
FR8	BG2, BG3, BG4	Users should be able to create new objects and store them in NGIS through a easy-to-use form	Medium
FR9	BG1, BG4	The application should give detailed feedback to the user in case something goes wrong	Medium
FR10	BG1, BG3	Users should be able to enable an aerial photo layer on the map to more accurately place objects	Low
FR11	BG1, BG4	The map should clearly mark objects that are not in use or operational	Low
FR12	BG2	Data displayed on the client should be portioned to enhance performance	Low

6.2 Non-functional requirements

Non-functional requirements, also known as quality attributes, are used to indicate how well the system satisfies the stakeholders needs beyond the basic function of the system (Bass et al., 2021, 39). Quality attributes often specify some measurable or testable property of a system, often concerning properties like availability and performance. The important quality attributes for this project are described below.

6.2.1 Usability

Usability is a quality attribute concerning how easy it is for users to accomplish desired tasks, and how the system supports the user in doing the task (Bass et al., 2021, 197). Early on in the project, the group had a meeting with Kartverket (described in Section 3.7.1). During this meeting it became clear that usability had to be considered one of the most important quality attributes for the project. One of the main requirements for this system is to be designed with the end user in mind. The system is supposed to make the process of updating geographical port data easier and more accessible, also to those who are not that familiar with technology. A lot of the end users are older people who are not that familiar with new technology, and may be opposed to start using new and modern solutions. It is therefore critical that the system is developed and designed with usability in mind.

Related business goals: BG1, BG3, BG4

6.2.2 Modifiability

Modifiability is a quality attribute that is all about how easy it is to make changes to a system (Bass et al., 2021, Chap. 8). Our product is a part of a bigger project, and will be further expanded in the future. It is therefore important that the system is built with modifiability in mind to ease the costs of change, and to make it easier for different development teams to understand and work on the project.

With this in mind, it is important to make the code modular to isolate potential bugs. Using third-party libraries can also make the software more modifiable, since there is less cost in adapting existing solutions than writing everything from scratch. In our case it would be totally infeasible to fulfill the business requirements of the project without using Leaflet as a third-party library for implementing the interactive 2D map. Finally, it is important that the code follows clear standards and has consistent formatting.

Related business goals: BG6

6.2.3 Security

The security quality attribute tackles the system's ability to protect data and information from unauthorized access while still serving this data to the authorized users (Bass et al., 2021, 169). During meetings with the customer, it became clear that input validation is an important requirement for the system. Input validation is also an important measure to ensure the availability and usability of a system, but in the context of the customers requirements, this is a security requirement. The security of the system will be further discussed in 9.

Related business goals: BG7, BG8, BG9

7 Architecture

This section presents the software architecture of the system. The architecture helps describe the different components of the system, their relations and communication between them. Creating an architecture for a system is intended to help plan and develop the system. In the section below, the implemented architectural tactics and patterns are presented, as well as different architectural views describing the architecture of the system.

7.1 Architectural Tactics and Patterns

Architectural tactics and patterns are techniques used in software architecture to achieve required quality attributes and to solve common problems often encountered in software development. All tactics and patterns mentioned in this section are taken from *Software Architecture in Practice* by Bass et al. (2021).

7.1.1 Tactics

Split module tactic
A module should only handle responsibilities that are cohesive. If the module handles multiple non-cohesive responsibilities, the cost of modification will be higher. To increase cohesion and reduce costs, modules handling multiple non-similar responsibilities should be split. This increases the modifiability of the system.

Restrict dependencies
Restrict which dependencies a module can have. This is to reduce the cost of change, as the probability of a change influencing multiple modules is reduced.

Use an intermediary
By inserting an intermediary between two modules, the coupling and dependencies between them are reduced and modifiabilty is improved.

Validate input tactic
This tactic tackles the validation of input data. The customer has made it clear that input validation and sanitation of data is an important feature of the system. By validating input, the possibility of system failures are reduced and increases the systems uptime. Input validation is also an important security feature, and acts as a basic defense against attacks. Thus, this tactic improves both the availability and security of the system.

Exception handling
Detected exceptions should be handled in some way to prevent the program from crashing. Handling exceptions improves the availability of the system, but also improves the usability of the system by providing feedback to the user about the system's state.

7.1.2 Patterns

Client-Server Pattern
A common pattern where there is a server which serves request for multiple clients simultaneously. This pattern removes the coupling between different clients, such that clients are not aware of other clients being served at the same time. Furthermore it is possible to develop and evolve the client-side and server-side of the product independently of each other as long as the interface

between them stays the same. This partitioning of the system prevents changes from propagating through the whole system, which greatly improves the modifiability.

7.2 Architectural Views

Architectural views are used to describe the software architecture of a system in smaller chunks, and to expose how the architecture handles the different quality attribute requirements. For this architecture, the 4+1 View Model as described in Kruchten (1995) is used for documentation. This view model consists of four main views, the logic, process, development, and physical view, as well as an extra scenario based view. In the following section, the scenario view has been omitted.

7.2.1 Logical View

The aim of the logical view is to model how the system answers to the functional requirements of the system by modelling how different functionality and responsibilities are assigned to various modules. Figure 5 shows the overall structure of the system. Data is retrieved from NGIS-OpenAPI which is an already established API for geological data in Norway. Instead of having each client connect directly to the NGIS-OpenAPI, client requests are directed through a proxy. The proxy works as an intermediary between the clients and the API, reducing coupling and dependencies between them, as well as hiding API-access credentials.

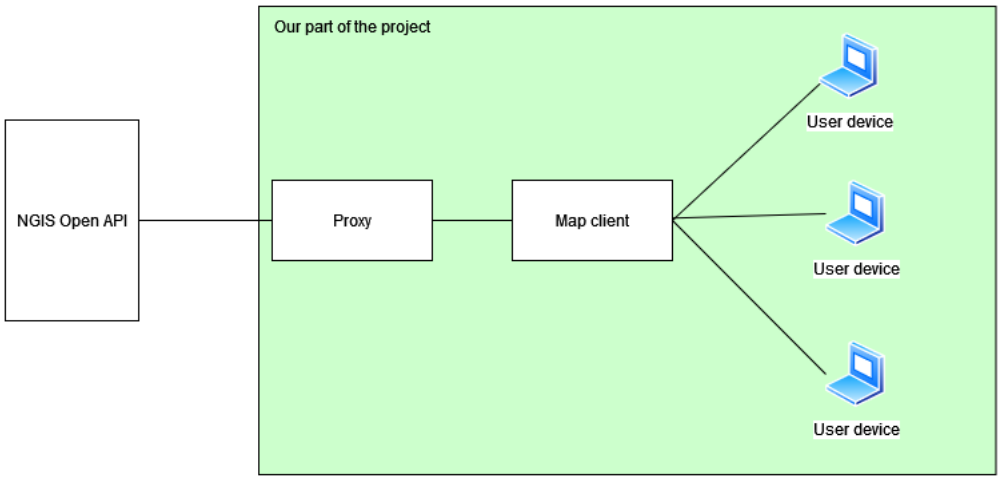


Figure 5: Simple overview of the system

The diagram in figure 6 outlines a simplified overview of the client part of the system. There are two kinds of modules here, components and util/helping-modules. As can be seen in the diagram, the components are not dependent on each other with the exception of the createFeature- and featureDetails-components which makes use of the multiselect-component. Not all of the components have dependencies with all of the util-modules. The dependencies between the components and util-modules have been simplified to make the diagram more readable.

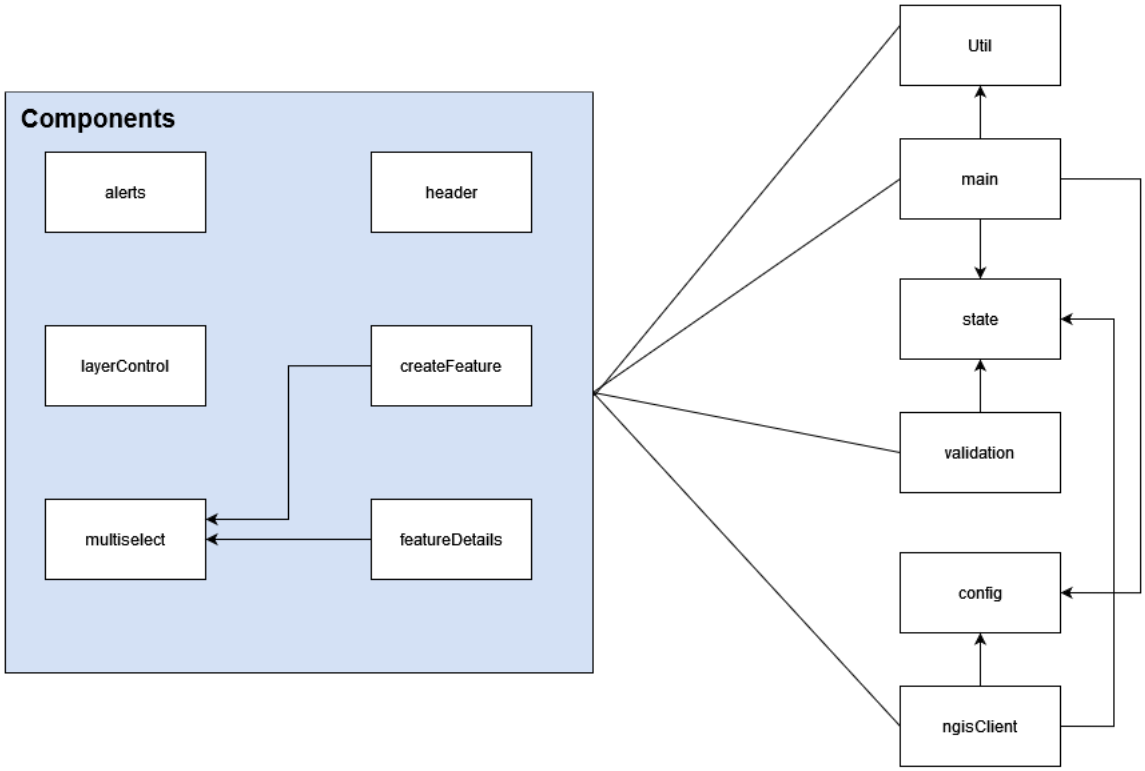


Figure 6: Logic view: diagram for the client-side of the system
The arrows indicate method calls from the originating module to the target module, while the lines show mutual calls between connected modules.

7.2.2 Process View

The process view aims to capture the run-time aspects of the system, and handles non-functional requirements like performance, availability and concurrency. The sequence diagram in figure 7 shows how the system communicates with NGIS-OpenAPI. The diagram models a scenario where the user adds a new feature (map object) to the map, and how the system handles this request. When the user creates a new feature, the createFeature-module calls the *putFeature()*-method from the ngisClient-module. This method sends a request over HTTP to the proxy, which relays this request to NGIS-OpenAPI. When the proxy receives a response from the API, it sends this to the ngisClient-module which can send the response to the createFeature-module. Once createFeature has confirmed that a response has been received, an update message is shown to the user. All communication with NGIS-OpenAPI happens in this manner: A module that wants to interact with the API calls a method in the ngisClient-module, which then sends a HTTPS request to the proxy, and so on. Having one module handle all of the external communication improves the system's modifiability as it reduces the number of dependencies in the system, instead of having multiple external dependencies to an external element.

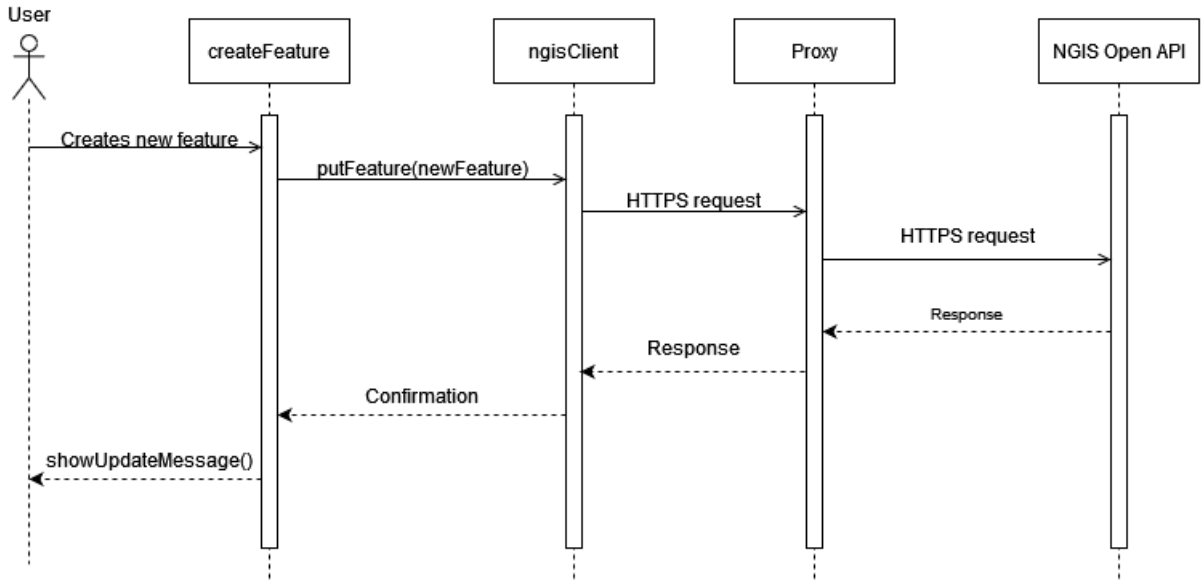


Figure 7: Process view: sequence diagram for creating a new feature

7.2.3 Development View

The development view, depicted in Figure 8, organizes software modules for the development environment, aiming to streamline system development and task allocation among developers. This view presents a layered module structure, clarifying dependencies: modules in upper layers rely on those in lower layers and cannot be fully developed until their dependencies are completed. Modules on the same layer, however, can be developed at the same time. For example, the *util* and *config* modules will evolve throughout the project, with functionality added as needed. Dependencies within the same layer are also indicated by lines between modules.

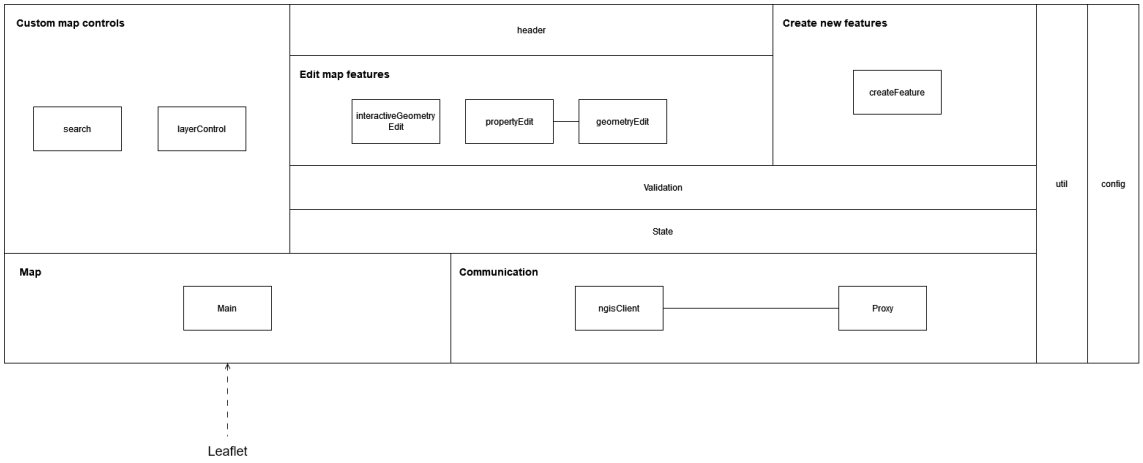


Figure 8: Development view of the system

7.2.4 Physical View

A software system executes on a number of different processing nodes. The physical view aims to allocate the different parts of the software onto these nodes. The physical view in Figure 9 outlines the physical aspect of the system. The system can be divided into two parts: The client part runs

on the users' own devices in a web browser, while the proxy runs on a server. The last node of the system is the NGIS-OpenAPI, with which the proxy communicates. The NGIS-OpenAPI retrieves data from a central database for Norwegian port data, but this is not modelled in the diagram as the details of this process is not relevant to this project. All communication between the nodes happens over the internet using the HTTP protocol.

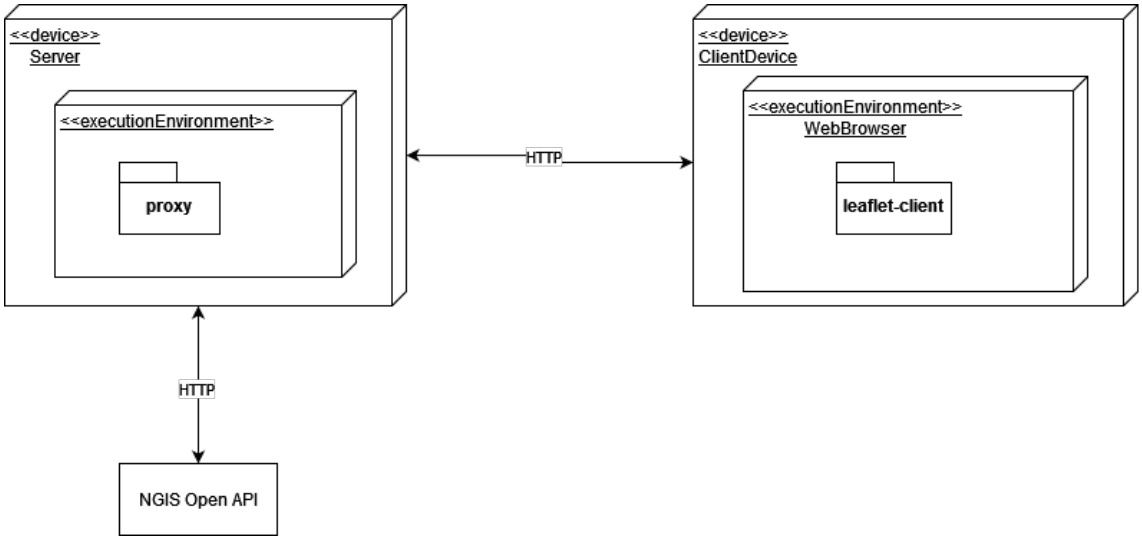


Figure 9: Physical view of the system

7.3 Issues

The application makes use of CSS to style the user interface and to create custom HTML components. The different CSS stylesheets are not represented in any of the architectural views. The use of CSS in the application has an important impact on the modifiability and sustainability of the code. The CSS often creates unintended coupling between components as styles are applied in a global scope. This means that changes in the styling of one component may have unintended consequences for another component in another part of the system. Especially as the application grew and more components were created and integrated with each other this became more challenging. This issue greatly decreases the modifiability and sustainability of the system as making changes to the system becomes more costly. In the design phase of the project, the group should have created a CSS architecture and agreed upon conventions for how CSS could be utilized in a good manner. For instance agreeing on a naming convention for class names and being more aware of how *CSS specificity* works would have mitigated some of the problems the group faced during development.

8 Sprints

The following sections will go through the planning, implementation, and review of all five sprints we held according to the project schedule (see Figure 1). All backlog issues completed for each sprint can found in the project's Milestones on Github. Throughout the project we also recorded all important decisions (mainly technical) we had to make during the sprints in a decision log, which is shown in Appendix D.

8.1 Sprint 1 - Design

Sprint Goal: Mapping user needs, create mockups, and design the architecture

Duration: 4. September - 18. September

8.1.1 Sprint 1 Planning

During the sprint planning meeting, we created the backlog for the sprint. This can be viewed in Figure 6. Recognizing the importance of having a clear early-stage design representation, we decided to create a Figma mockup. This mockup would not only help the development process, but also allow for usability testing to better understand and map user needs. Additionally, we collectively agreed that if there was enough time we would initiate the setup of the map client, even though this formally belonged to the next sprint. This decision was made to ensure that every team member had meaningful work to engage with, but it was contingent on the successful completion of issue 2 within the sprint timeframe.

Table 6: Sprint 1 Backlog

ID	Issue	Estimate	Related FR/BG	Completed?
#1	Create a sketch of the architecture	M		No
#2	Setup proxy component	M	FR1	Yes
#4	Communicate with NIGS Open API	XL	FR1, FR2, FR3, FR4, FR5, FR7, FR8	Yes
#5	Create a simple Figma mockup	M	BG1	Yes
#9	Create a CONTRIBUTING.md	M	BG6	Yes

8.1.2 Sprint 1 Implementation

As this sprint was a part of the design phase of the project, most tasks were administrative or a form of research. The team experimented a bit with the NGIS-OpenAPI to learn its workings, and gain better understanding of the data it provides. This is reflected in issue #4. During initial conversations with the customer, it was suggested that the group could utilize a proxy component between the frontend, and NGIS-OpenAPI to mainly hide API credential details that were provided for a test server hosted by Kartverket. This proxy would also double as the backend for the project, and the setup of this proxy was prioritized for this sprint. Once the team understood how to communicate with NGIS-OpenAPI, setup for the proxy started and communication with the API through the proxy was established.

A Figma mockup was created. This was both to create an initial design for the user interface for the project, but also to map the team's understanding of the project. These Figma sketches can be found **using this link**. For the administrative part, the group defined coding conventions for the projects. These were outlined in a CONTRIBUTING.md file to make it easily accessible. Other administrative work that is not reflected in the sprint backlog was also completed. For instance, templates for meetings (see Appendix E) and an outline for the report were created.

8.1.3 Sprint 1 Review

While we made good progress towards the sprint goal, it was not fully completed. One primary reason for this was that the team realized that the design phase needed more time and attention. This insight was prompted by our scheduling of a user interview in week 38, the week after the conclusion of the sprint. It also became clear that the team's limited understanding of the project restricted the team's work in this area, and that the design phase needed to be more fluid to align with the team's evolving understanding of the project's requirements. Issue #1 for instance, was not completed during this sprint because the team felt they did not have a good enough understanding of the project requirements to design a proper architecture. This issue was therefore moved to a later sprint. Despite this, the customer confirmed that our understanding of the project aligned with their requirements during the sprint demo meeting. In this meeting, the Figma mockup was presented to the costumer, as this represented our work this far.

Action points

A retrospective was held to conclude this sprint as described in Section 4.2.9. The action points extracted from this meeting are listed below:

- Start each meeting with a 1-minute check-in with team members to foster a better understanding of how everyone is feeling and if they've had any enjoyable experiences since our last meeting. Additionally, consider scheduling a social dinner or lunch to strengthen team bonds.
- Encourage everyone to summarize their daily standup updates in Slack either before or after each meeting. This helps keep everyone informed and aligned.
- Reserve Fridays for technical demos showcasing significant changes. Keep these sessions informal and interactive, and having a screen available would enhance the experience.
- Following the daily scrum, inquire if anyone is interested in pair programming to promote knowledge sharing and collaboration within the team.

8.2 Sprint 2 - Create Map Client

Sprint Goal: Create a map client that displays data from NGIS-OpenAPI

Duration: 18. September - 2. October

8.2.1 Sprint 2 Planning

Before the sprint planning started, the team conducted a user interview with Kartverket. This is described in more detail in Section 3.7.1. After this interview, the team gained a better understanding of the requirements for the project, especially what kind of functionality the project should provide. During the sprint planning meeting, the team's improved understanding made it easier to create relevant tasks. The sprint backlog is presented in Table 7.

Sprint 2 Backlog:

Table 7: Sprint 2 Backlog

ID	Issue	Estimate	Related FR/BG	Completed?
#1	Create Architecture sketch	M	BG6	Yes
#6	Setup frontend map component	L	FR2	Yes
#7	Create a CI pipeline with Github actions	M	BG6	Yes
#10	Setup Lerna monorepo	M		Yes
#11	Set up husky precommits	M	BG6	Yes
#13	Set up testing environment	M		Yes
#27	Display details about features	XL	FR3	Yes

8.2.2 Sprint 2 Implementation

Our primary focus for this sprint was to set up the frontend map client (issue #5). This required us to do some research regarding Leaflet and GIS in general, as it was mostly new for all of us. Luckily, Leaflet turned out to be quite easy to use. The only slight bump in the road was figuring out the right map projection to use with NGIS-OpenAPI, and differing conventions regarding the order of latitude and longitude in different standards, to make the data display correctly with Leaflet. Additionally, we also focused on setting up all infrastructure surrounding our code. This included a CI pipeline with Github actions, a testing environment with Vitest, husky precommits, and a monorepo using Lerna.

8.2.3 Sprint 2 Review

The team managed to complete the goal for sprint 2. The developed map client shows data from the NGIS-OpenAPI, and it is possible for a user to click on a map point to access more information about that point. During the sprint demo, the customer again confirmed that our understanding of the project requirements aligned with theirs. The team also discussed user interview with the customer, and how the team was very pleased with its results. To take this process a step further, the customer suggested that we could meet with representatives from Port of Arendal and Port of Kristiansand as well. These user interviews were then scheduled for sprint 3.

Action points

The action points created during the sprint retrospective are presented below:

- Create agenda for Monday meetings on Fridays and Friday meetings and Wednesday meetings on Mondays
- React to slack message if you will be present at tomorrow's meeting. Create a slack reminder.
- Have at least one hour of time dedicated to sprint planning

8.3 Sprint 3 - Edit functionality on properties

Sprint Goal: Create simple editing functionality on properties via NGIS-OpenAPI

Duration: 2. October - 16. October

8.3.1 Sprint 3 Planning

According to the action point created in the sprint 2 retrospective, the group dedicated some extra time for the sprint planning. This was because the team desired to created more detailed tasks than what had been done in sprint 1 and 2, to make it easier to understand and get started on a task. The sprint 3 backlog is presented in Table 8.

Sprint 3 Backlog:

Table 8: Sprint 3 Backlog

ID	Issue	Estimate	Related FR/BG	Completed?
#28	Edit attributes	L	FR4	Yes
#33	Create loading indicator	S	BG1	Yes
#34	Set up WMS	M	FR7	Yes
#37	Edit geometries	L	FR5	Yes
#39	Set max zoom of leaflet map	XS	BG1	Yes
#51	Use official symbols on objects	M	FR6	Yes

8.3.2 Sprint 3 Implementation

The interviews with Kristiansand port and Arendal port were conducted during the first week of the sprint. These are described in more detail in Section 3.7.

The backlog items for this sprint required further research of NGIS-OpenAPI and Leaflet. In order to implement issue #37, we went for a simplified solution where *latlng* coordinates would have to be manually entered to change the position of objects, instead of being able to edit positions on the map in an interactive way. This was mainly to avoid the issue becoming too large, but would definitely have to be changed in a later sprint.

Some features for usability were also implemented during this sprint. Previously all map points were denoted using the same standard Leaflet symbol. To make a distinction between the different types of map objects, the Kartverket's official map symbols where implemented to denote the different object types. Furthermore a loading indicator was implemented, as well as a restriction on how far the user is able to zoom out on the map. Both of these features improved the usability of the system, as they make it easier for the user to navigate and use the application.

8.3.3 Sprint 3 Review

Due to the customer's participation in both of the conducted usability tests during the sprint, there was no need for a designated customer meeting after this sprint. However, we got plenty of inputs from the usability tests and interviews that we were able to discuss in detail with the customer. This helped the team identify the key takeaways from the tests. The team was also able to perform a status check-in with the customer during these sessions. The general feedback was that the customer was pleased with the team's progress. However they did also point out that the current solution for editing geometries was too inconvenient, and suggested using the Leaflet.draw plugin to implement interactive geometry editing.

During the sprint, the team really experienced the significance of consistent progress updates through daily scrums and the collaborative approach in development work. A lack of cooperation

in the technical sphere led to a few merge conflicts in the code, which could have been mitigated with improved communication and teamwork in coding and reporting.

It should also be noted that the team was invited by the customer to participate in the yearly geomatics conference FOSS4G (`https://foss4g.no/`) in Trondheim on the 25th of October. The conference aims to create a meeting place for discussing open-source software in the geomatics industry. The team was asked to provide a 20 minute presentation about the project, which we were happy to do.

Action Points:

The action points created during the retrospective are presented below:

- Schedule a sustainability meeting for next week.
- Assign tasks and promote pair programming during sprint planning while monitoring status during daily stand-ups.
- Draft content about AI, diversity, and sustainability in the report during Sprint 4.
- Document insights from usability testing
- Plan for FOSS4G.

8.4 Sprint 4 - Edit functionality on geometries

Sprint Goal: Create editing functionality on point and line geometries.

Duration: 16. October - 30. October

8.4.1 Sprint 4 Planning

After Sprint 3 we had completed the three main sub goals of creating a map client that could display and edit attributes of objects with NGIS-OpenAPI. The team was free to choose which remaining sub goals to focus on going forward. During the planning of Sprint 4, we decided to mainly focus on the fourth sub goal of implementing geometry editing, which was partially completed in Sprint 3 already. The backlog for this sprint, presented in Table 9, also included a feature for creating new objects, and a custom layer control for the map as we were not satisfied with the user experience of the default Leaflet layer control.

We agreed with the customer in a later meeting to formally drop the 3D map client from the project scope as it would be difficult to learn something completely new this late in the development process. Instead we decided to potentially do a security analysis of NGIS-OpenAPI if time would allow it. Furthermore, a sustainability focused discussion with the team's supervisor was scheduled, and the team started preparations for the FOSS4G event.

Sprint 4 Backlog:

8.4.2 Sprint 4 Implementation

During this sprint, a lot of functionality was added to the map client. The user can now choose which dataset, or port, they are intrested in, and the map will only display this dataset. Other

Table 9: Sprint 4 Backlog

ID	Issue	Estimate	Related FR/BG	Completed?
#38	Create custom layer control	L	BG1	Yes
#49	Set active dataset	S	BG1	Yes
#52	Interactive geometry editing	L	FR5	Yes
#53	Create new features	L	FR8	Yes
#57	Change map that is displayed based on zoom	M	BG1	Yes
#58	Add satellite and sea map layers	XS	FR7, FR10	Yes
#59	Display "description" property from JSON schema	M	FR3	Yes

custom map controls, like the layer control, were also implemented. While Leaflet already have a lot of default map controls, creating custom controls made it easier for the team to create an UI that aligned with the requirements for the project.

For implementing interactive geometry editing, we ended up not using Leaflet.draw like we originally planned, since we had some problems getting it to work properly. However, later on when implementing creation of new objects, we managed to make Leaflet.draw work for placing new points and lines on the map. This is not a major issue, but we only managed to make editing points possible without Leaflet.draw, so migrating everything over to Leaflet.draw should definitely be done in the future.

8.4.3 Sprint 4 Review

When planning for this sprint, the team did not consider the extra workload the FOSS4G event would impose on the team. The led to some issues originally planned for this sprint to be moved to sprint 5 in order to ensure a balanced workload. FOSS4G also doubled as the sprint review meeting with the customer, as a demo of the project were presented at the event. During this meeting the customer also asked if the team could present the project at a pivotal meeting regarding the continuation and funding of the project. This meeting will take place near the end of the project.

At this this point in the process, the project was becoming of considerable size, and problems concerning the choice of – or rather the lack of – JavaScript frameworks and UI libraries became more apparent. Due to time constraints, the team did not think that the value of incorporating such libraries and frameworks would outweigh the perceived costs of integrating these technologies into the project.

Action Points:

The action points derived from the retrospective meeting is presented below:

- Enhance the granularity of task status reporting during the daily scrum, covering completed tasks and pending work.
- Prioritize pair programming for resolving minor issues.
- Introduce deadlines for pending tasks (report and coding) within the final sprint.
- Provide commentary on UI libraries and frameworks within the project report.

8.5 Sprint 5 - Finalize the project

Sprint Goal: Finalize the product

Duration: 30. October - 13. November (extended to 20. November)

8.5.1 Sprint 5 Planning

During the sprint planning meeting, the team prioritized tasks that would enhance existing functionality rather than implementing completely new features. This was done to ensure that the product would be finished on time, and to ensure the quality of the end product.

Sprint 5 Backlog:

Table 10: Sprint 5 Backlog

ID	Issue	Estimate	Related FR/BG	Completed?
#35	Implement error handling and notifications	M	FR9	Yes
#40	Create search for locations	M	BG1	Yes
#45	Request data based on visible area	L	FR12	Yes
#60	Handle property ”array” types	M	BG1	Yes
#65	Display ”ikke i bruk” symbol	S	FR11	Yes
#69	Confirm deletion	S	FR4	Yes
#71	Create a logo	S	BG1	Yes
#75	Test security	XL	BG9	No

8.5.2 Sprint 5 Implementation

As this was the last planned sprint for the project, more of the team's focus was directed towards the writing of the report, but there was still technical work to be done in order to complete the sprint goal and conclude the technical aspects of the project. As previously mentioned, the focus of this sprint was to enhance existing functionality. All issues except #71 and #75 in Table 10, are such enhancement issues. At this point, the system was not providing much feedback to the user concerning the state of the system. To improve this, proper error handling was implemented. Error handling does not only prevent the application from crashing, it also helps provide feedback to the user about what has gone wrong.

Furthermore, the application was also quite slow because of the high number of data points being fetched from NGIS-OpenAPI. To reduce load times as well as lag in the application, the application was changed to only request data from the area that is currently visible on the map. This issue (#45) was not completed within the initial time frame of the sprint, but we decided to extend the sprint one week to finish this issue, and a few other minor issues like deployment and adding extra map tiles which Norkart suggested towards the end of the sprint. For issue #75, the customer requested that the team would do a security analysis of NGIS-OpenAPI, but due to slow communication with our contact person and other priorities, this was not achieved in time.

8.5.3 Sprint 5 Review

The team had a meeting with the customer to demonstrate the last features that had been implemented. During this meeting the customer requested some additional work. This was mostly adding more WMS map layers and map tiles, but also finding somewhere to temporarily deploy the application. The customer was happy with the implemented functionality, and even expressed that we probably implemented more than what was needed for a prototype. There have also been several meetings with the customer afterwards to discuss the presentation and demo the team is having in the pivotal meeting.

Since this was the last sprint of the project, its duration should have been longer from the start than previous sprints. During the spring planning, the group should have taken unforeseen work into consideration. When reflecting about this during the sprint retrospective, the group agreed to extend the sprint.

Regarding the report, the team lacked a good system for ensuring that parts of the report that were already written were thoroughly reviewed by multiple team members. Additionally, the team should have utilized our supervisor and customer earlier in the report process to ensure that the entire document is reviewed by both parties. To improve this, the team expanded our worksheet for the report to include a column where team members can write their names when they have reviewed a section. By doing this, the team gains an overview over what have been reviewed, and which parts need more attention.

Action Points:

The action points created for the conclusion of the project during the retrospective meeting are listed below:

- Review parts of the report internally as they are finished and then send to the supervisor
- Produce all report text Friday 17.11 (preferably before Friday)
- Go through the whole report and finalize it on Monday 20.11

9 Security

Software systems are critical to modern business operations, making their security and reliability a top priority, particularly for those exposed to external risks via internet connectivity. Our internet-connected software as a result, necessitates a thorough security assessment to identify, understand, and mitigate potential threats. This process, known as threat modeling, is crucial for safeguarding valuable assets against security vulnerabilities (OWASP, 2023).

Threat modeling is an extensive, continuous endeavor that is never truly complete. However, due to the limitations in the project's scope and time, we have opted for a more focused security evaluation. This evaluation follows the principles of the Risk Assessment Framework by McGraw (McGraw, 2022), a framework established by the American Cybersecurity & Infrastructure Security Agency. It outlines five essential steps for conducting security assessments.

1. Understand the business context
2. Identify and link the business risks and technical risks

3. Synthesise and rank the risks
4. Define a risk mitigation strategy
5. Carry out fixes and validate

To enhance the robustness of our security assessment, we integrated key practices from "The Seven Touch Points of Secure Software" (McGraw, 2005), which are highlighted as important in our course compendium (Jaccheri, 2023). This approach involved the development of abuse cases to pinpoint risks (Table 12), the establishment of explicit security requirements, and the execution of risk analysis at the design and architectural stages. Additionally, we leveraged recognized security frameworks such as the STRIDE Threat Model (Microsoft, 2009) to identify potential threats and the OWASP Cheat Sheet (Microsoft, 2009) to inform our risk mitigation strategies.

The following subsections provide an in-depth look at the outcomes of our risk assessment. These findings were instrumental in developing a comprehensive test plan, which is outlined in Section 10 Testing. It's important to mention that our security assessment was somewhat limited in scope, as E.g. managing access control was not part of the project requirements.

9.1 Understanding the Business Context

In the document, the business context is established by examining the organization's objectives and valuable assets. This context, as outlined by McGraw (2022), is pivotal for understanding the impact of various risks. An initial exploration of this context is presented in Section 1.1, titled 'Overall Context.' This section sets the stage for a more detailed discussion found in Section 3, which delves into the 'Problem Space and Solution Space.'

Further, an overview of the business goals is systematically laid out in Table 4. This table provides a clear and concise reference for understanding the organization's objectives. Building upon this foundation, we have addressed the business assets, which are shown in Table 11.

Table 11: Business Assets

Business Assets	
ID	**Description**
BA1	Port and location data
BA2	Source code maintainer rights
BA3	Servers and databases
BA4	Brand reputation

9.2 Risk identification, evaluation and mitigation strategy

This section addresses step 2, 3 and 4 of the Risk Assessment Framework, focusing on the identification, evaluation and mitigation of risks.

Risks are categorized into two main types: business risks and technical risks. Business risks are those that have a direct effect on the achievement of business objectives. Technical risks, on the other hand, concern the practical aspects of how the system might be compromised by potential attacks (McGraw, 2022).

To effectively identify and prioritize these risks, it is essential to possess a comprehensive understanding of the application's design as well as insights into the profiles and capabilities of potential attackers.

9.2.1 Application design

Following the guidelines on threat modeling from the OWASP, 2021 Cheat Sheet, understanding the application's design is essential to identify potential risks effectively. A crucial aspect is knowing how data circulates within the system and identifying trust boundaries, as detailed in Section 7 Architecture.

In the back end's data layer, transactions are confined to a physical server, making unauthorized access to this server or the client's machine highly challenging. As such, internal interactions within each layer are considered secure. The primary trust boundaries, therefore, are where data transfers over the public Internet.

The back end is designed not to store user data persistently or modify data through endpoints, indicating that the likeliest attack vector is the data flow via internet communications. Accordingly, our threat modeling focuses on safeguarding data transmitted over the Internet.

9.2.2 Potential attackers

By evaluating potential attackers' motivations, skills, and resources, we can assess the likelihood of threats to our system. Given that this project focuses on developing a prototype with a very limited user base and doesn't involve personal or sensitive data like user credentials or credit card details, the chance of sophisticated, organized cyber attacks is low. The likely attackers might be driven by the intent to disrupt the application's operations, get access to geological data or to spread miss information about port data.

However, the risks of legal issues and reputation damage from such attacks act as deterrents, making us think potential attackers are only moderately motivated. In terms of skills and resources, these attackers are also considered to have a moderate level. They are likely to have a basic knowledge of internet protocols and programming, along with access to standard hacking tools. Advanced, technical attacks like buffer overflows, requiring more sophisticated expertise and resources, are considered unlikely from such attackers.

9.2.3 Abuse cases

Beginning the process of risk identification, a strategic approach is to construct abuse cases, which effectively channel the mindset of a potential attacker. Abuse cases, akin to use cases, articulate how the system might react when compromised. The development of abuse cases necessitates a clear outline of the assets requiring explicit coverage of what should be protected, from whom, and for how long (McGraw, 2005). Given that this outline is given in preceding sections we are well-positioned to apply the OWASP Abuse Case Cheat Sheet to generate plausible abuse scenarios. By following the guidelines provided in the cheat sheet, our team conducted an abuse case development workshop. The artifacts of this workshop are the abuse cases enumerated in Table 12.

Table 12: Abuse cases

Abuse Case	Description
Sensitive Data Exposure	As an attacker, I steal keys that were exposed in the application to get unauthorized access to the application or system.
Using Components with Known Vulnerabilities	As an attacker, I find common open source or closed source packages with weaknesses and perform attacks against vulnerabilities and exploits which are disclosed.
Security Misconfiguration	As an attacker, I find unnecessary features which are enabled or installed (e.g. unnecessary ports, services, pages, accounts, or privileges) and attack or exploit the weakness.
Denial-of-service attack with anonymous accounts	Attackers can take advantage of the anonymity of the application to attack the system by repeatedly opening a browser, or overwhelm the web page.
Spread miss information	An attacker can simply edit port data deliberately to make information wrong. This can be done unintentionally by a user as well.
Unauthorized Access and Data Tampering	As an attacker, I gain unauthorized access to manipulate geographical and port data, causing disruptions or safety risks.
SQL Injection	As an attacker, I exploit input fields to inject SQL queries, manipulating the database and leading to data theft or corruption.
Cross-Site Scripting (XSS)	As an attacker, I exploit XSS vulnerabilities to inject scripts, stealing cookies or redirecting users to malicious sites.
Cross-Site Request Forgery (CSRF)	As an attacker, I trick users into making requests that alter port data, exploiting lack of CSRF protection.
Man-in-the-Middle (MitM) Attack	As an attacker, I intercept unencrypted communications to access or alter sensitive information.
API Security Flaw Exploitation	As an attacker, I exploit insecure APIs to access or corrupt data.

9.2.4 Identification and ranking of risks

Each identified business and technical risk is evaluated and assigned a likelihood, impact, and overall risk rating. These ratings are categorized as low (L), medium (M), or high (H). Taking into account the business context, the application's design, and the profile of the probable attacker, we have pinpointed the primary business risks (BR), which are cataloged in Table 13.

Table 13: Business Risks

Business Risks				
ID	**Description**	**Likelihood**	**Impact**	**Risk**
BR1	Application data accessed by unauthorized users	M	M	M
BR2	Application is not user friendly	L	H	M
BR3	Possibility to edit without authorization	L	H	M
BR4	High cost of maintaining the system	L	M	M
BR5	Low availability of application	L	M	M
BR6	Application is not useful	L	M	L

The technical risks have been identified, prioritized, and correlated with their corresponding business risks in Table 14. While this list is not comprehensive, it highlights the most significant risks pertinent to the application's architecture and the characteristics of a potential attacker. Specifically, the emphasis is on high-level network threats such as request tampering, packet sniffing, and injection attacks. Additionally, we propose a mitigation strategy for each identified risk.

Table 14: Technical Risks

Technical Risks						
ID	**Description**	**Likelihood**	**Impact**	**Risk**	**Mitigation Strategy**	**Related Business Risk**
TR1	Vulnerability to SQL injection or XSS through user inputs	M	H	H	Implement rigorous input validation protocols and regular security patch updates	BR1, BR3
TR2	Potential exposure of sensitive data in source code	L	H	M	Enforce a strict policy against hardcoding passwords/API keys	BR1
TR3	Risk of unencrypted traffic leading to data exposure	L	H	M	Implement mandatory TLS/SSL encryption for all data transfers	BR1
TR4	Exceeding API rate limits due to CSRF attacks	L	H	M	Establish a robust CORS policy limited to authorized frontends	BR5
TR5	Issues due to complex or suboptimal code	M	M	M	Adhere to industry-standard coding practices and regular code reviews	BR2, BR3, BR4, BR5
TR6	User interface design not meeting user expectations	M	M	M	Commit to standard UI design guidelines and usability testing	BR2, BR6

9.3 Fixes and validation

The final phase of the Risk Assessment framework involves implementing solutions to address the identified technical risks and establishing tests to confirm the effectiveness of these fixes. The details of this process, including the outcomes and validation tests, are documented in Table 15.

Table 15: Implemented Solutions and Their Validation Methods and Outcomes

ID	Implemented Solution	Validation Method and Outcome
TR1	Validated all input fields	Monitored logs for SQL injection attempts; no such attacks detected. Also ensured to never set innerHTML directly with user input to prevent XSS attacks.
TR2	Conducted thorough code review	Verified absence of exposed tokens in the codebase
TR3	Ensured all endpoints use HTTPS	Performed requests and inspected messages for secure transmission
TR4	Set CORS policy to allow only frontend requests	Executed requests from different origins to test CORS policy enforcement
TR5	Established continuous integration with GitHub Actions for build and unit testing	Confirmed smooth build process and passed all unit tests without issues
TR6	Interactive feedback gathering from customers and stakeholders (Acceptance Testing)	Received positive feedback, particularly on UI design aspects

10 Testing

Testing has many purposes in software development. The most obvious benefit is to detect and fix bugs, but it can also serve a crucial role in requirement analysis and validation. Going into the project the customer emphasized that the product should mainly be a prototype, meaning that it did not need to work perfectly. Regardless we still wanted the product to be free of any major bugs or vulnerabilities. Therefore we employed four main testing strategies during the project, which we will describe in the following sections. These strategies can be linked to quadrants 1 and 3 of the *Agile Testing Quadrants* by Crispin & Gregory (2008).

10.1 Exploratory testing

Exploratory testing is a manual testing strategy where developers try to use the system in the same way as end users, guided mainly by creativity, intuition and critical thinking (Crispin & Gregory, 2008, 102). The team used exploratory testing while developing new features to go beyond the obvious variations that are already captured by unit tests. In our case we also needed to use a lot of exploratory testing to figure out how to use NGIS-OPENAPI. This involved reading the documentation thoroughly, sending example requests to the API to see what data we received with a lot of trial and error.

Another major benefit we noticed is that exploratory testing highlighted quite a few new ideas and potential features that we ended up including in the backlog and developing in future sprints. An example of this is how we noticed that it would be very useful to have tooltips that explain

different properties of objects being created, since they would otherwise require some technical knowledge to understand.

10.2 Unit testing

Unit testing is an automated way of testing small, isolated components or "units" to make sure they work as intended. They belong in quadrant 1 of the *Agile Testing Quadrants*, meaning they are tests that support the team. The reason for this is that they help avoid unintended changes to the rest of the system when implementing new features, as this would hopefully break existing unit tests (Crispin & Gregory, 2008, p. 99). This creates a safety net for developers to confidently make changes without worrying about breaking the application. We did not have a specific code coverage goal in mind when writing unit tests. Instead we mainly focused on utility functions that were used multiple places.

Vitest was chosen as our unit testing framework as it was easy to configure with our existing build tool (Vite), and unit tests ran automatically on each commit with Husky precommits. We also set up a continuous integration pipeline with Github actions to build the application and run unit tests. This was automatically run on pull requests to the develop branch to ensure that it would always be in a stable state. Overall we did not have a huge focus on unit testing and automated tests during the project, as we found manual testing strategies such as exploratory testing sufficient.

10.3 Acceptance testing

During some customer meetings, usually after a sprint, the team had an informal showcase of the current state of the product. This allowed the customer to give feedback and further clarify their intentions for how the application should work. The team experienced that some existing tasks were modified, and a few new issues were created as a result of this feedback.

10.4 Usability testing

Usability testing is an important procedure to discover requirements and demands from the users of the system. It helps the designers measure the usability of a system or product in cooperation with the actual users (Niranjanamurthy et al., 2014). The team was interested in the systems efficiency, which can be described as how well the system supports the user in performing specific tasks.

In advance of the usability tests, a script with tasks was prepared for the person conducting the test, which described specific scenarios a typical end user might encounter. These scripts can be viewed in Appendix 20.

To prepare for the tests, the team held a meeting and discussed which key features in the application would be valuable to get feedback on. After discovering these features the team made a Figma prototype. This prototype resembled the actual map client, but had further functional and design solutions that were not fully implemented. This allowed the team to test the user experience of certain features before spending more time implementing them in the actual product. Additionally, the Figma prototype later served as a wireframe, or blueprint for how the GUI should be designed in the end product.

11 Internal and External Documentation

In the following sections we will go over all the different types of internal and external documentation that were used. By internal documentation we mean all the documents that were used with the purpose of assisting the development team and communicating with the supervisor and customer. External documentation on the other hand is targeted at explaining different aspects of the product to external parties such as potential users, the examiner, and anyone else who may be interested in our project.

11.1 Internal documentation

- **Meeting documents:** Meetings were separated into three main categories: development team meetings, supervisor meetings, and customer meetings. A single Google docs document for each category was used to summarize what we discussed during the meetings. An agenda was created before each meeting with a predefined template, which can be found in Appendix E.
- **Sprint retrospectives:** As mentioned in Section 4.2.9, we used RetroTool to conduct and document our all our retrospectives.
- **Github issues** were used as the primary tool for documenting requirements related to product functionality, and also coordinating work within the team. Grouping issues by different statuses as described in Section 4.3 gave a further overview of the progress made in a sprint. We also grouped issues in specific Github milestones that contain small summaries for the goal of their corresponding sprint.
- **Code comments:** We mostly followed the principle that "good code should be self-documenting." Adding line comments simply describing what the code is doing is completely useless in most cases (Sourour, 2017). Instead we focused on documenting functions in a standard format with JSDoc when we felt like code documentation was necessary.
- **CONTRIBUTING.md:** Contributing guidelines were created on Github early on in the project to serve as a reference for everything we agreed on related to code style, commits, and workflow. This document can also be used by future developers on the project if they want to adapt the same conventions that we did.
- **Decision log:** To keep track of important decisions related to functionality, technology, and teamwork, we documented them in a shared table along with additional background information for the decision, the decision taker(s), and date. The idea was also that the customer and future developers have all the relevant context for why things were done the way they were. The full decision log can be viewed in Appendix D.

11.2 External documentation

A temporary deployment of the product can be viewed at `https://folk.ntnu.no/eriksalv/TDT4290-leaflet-client/`. This version captures the state of the product at the end of the student project. If development continues later on, the application will most likely be deployed somewhere else. For those who want to run the project locally instead, all documentation related to this is contained in the README file on the project's Github page. For the sake of convenience we have also included this information in Appendix B. A user manual for all product functionality can be found in Appendix C.

12 Self-Evaluation

We will now go over some reflections about the project as a whole, such as how we worked together, what we learned, what went well, what did not go so well, and our experiences with the customer and supervisor. We will also present some potential future work for the project, and feedback for the course.

12.1 Working Together as a Team

We collaborated and communicated effectively as a team, maintaining a positive dynamic and great team spirit throughout the entire project. Notably, we successfully avoided any personal conflicts. We effectively addressed uncertainties as they arose, engaging in professional and inclusive group discussions. Encouraging everyone to voice their opinions, we navigated through different perspectives, not always reaching an immediate consensus but consistently arriving at solutions that satisfied everyone after thorough discussion.

Although our team collaboration was successful, in hindsight, we recognize that we could have done better in getting to know each other more personally. While some group members were acquainted before the project, none of us knew each other well. Early on, we discussed the idea of fostering stronger connections through social activities outside of school, but unfortunately, this was never prioritized. Our differing academic schedules and social commitments made it challenging to find a suitable time. Looking back, we realize that with a bit more effort, we could have made it happen, something we wish we had attempted. We did manage to have a social lunch at school once, where we intentionally focused on topics unrelated to school. This experience positively impacted our team spirit, providing a valuable opportunity to deepen our understanding of each other, especially considering the significant time we spent together on this project throughout the semester.

12.2 The Work We Are Proud of

One thing we really nailed throughout the project, although we sort of lost our groove towards the end due to illness, was our group work sessions. We firmly believe that these in-person sessions played a crucial role in our success and contributed to maintaining a positive team dynamic.

Additionally, we take pride in our adaptability at planning and distributing tasks within our team. Our commitment to the seriousness of sprint planning sessions laid a robust foundation for achieving optimal success in each sprint. We made a conscious effort to allocate work efficiently, ensuring that team members received tasks aligned with their strengths and preferences. This approach not only promoted a harmonious work environment but also prevented any team member from feeling overburdened.

We were good at conducting sprint retrospectives, believing that, like planning, it would establish a strong foundation for the next sprint. We allocated time and meticulously planned to ensure the participation of everyone in the group. Our retrospectives were carried out in a serious manner, and we felt that we generated valuable action points for the upcoming sprint. However, one challenge we faced was consistently implementing these action points in subsequent sprints, such as reserving technical demos of the product on Fridays after Sprint 1. We recognize that this might be attributed to the lack of repetition and reminders within the group.

12.3 Reflecting on Project Challenges

We acknowledge that our performance in code testing and project security fell short of our expectations. Reflecting on the project, we realize the need for more collective enforcement of the test plan. The initial stages were challenging, with numerous new considerations, leading to oversight on testing and security. Recognizing this, we made concerted efforts to address these aspects in the latter half of the project.

Looking back, we acknowledge that reassessing our team's capacity and distributing the workload more evenly could have been a prudent choice. Nevertheless, the experience taught us valuable lessons in capacity assessment, and how to implement new templates and standards. If afforded more time, we believe we could have demonstrated what we learned more in depth.

12.4 The Customer

Our communication with the customer was satisfactory, despite most of the meetings being remote. Primarily, the team lead took charge of interactions such as scheduling meetings and sending agendas. However, they actively encouraged all group members to pose questions independently, fostering an open line of communication. The customer displayed good responsiveness to emails and Teams messages. Moreover, they were approachable and provided encouraging feedback, making us feel respected as a group. They were also very helpful in pointing us in the right direction regarding the technical solution, including recommended libraries, and online tutorials and documentation.

In hindsight, if given the opportunity to redo the project, we might consider establishing fixed meeting times with the customer every other week. This could have streamlined the planning process and eliminated the need to consistently remember and schedule meetings.

12.5 The Project Assignment

The client's project assignment, displayed in Appendix F, offered valuable and well-detailed information, providing a solid foundation for our task. The project description was beneficial, reflecting the client's thoughtful input. We heavily relied on it at the project's start, appreciating its thorough completion.

We noted that "AI" was selected under "The Technology to be exploited in the project." Initially, we assumed this meant developing a large-scale AI solution for the product. However, there was little mention of the machine learning aspect when presented the project, so we asked for more clarification on the topic during the next customer meeting. After a more detailed conversation with the customer, it became clear this was more of a low priority task to potentially do if we had the time.

12.6 The Supervisor

As detailed in Section 2, we maintained weekly meetings, where her support was invaluable, and her positive attitude greatly contributed to our collaborative environment. Whenever we sought assistance, she consistently provided helpful answers and guidance, offering valuable advice and tips throughout the project. She was for instance happy to give feedback and prepare us for the FOSS4G conference, and gave us concrete tips for the report. As a group, we were satisfied with the level of supervision.

Reflecting on our experiences, one aspect we discussed for potential improvement involves clarifying the purpose of supervisor meetings. Initially, there was some ambiguity surrounding the goals of these sessions, but over time, the purpose became clearer. We believe a more defined understanding of the meeting objectives from the outset could enhance the overall effectiveness of the supervision process.

12.7 Future work

The goal of the project was to produce a prototype for a map application that allowed the user to view and manipulate geographical port data, as detailed by our functional requirements in Table 5. Additionally, the product needed to be user friendly as this is the main issue with existing applications.

The team believes the final product meets all the requirements in a satisfying manner. This view was further reinforced through feedback from the customer as well. However, there is still room for further work on the product. The list below presents what can be further worked and improved upon:

- Implement a 3D map client with CesiumJS to visualize data from NGIS along with depth data. This was one of the optional subgoals presented by Norkart initially.
- Some form of user management and authentication, mainly for security, as well as giving individual users different rights. This would remove the need for hiding API credentials through a proxy.
- A log of user activity, i.e., who has edited what and when, and a comment section to communicate with coworkers, as suggested in the interview with the Port of Arendal (see Section 3.7.3). This would likely require the proxy to be extended with a database.
- Design the map client for smaller screens. We only had large screens like desktops and laptops in mind when designing the GUI, as this was seen as the most prominent use case. However it would still be a good practice to make the design more mobile-friendly.
- Add additional filtering of the data shown. Sometimes the user could be interested in only viewing the objects with a certain field value. Adding customizable filters could make the experience faster and smoother.
- Add more visualization of the data on the map, such as highlighting which objects are "In use" or similar. This should be apparent to the user without the need for clicking the object to manually check.

12.8 Suggestions for Improvement

The training sessions on AI, Diversity, and Sustainability should have been extended to include all students, not just the group leaders. We found it somewhat odd that only one person from each group, the team lead, received information on these crucial aspects during group lead meetings. Considering that these are new additions to the course and account for 15% of the final grade, we believe that all group members should have been informed. It could have been beneficial to organize lectures for all the students, where these topics are thoroughly covered.

Moreover, we feel that insufficient time was allocated to learn about AI, Diversity, and Sustainability during group lead meetings. The extensive discussion of each point in the Word document left

little time for the main agenda. Additionally, the expectations regarding the topics team leaders should cover in these meetings were somewhat unclear.

To enhance the efficiency and focus of our group lead meetings, our team discussed a proposal where each group shares one of their successful practices as a helpful tip for others, in addition to discussing any challenges they are currently encountering. Furthermore, we agreed it would be unnecessary for each group to detail aspects like meeting schedules, since these are already comprehensively documented and available for reference when needed.

References

Agile Alliance. (2023, March 13). *What is collective code ownership?* Retrieved from https://www.agilealliance.org/glossary/collective-ownership/

Altexsoft. (2021, January 18). *Extreme programming: values, principles, and practices.* Retrieved from https://www.altexsoft.com/blog/extreme-programming-values-principles-and-practices/

Anthony Jnr, B., Che Pa, N., Khalefa, M., Alasad, H., & Zmezm, H. (2016, 07). A proposed risk assessment model for decision making in software management. *Journal of Soft Computing and Decision Support Systems*, *3*, 31-43.

Arend, R. J., Zhao, Y. L., Song, M., & Im, S. (2017). Strategic planning as a complex and enabling managerial tool. *Strategic Management Journal*, *38*(8), 1741-1752.

Atlassian. (n.d.-a). *Gitflow workflow — atlassian git tutorial.* Retrieved from https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow (n.d.)

Atlassian. (n.d.-b). *Trunk-based development.* Retrieved from https://www.atlassian.com/continuous-delivery/continuous-integration/trunk-based-development (Atlassian. (n.d.-b))

Bass, L., Clements, P., & Kazman, R. (2021). *Software architecture in practice* (4th ed.). Pearson Addison-Wesly.

Brundtland, G. H. (1987). *Report of the world commission on environment and development: Our common future.*

Burnett, M., Stumpf, S., Makri, S., Macbeth, J., Beckwith, L., Kwan, I., ... Jernigan, W. (2016). Gendermag: A method for evaluating software's gender inclusiveness. *Interacting with Computers*. doi: 10.1093/iwc/iwv046

Capiluppi, A., & Jaccheri, L. (2023, 9 25). Booting and Rebooting Academia-Industry Collaborations within Software Engineering Courses. *IEEE Software*.

Condori-Fernandez, N., & Lago, P. (2018). Characterizing the contribution of quality requirements to software sustainability. *Journal of Systems and Software*, *137*, 289–305.

Crispin, L., & Gregory, J. (2008). *Agile testing: A practical guide for testers and agile teams* (1st ed.). Addison-Wesley Professional.

Dönmez, D., Grote, G., & Brusoni, S. (2016). Routine interdependencies as a source of stability and flexibility: A study of agile software development teams. *Information and Organization*, *26*(3), 63-83.

GeeksforGeeks. (2023, November 2). *Waterfall model software engineering.* Retrieved from https://www.geeksforgeeks.org/waterfall-model/

Google. (2022). *Diversity annual report - google diversity equity & inclusion.* Retrieved from https://about.google/belonging/diversity-annual-report/2022/

Han, W.-M., & Huang, S.-J. (2007). An empirical analysis of risk components and performance on software projects. *Journal of Systems and Software*, *80*(1), 42-50. Retrieved from https://www.sciencedirect.com/science/article/pii/S0164121206001440 doi: https://doi.org/10.1016/j.jss.2006.04.030

Heijstek, W., & Chaudron, M. (2008, September). Evaluating rup software development processes through visualization of effort distribution. In *2008 34th euromicro conference software engineering and advanced applications* (p. 266-273).

Hirshfield, L., & Koretsky, M. D. (2017). Gender and participation in an engineering problem-based learning environment. *IJPBL*.

Iso 9001:2015. (2015, September 1). Retrieved from https://www.iso.org/iso-9001-quality-management.html

Jaccheri, L. (2023, August 23). *Compendium.* Retrieved from https://miro.com/blog/scrum-kanban-boards-differences/ (Course compendium found on BlackBoard,)

Kalla, D., & Smith, N. (2023). Study and Analysis of Chat GPT and its Impact on Different Fields of Study. *International Journal of Innovative Science and Research Technology*, *8*(3).

Kniberg, H. (2007). *Scrum and xp from the trenches: Enterprise software development.* Lulu.com eBooks. Retrieved from http://dl.acm.org/citation.cfm?id=1554790

Kruchten, P. (1995, November). Architectural blueprints - the 4+1 view model of software architecture. *IEEE Software*(12(6)), 42-50.

McDonald, K. (2023, October 19). *What is extreme programming (xp)?* Retrieved from https://www.agilealliance.org/glossary/xp/

McGraw, G. (2005). *The 7 touchpoints of secure software.* Retrieved 2023-11-07, from https://www.drdobbs.com/the-7-touchpoints-of-secure-software/184415391

McGraw, G. (2022). *Risk management framework (rmf).* Retrieved 2023-11-07, from https://www.cisa.gov/uscert/%20bsi/articles/best-practices/risk-management/risk-management-%20framework-%5C%28rmf%5C%29

Microsoft. (2009). *The stride threat model.* Retrieved 2023-11-07, from https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)

Microsoft. (2023). *Diversity & inclusion report.* Retrieved from https://www.microsoft.com/en-us/diversity/inside-microsoft/annual-report?activetab=innovation-spotlights:primaryr4

MiroBlog. (2023, September 17). *Kanban vs. scrum boards: 11 major differences.* Retrieved from https://miro.com/blog/scrum-kanban-boards-differences/

Nguyen-Duc, A., Jaccheri, L., & Abrahamsson, P. (2019). An empirical study on female participation in software project courses. *IEEE/ACM 41st International Conference on Software Engineering*.

Niranjanamurthy, M., Nagaraj, A., Gattu, H., & Shetty, P. K. (2014). Research study on importance of usability testing/user experience (ux) testing. *International Journal of Computer Science and Mobile Computing*(3(10)), 78-85.

Norkart. (2023). *Norkart.no.* Retrieved 2023-11-17, from https://www.norkart.no/

Oslo Havn. (2023). *10,5 millioner i støtte til digitalisering av norske havner.* Retrieved 2023-11-20, from https://www.oslohavn.no/no/aktuelt/105-millioner-i-stotte-til-digitalisering-av-norske-havner/

OWASP. (2021). *Owasp top 10.* Retrieved 2023-11-07, from https://owasp.org/Top10/

OWASP. (2023). *The owasp risk assessment framework.* Retrieved 2023-11-07, from https://owasp.org/www-project-risk-assessment-framework/

Pham, Y. D., Bouraffa, A., & Maalej, W. (2020, August). Shapere: Towards a multi-dimensional representation for requirements of sustainable software. In *2020 ieee 28th international requirements engineering conference (re)* (p. 358-363). IEEE.

Rehkopf, B. M. (n.d.). *What is a kanban board?* Retrieved from https://www.atlassian.com/agile/kanban/boards

Satell, G. (2019, May 19). *How to make an ai project more likely to succeed.* Retrieved from https://hbr.org/2018/07/how-to-make-an-ai-project-more-likely-to-succeed

ScrumAlliance. (n.d.). *What is scrum: a guide to the most popular agile framework.* Retrieved from https://www.scrumalliance.org/about-scrum (n.d.)

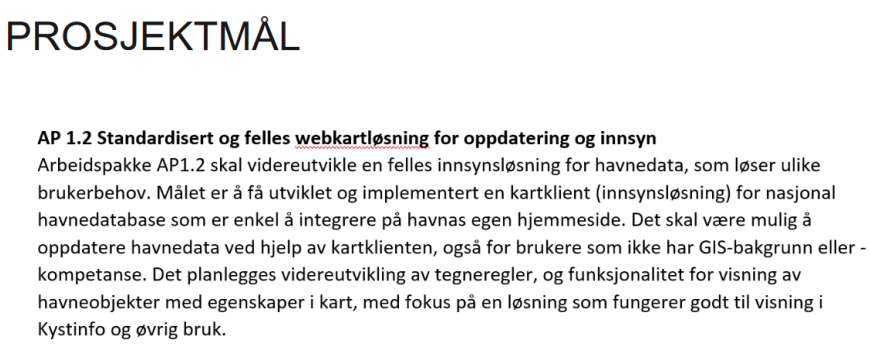
Sommerville, I. (2016). *Software engineering.* Pearson Education Limited.

Sourour, B. (2017, April 20). *Putting comments in code: the good, the bad, and the ugly.* Retrieved from https://www.freecodecamp.org/news/code-comments-the-good-the-bad-and-the-ugly-be9cc65fbf83/

Vaughan-Nichols, S. J. (2003). Building better software with better tools. *Computer*, *36*(9), 12-14.

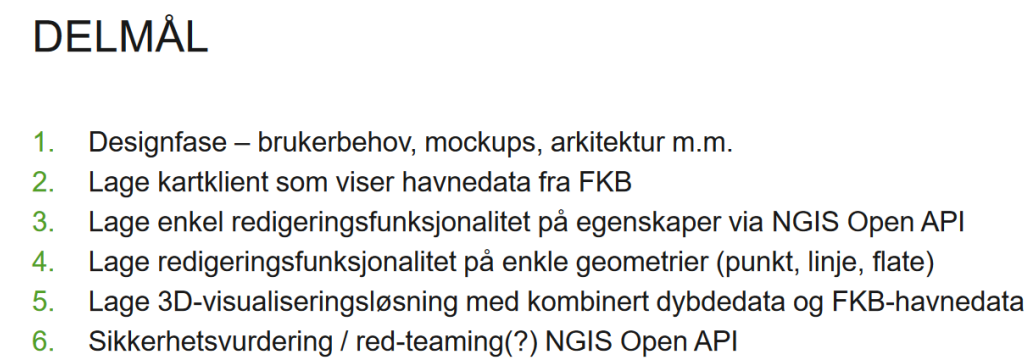
Westland, J. (2023, October 30). *The quality assurance process: Roles, methods & tools.* Retrieved from https://www.projectmanager.com/blog/quality-assurance-and-testing

A Project goals as presented by the customer


PROSJEKTMÅL

AP 1.2 Standardisert og felles webkartløsning for oppdatering og innsyn
Arbeidspakke AP1.2 skal videreutvikle en felles innsynsløsning for havnedata, som løser ulike brukerbehov. Målet er å få utviklet og implementert en kartklient (innsynsløsning) for nasjonal havnedatabase som er enkel å integrere på havnas egen hjemmeside. Det skal være mulig å oppdatere havnedata ved hjelp av kartklienten, også for brukere som ikke har GIS-bakgrunn eller -kompetanse. Det planlegges videreutvikling av tegneregler, og funksjonalitet for visning av havneobjekter med egenskaper i kart, med fokus på en løsning som fungerer godt til visning i Kystinfo og øvrig bruk.

Figure 10: The overall goal of the bigger project this project is a part of


DELMÅL

1. Designfase – brukerbehov, mockups, arkitektur m.m.
2. Lage kartklient som viser havnedata fra FKB
3. Lage enkel redigeringsfunksjonalitet på egenskaper via NGIS Open API
4. Lage redigeringsfunksjonalitet på enkle geometrier (punkt, linje, flate)
5. Lage 3D-visualiseringsløsning med kombinert dybdedata og FKB-havnedata
6. Sikkerhetsvurdering / red-teaming(?) NGIS Open API

Figure 11: The project plan as proposed by the customer

B Installation guide

The following guide describes how to run the application locally. If you just want to view the application, we recommend visiting `https://folk.ntnu.no/eriksalv/TDT4290-leaflet-client/` instead. Also note that credentials are required to access data from NGIS-OpenAPI. This can only be provided by a data owner. However if you don't have credentials, and are not interested in editing data you can still follow the guide below to view the map with WMS data (images).

Prerequisites

- Use Windows, macOS, or Linux
- Have git installed
- Have Node.js installed

Clone remote git repository

1. `git clone https://github.com/digitalhavn/prototype-redigeringsklient-ngis.git`
2. Navigate into the root project folder: `cd prototype-redigeringsklient-ngis`

Install dependencies

From the root folder run: `npm install` to install dependencies for both leaflet-client and proxy at the same time.

Optional: run proxy

1. Create a file called `.env` inside the `proxy` folder with the following variables:
   - ⋆ NGIS_URL: URL for the server hosting NGIS-OpenAPI
   - ⋆ NGIS_USERNAME: Username to access NGIS-OpenAPI
   - ⋆ NGIS_PASSWORD: Password to access NGIS-OpenAPI
2. Run `npm run dev -w proxy` from the root folder

Run frontend leaflet client

From the root folder run: `npm run dev -w leaflet-client`

Alternatively, both leaflet-client and proxy can be run at the same time from the root folder with `npm run dev`

Run leaflet client locally without proxy

If you don't have credentials for a server running NGIS-OpenAPI, you can still run leaflet-client locally by pointing it to the proxy deployed to Microsoft Azure. This can be done automatically by building leaflet-client for production:

1. From the root folder run: `npm run build -w leaflet-client`
2. `npm run preview -w leaflet-client`

C User manual

C.1 Start screen

- Reset the location and zoom of the map back to the initial position by clicking the button marked "Reset map" in Figure 12
- Change the dataset (port) to fetch data from by changing the input marked "Change dataset" in Figure 12. Note that attempting to fetch data outside of the selected dataset boundaries will result in an error.

- Search for addresses and move the maps location to a specific address by using the search input marked "Search address" in Figure 12.
- Switch the tile layer of the map, or toggle symbol and depth WMS layers in element marked "Tile/WMS layer control" in Figure 12

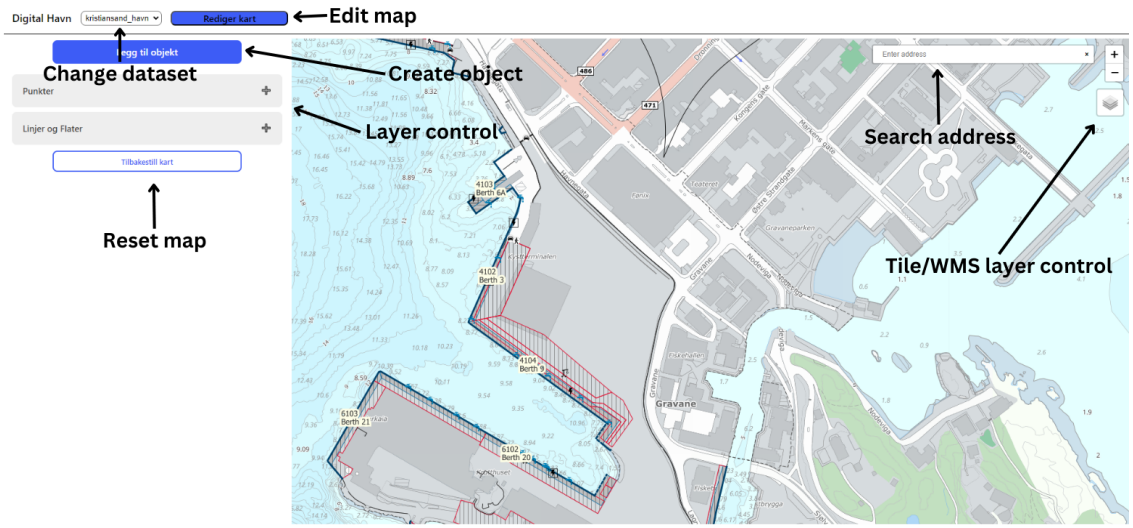


Figure 12: Start screen

C.2 Display data layers

The application will initially load data from NGIS-OpenAPI based on the area that is visible on the map. After moving the map to a new location, it will load in new data, as indicated by a loading spinner. To actually display this data, you have to enable specific features in the "Layer control" marked in Figure 12. This layer control is split into two main sections: points (markers), and lines + polygons. Open up one of these sections by clicking on it. This will create a dropdown of all feature types that are displayable within that section. Click on the checkbox for a feature type to display all the objects of that type of the map. Figure 13 shows a snippet of the map after enabling the "Fortøyningsinnretning" type.

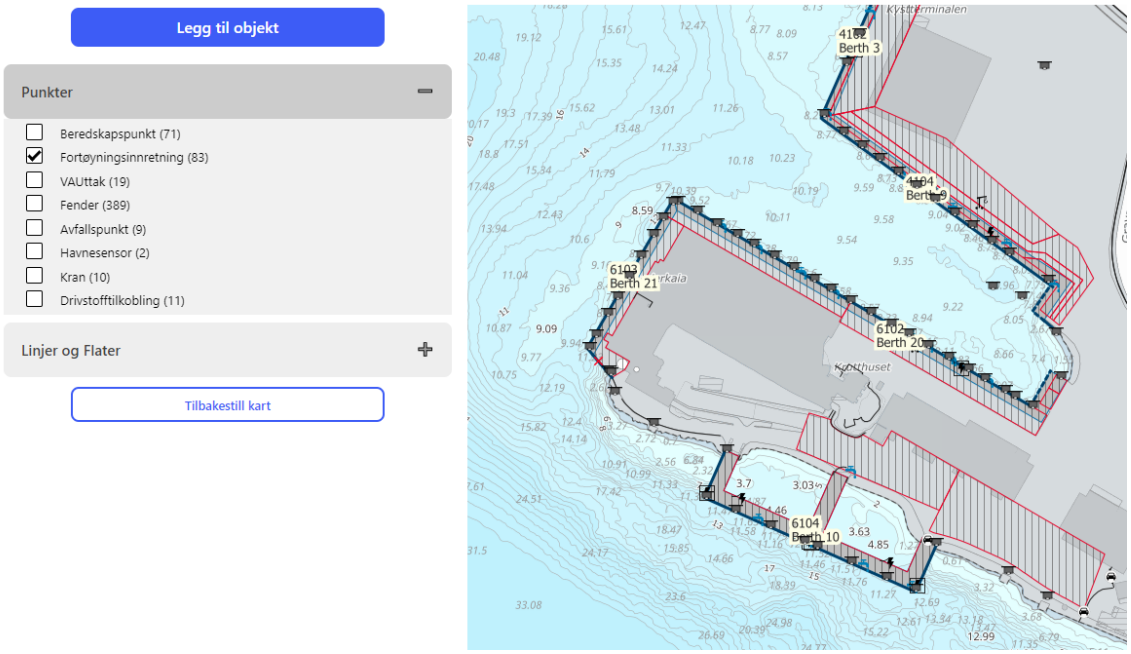


Figure 13: Display data

C.3 Display and edit feature details

After displaying features on the map, they can be clicked to display detailed information about that specific feature. Figure 14 shows the details of a "Fortøyningsinnretning" and its corresponding location on the map. 14.

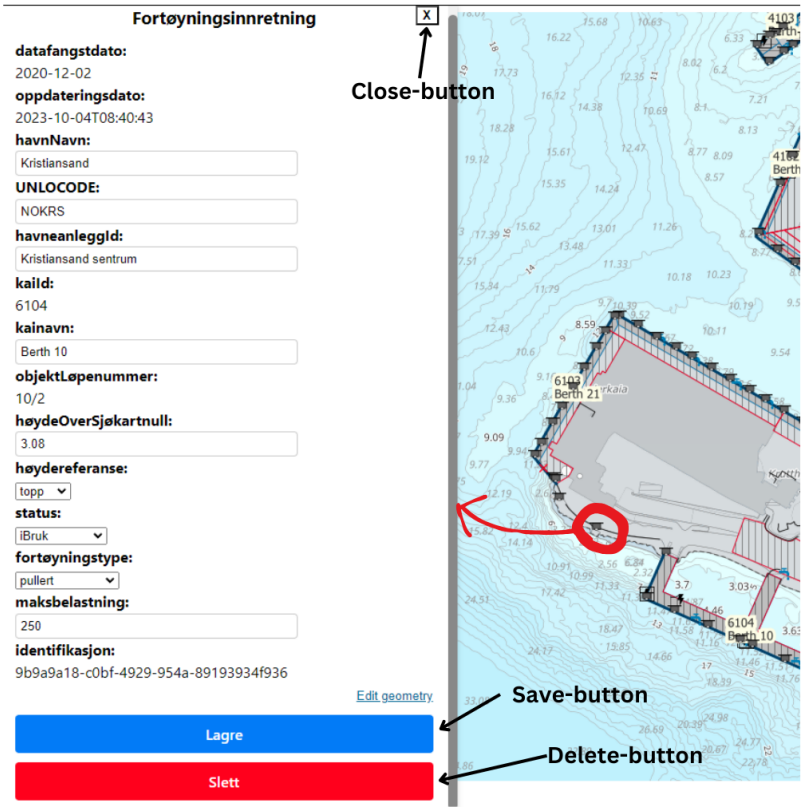


Figure 14: Feature details

To edit the details, simply change one or more of the input elements that are editable, and click the Save-button in Figure

C.4 Delete a feature

After displaying the details of a feature, you can click the Delete-button in 14 to delete the entire feature. This will first open up a modal to confirm that you want to delete it or not.

C.5 Create a new feature

Click the button marked with "Create object" in Figure 12 to open up a modal, which initially contains a single input to choose what type of feature you want to create. Choose a specific feature type to generate additional inputs where you can enter the details you want to, as shown in Figure 15 with the "Beredskapspunkt" type. You can hover over any of the input labels to display a tooltip containing additional information about the meaning and purpose of specific inputs. Inputs that are required are marked with a red star (*).

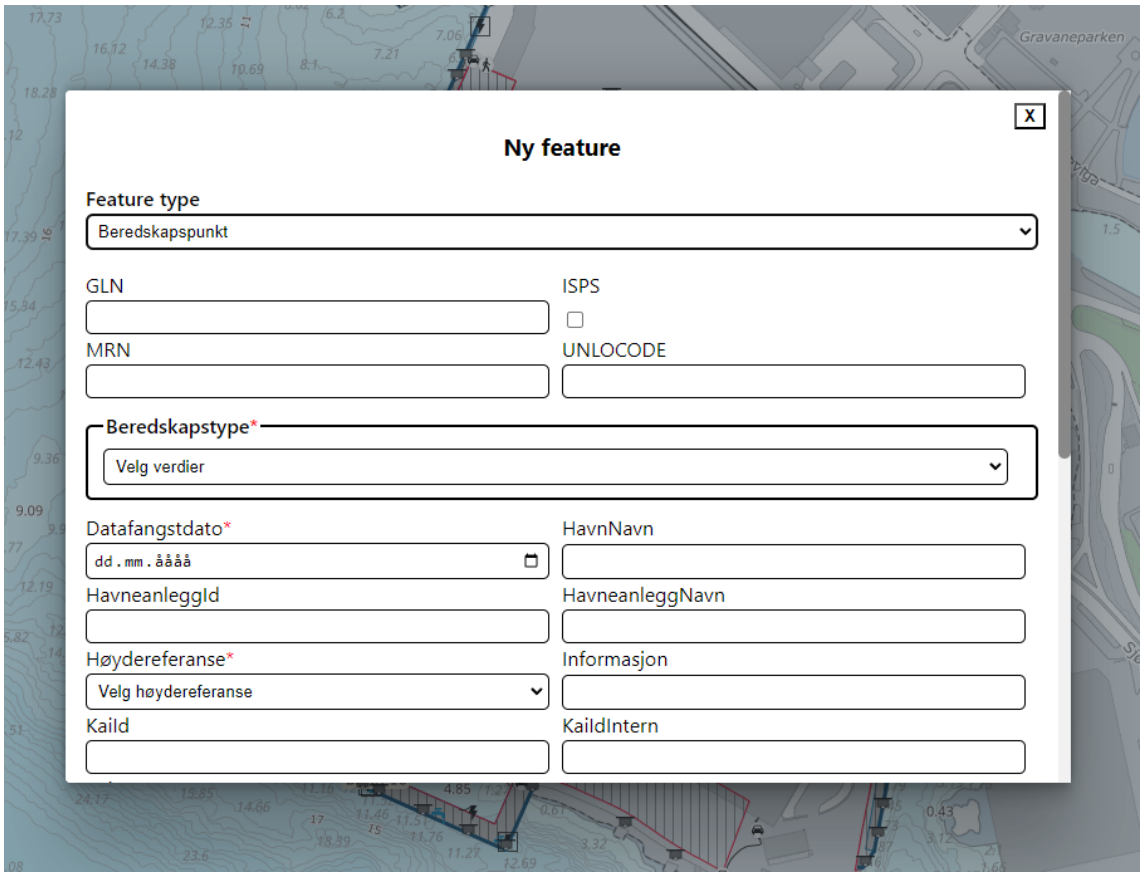


Figure 15: Modal for creating a new feature

After adding information, create the feature clicking the button at the bottom of the modal (you may need to scroll down). This will close the modal and let you click on a point on the map to place the feature.

C.6 Move features around

NOTE: Only editing the geometry of points/markers is currently supported.

Start by clicking the button marked "Edit map" in Figure 12 to enter the map into edit-mode. You will then be able to drag displayed features around on the map. When you are done moving features around, click the Save-changes button marked in Figure 16 to confirm the changes with NGIS-OpenAPI, or click the Undo-changes button to reset the changed features back to their original locations.

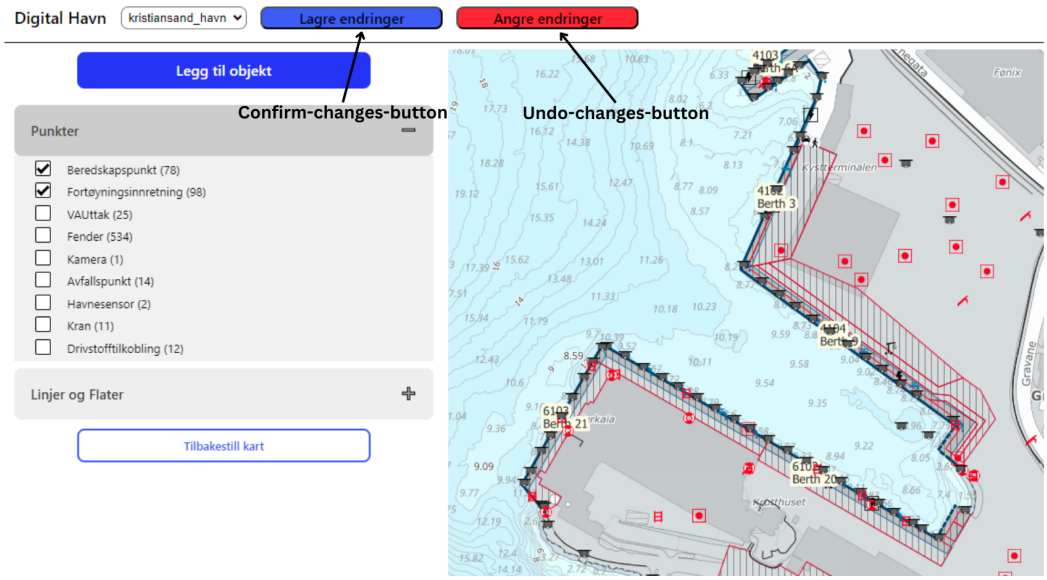


Figure 16: When inside edit mode, features become draggable, and two new buttons are added to the header to confirm or undo changes to the map

C.7 Configuration

The application exposes a few configuration options through environment variables to tailor its experience to whoever is hosting it. Table 16 shows all the possible configuration options, and what their purpose is. Read the projects README for more details.

Table 16: Configuration options

Name	Purpose
VITE_MAPTILES_API_KEY	API key to access Norkart's webatlas maptiles
VITE_START_LOCATION_LAT	Start location latitude for leaflet map
VITE_START_LOCATION_LNG	Start location longitude for leaflet map
VITE_START_ZOOM	Start zoom for leaflet map
VITE_NGIS_DEFAULT_DATASET	Default NGIS-OpenAPI dataset to fetch data from initially
NGIS_URL	URL pointing to a server hosting NGIS-OpenAPI
NGIS_USERNAME	Username for accessing NGIS-OpenAPI
NGIS_PASSWORD	Password for accessing NGIS-OpenAPI

D Decision log

Table 17 contains all important choices made in the course of the project.

Table 17: Decision log

Decision	Background	Decision taker	Date
Mandatory meetings mondays 09-14, and wednesdays with supervisor	Vital to have a timeframe where everyone can participate in meetings	All	04.09
Optional meetings on fridays 08-16	To work on issues together. Come and go as you want.	All	04.09

15 minutes daily standup on mondays and fridays	Short status check to plan and adjust what to continue working on	All	04.09
Github projects for project management	Preferred by the customer.	All	04.09
Use Figma for mockups	Great tool, and we have experience using it before.	All	04.09
Hold a presentation about project 25. october at FOSS4G	Suggested by the customer, and we thought it was a great opportunity	All	07.09
The start location for the map client should be configurable	Makes it more convenient if the web page is embedded on specific websites for different ports	Norkart	07.09
Use test server from kartverket	Norkart set up credentials for us to read and write data on a non production test server to use during development.	All	11.09
Use Typescript on both frontend and backend	Javascript is required on the frontend for leaflet, and Typescript has additional benefits with static typing. Using the same language on both front and back makes it easier for all team members to quickly learn the prerequisites to contribute since we have limited time.	All	11.09
Don't use any framework on the frontend	Leaflet already handles the map GUI, and the application will only contain one page (the map), which makes frameworks like react or vue less beneficial. There will probably be a little more work without a framework, however the application will probably also be more performant, and we have greater control.	All	11.09
Use express js framework on proxy	Minimal, easy to work with. It is probably overkill to use a framework just for the proxy, but considering the proxy is probably going to be removed in the final product when user authentication is implemented, we want to spend minimal time on this component.	Erik	11.09
Change daily standup from fridays to wednesdays	It was a bad idea to put daily standups in an optional meeting.	All	17.09
Use Vitest for unit testing, and husky precommits	Vitest modern and easy to integrate with Vite. Husky precommits runs unit tests and linting before a commit is accepted to ensure code quality and that nothing breaks.	All	25.09
Structure the project as a monorepo with Lerna	Makes it possible to share commands for building, testing, etc. between frontend and backend since both are written in Typescript	Erik	25.09

Portion the data from NGIS-OpenAPI	Retrieving all available data at the same time takes a long time, and slows down the client. Firstly, we can limit data to one dataset at a time, and then further limit data to only retrieve what is visible on the map.	All	09.10
Use leaflet draw for interactive geometry editing	Recommended by customer	All	09.10
Use AJV for JSON schema validation	Popular and easy to use library.	Jesper	13.10
Limit the scope of the student project to exclude editing geometry of polygons	Customer mentioned that this is much more complicated to implement and not worth it for the prototype.	All	20.10
Exclude 3D visualization from the student project and instead focus on security if we have time	We would rather focus on implementing more functionality for the 2D map client instead of starting something completely new that we may not have time to get working. Customer also said that testing security was more important	All	20.10
Use native leaflet event handlers for interactive geometry editing instead of leaflet draw	Couldn't get leaflet draw to work correctly, but managed to make editing markers work just using leaflet (but not linestrings).	Jesper, Erik	27.10
Use leaflet draw for placing new objects on the map	Managed to get leaflet draw to work for creating new markers and linestrings. Planned to implement this for editing existing features as well.	Erik	30.10
Use a multiselect to handle "array" types in JSON schema	Some properties like "beredskapstype" can have multiple values at the same time	Eli	10.11
Created temporary deployment of project	Nice to have for stakeholders and examiner to try out	All	14.11

E Meeting templates

E.1 Student meetings

Student meeting template

Time: -

Place: -

Facilitator: -

Transcriber: -

Preparations:

- -

Agenda:

- Attendance (5 min):
  - ☐ Erik
  - ☐ Eli
  - ☐ Jesper
  - ☐ Johan
  - ☐ Sara
  - ☐ Gard
- Stand up - Scrum (15 min)
  - What have you done since last time?
  - What are you going to work on?
  - Do you have any obstacles preventing you?
- Walk the board (15 min)
- Discuss questions (20 min)
- If end of sprint: do sprint retrospective, else work on issues (40 min)
- Break (15 min)
- If end of sprint: do sprint planning, else continue work (60 min)
- Lunch (30 min)
- Status check, assign new tasks (10 min)
- Keep working (60 min)
- End of meeting check out

Other notes:

- -

TODOs for next meeting:

- -

Figure 17: Template for meetings held between students on mondays and fridays

E.2 Supervisor meetings

Supervisor meeting template:

Time: -

Place: -

Facilitator: -

Transcriber: -

Attendance:

- ☐ Erik
- ☐ Eli
- ☐ Jesper
- ☐ Johan
- ☐ Sara
- ☐ Gard
- ☐ Letizia

Minutes from the last meeting

Stand up - Scrum (15 min)

- What have you done since last time?
- What are you going to work on?
- Do you have any obstacles preventing you?

Work done since the last meeting

- List below:

Problems encountered

- List below:

Planning of work for the next period:

- List below:

Other issues (if there are any):

Figure 18: Template for meetings held between students and supervisor on wednesdays

E.3 Customer meetings

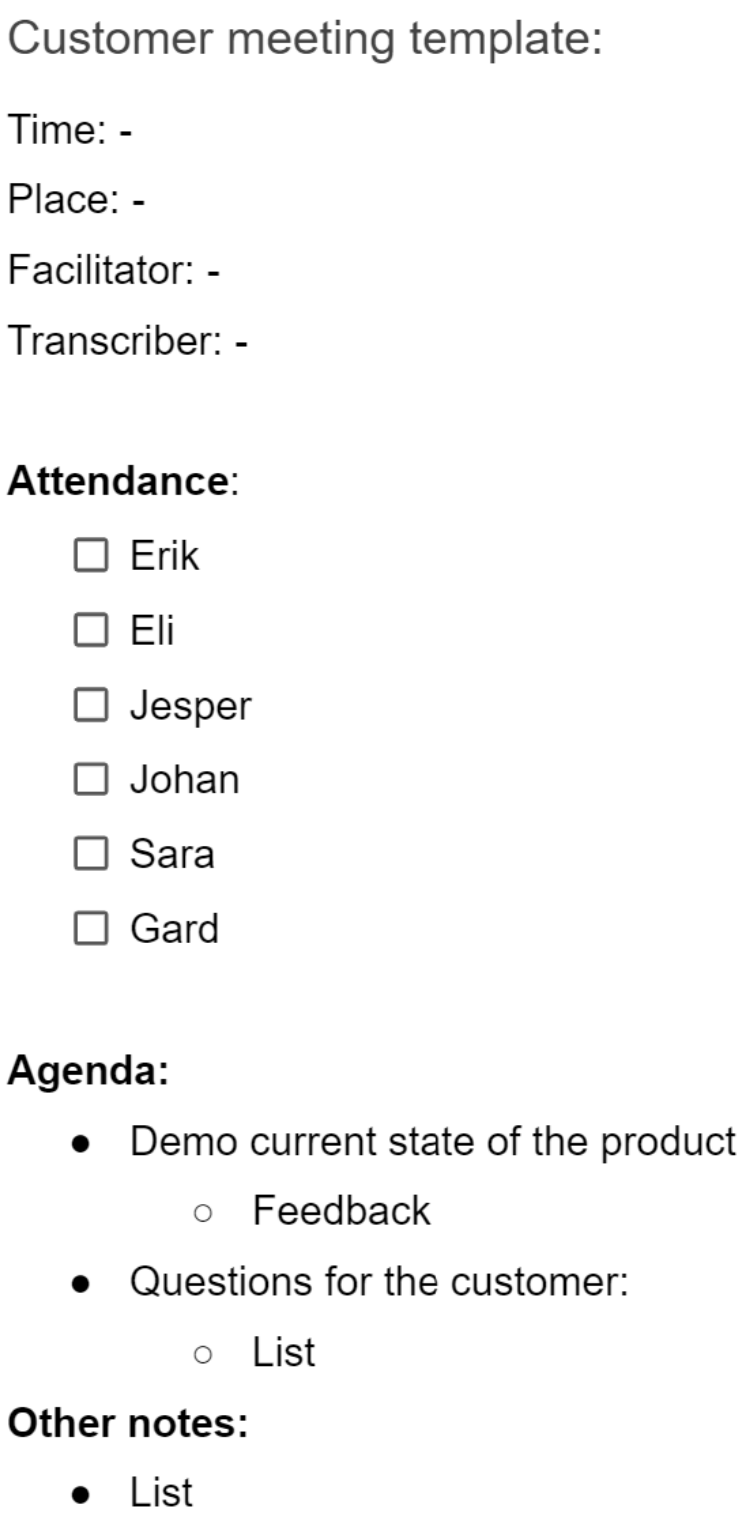


Figure 19: Template for meetings held between students and Norkart

E.4 Usability test meetings

Vi introduserer oss selv, og målet med testen (Johan leder intervjuet)

- Målet med testen er å avdekke om vi har skjønt oppgaven og hva som er bra, hva som kan gjøres bedre.

Innledende spørsmål:

- Vi ønsker å ta opptak av møtet, er det greit for deg?
- Hvem er du?
- Hvilke rolle har du i Kartverket?
- Erfaringer?
- Hvor godt kjenner du til Havnedata prosjektet?
- Kjenner du til oppgaven vi har fått i forbindelse med Havnedataprosjektet?

Før brukertest

- Forklar at vi ikke kan svare på tekniske spm, men at generelle spørsmål kan besvares
- Snakk høyt og reflekter over valg i prototypen
- bruk fit to screen

Oppgaver til brukerintervju i Figma:

1. Finn en livbøye på kartet, trykk på denne for å få opp informasjon
2. Rediger informasjonen til livbøyen.
3. Vis kaiområdet på kartet.
4. Finn informasjon om søndre kai.
5. Kaien er litt for lang, gjør den kortere.
6. Skjul kaiområdene.
7. Legg til drivstoff tilkobling som et punkt på kartet og fyll inn egenskaper.
8. Du fant ut at du la til drivstofftilkoblingen på feil side av kaien, flytt den til andre siden.

Spørsmål:

Samsvarer det du har blitt presentert med de delmålene over?
Hva var bra?
Hva kan gjøres annerledes?
Mangler det noe?
Samsvarer din forståelse av det som skal gjøres med vår forståelse?
Hvilke konkrete problemer skal løsningen adressere?

- gjerne eksempler

Hvordan skal denne løsningen skille seg ut fra allerede eksisterende løsninger?

- Finnes allerede gode kartløsninger

Figure 20: Template for usability test meetings held between students and different ports

F Project assignment

View results

Respondent

74 Anonymous

36:59
Time to complete

1. First Name *

Alexander

2. Last Name *

Nossum

3. Organization *

Norkart

4. Organization Type *

- [ ] Startup Company
- [ ] NGO
- [x] Private Sector
- [ ] Public Sector
- [ ] Other

5. Number of Employees in your organization (Approx.) *

250

6. Email address of the contact person *

alexander.nossum@norkart.no

7. Mobile Number

41293632

8. Role of the contact person (CEO, CTO, Developer, Tester, Other) *

Innovation manager

9. Title of the project *

Leveraging spatial technologies in port management

10. The Technology to be exploited in the project *

- [x] AI
- [x] Web Technology
- [ ] Augmented Reality
- [ ] Virtual Reality
- [ ] IoT
- [ ] Other

11. The process to be exploited in the project *

- (•) Scrum
- ( ) Waterfall
- ( ) Extreme Programming
- ( ) Lean programming
- ( ) Other

12. Abstract (Max. 200 words) *

Port/shipping management is highly concentrated around the geographical aspects of a port. Efficient communication of information from the port to the ships are essential. This can be security aspects, capacity, depth/keel clearance. Several Norwegian ports have the last years been laser scanned and made detailed digital twin models of - in addition to highly detailed map objects of the port and the bathymetry. This enables digitalization of the communcation, self-service solutions and also autonomous robots to handle containers and similiar objects.

However, the port management systems do not integrate the digital twin models efficiently. The task at hand is to develop smarter systems for port management that incorporate a high degree of user friendlieness in addition to integrate the spatial datasets and API's available to share, manage and update the digital twin model. If the group have sufficient interest it is possible to also include machine learning aspects on the actual datasets in order to better extract data from the digital twin and/or make smart suggestions in the actual software system.

The project will require developing a system in cooperation with users and stakeholders from both the port, Norkart, Kartverket and ship captains and is expected to be of real value for the end users.

13. Project description (Max. 1 page of 3000 characters) *

Port/shipping management is highly concentrated around the geographical aspects of a port. Efficient communication of information from the port to the ships are essential. This can be security aspects, capacity, depth/keel clearance. Several Norwegian ports have the last years been laser scanned and made detailed digital twin models of - in addition to highly detailed map objects of the port and the bathymetry. This enables digitalization of the communcation, self-service solutions and also autonomous robots to handle containers and similiar objects.

However, the port management systems do not integrate the digital twin models efficiently. The task at hand is to develop smarter systems for port management that incorporate a high degree of user friendlieness in addition to integrate the spatial datasets and API's available to share, manage and update the digital twin model. If the group have sufficient interest it is possible to also include machine learning aspects on the actual datasets in order to better extract data from the digital twin and/or make smart suggestions in the actual software system.

The project will require developing a system in cooperation with users and stakeholders from both the port, Norkart, Kartverket and ship captains and is expected to be of real value for the end users.

The project goal is to create new web based, multi-tenant systems which integrates vast amount of spatial data sets from the digital twin. Both geographic maps, in coordination with tabular data and data sources from other related API's. The user interface will need to be adaptable to different sets of users and include both a dashboard overview and an advanced data management interface for updating different aspects of the port data set.

An important feature of the software system will be to share and collaborate between the port and the ship captains/agents. This includes sharing views of the data - but likely also to include annotations and sharing text/images in a dialog. Researching the user requirements and what encompasses a minimum viable product is part of the task.

The project/group will be integrated in product development teams at Norkart and expert stakeholders from Kartverket and Ports in Norway. The group is expected to be part of scoping the project to fit the capacity of the group's capabilites.

14. Technology Constraints (Max. 1 page of 3000 characters) *

The architecture will be required to be developed using Open Source technologies. The data sets and management will largely be based on a modern national API-standard called NGIS Open API which Norkart develops for Kartverket. There are a vast amount of geospatial technology/libraries that can be used for developing the system. Likely there will be a combination of Docker, PostgreSQL, Javascript libraries (Leaflet) and map based platforms such as GeoNode, MapStore, GeoServer. The group will leverage several open source components which will do the heavy lifting of the geospatial part which will enable the group to focus and deliver rapidly on the innovative features of the system.

The group will in collaboration with Norkart agree on the technology architecture upon start of the project.

15. How does the project targets sustainability issues?

Ports are essential parts of trading between sea and land. Even small adjustments to the efficiency of handling container ships and the required land based car operations have major environmental impacts. One example from earlier projects is the effect of a new sea depth measurements which enabled the ship to make one less round-trip back and forth from Norway to Canada - every year.

There are a lot of other similar related impacts on sustainability in port management that the project will touch upon.

16. How does the project targets diversity issues?

Better digital communication will enable people from more backgrounds to be better included in the dialog between ship and port.

17. Other Constraints (Max. 1 page of 3000 characters) *

-

18. I confirm that I have understood the NTNU rules according to which, the student has copyright to the assignment he/she writes. Having copyright means deciding whether the work should be made available to the public, e.g. by publishing through NTNU Open. It also means that it is the student who decides whether the thesis can be copied, but NTNU can take the necessary copies for carrying out censorship and archiving. *

Yes - we expect the outcome to be published open source.

G Group contract

Group Contract: Customer Driven Project

Motivation and work effort

- What level of ambition do you have as a team in the project work?
  - Everyone does their best
  - Try to at least get a B or better
- How do you define that work is "completed"?
  - when the work is tested and approved by the group
- How much do members expect to work with the subject?
  - work within reasonable hours and work more when needed
  - no more than 24 hours per week
  - minimum 10 hour per week
  - we have to track time spent and what we spend the time on

Routines and communication

- What meetings and routines for work should the group have?
  - mondays 9-14 mandatory from 9,
  - wednesday 8:15-9 with Letizia, mandatory
  - biweekly - meeting with customers, mandatory
  - fridays - 8-16, elective
- How many daily scrums: mondays, wednesdays
- How will you communicate in the group when you are not together?
  - Slack, messenger for more urgent matters, teams for customer
- How will you give each other feedback?
  - Comments on GitHub, also oral feedback, comments in report

Roles

- What expectations do you have for the roles in a development team?
  - **Team leader/ project manager**
    - Gather all data from their respective groups and enter it in the group leaders' meeting agenda one day before.
    - Negotiate with the customer to solve problems and challenges.

Figure 21: Group contract written and signed at the start of the project. Part 1

        - Inform his/her group about the findings and decisions in the group leaders' meeting.
        - Follow up with the supervisor about the impediments.
        - coordinating meetings within the group and with the customer
        - sending invite to the customer
    - **Developer member** - Everyone
        - be up to date with course information
        - keeping track for time spent on various tasks
        - motivating team members
    - **Scrum master** - Erik starts
        - make sure the scrum is upheld.
    - **Report manager -** Johan
        - ensure that current version of the report is up to date,
        - ensure the team is on track according to the timeline of report deliverable
        - assign tasks to other team members
    - **Tech lead** - Jesper
    - **Architecture lead** - Eli
    - **Front End design** - Eli
    - **Documentation manager** - Erik
        - organizing writing notes and reflections
    - **Quality Manager** - Gard
        - ensure quality of end product and report
- How do you plan to distribute and/or roll the roles?
    - Scrum master rotates on request

Handling of conflicts

- How do you handle disagreements?
    - everyone gets a say and the final decision is in the majority of votes
    - keep it professional
- How is breach of contract handled?
    - warning at first, then if its a big problem for the group and the group agrees it can be taken to supervisor.

Figure 22: Group contract. Part 2

- breach of meetings result in bringing something positive to the next meeting

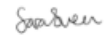
Deliverables

- demo
  - make presentation after each sprint
- final report - 23. november
  - plan to be done at least 2 days before
  - make draft to presentation
- presentation - 28. november

Signatures:

Date: 04.09.2023
Location: Trondheim

Sara Sveen, sarasvee@stud.ntnu.no



Jesper Samuelsen Elverum, jespee@stud.ntnu.no

Eli Fjellbirkeland Johannesen, elifjoh@stud.ntnu.no

Johan Otto Munkeby

Erik Salvesen, eriksalv@stud.ntnu.no

Gard Drag-Erlandsen, gardd@stud.ntnu.no

Figure 23: Group contract. Part 3

H Results From Sprint Retrospectives

H.1 Sprint 1

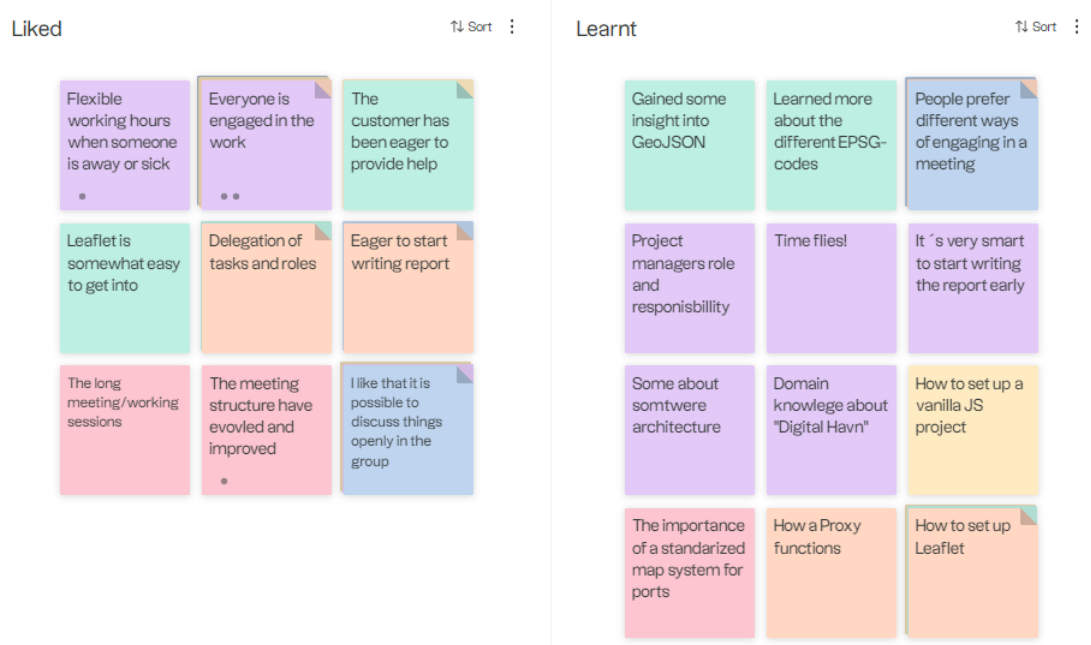


Figure 24: Results from Retrospective Sprint 1: Liked and Learnt

Figure 25: Results from Retrospective Sprint 1: Lacked and Longed for

H.2 Sprint 2

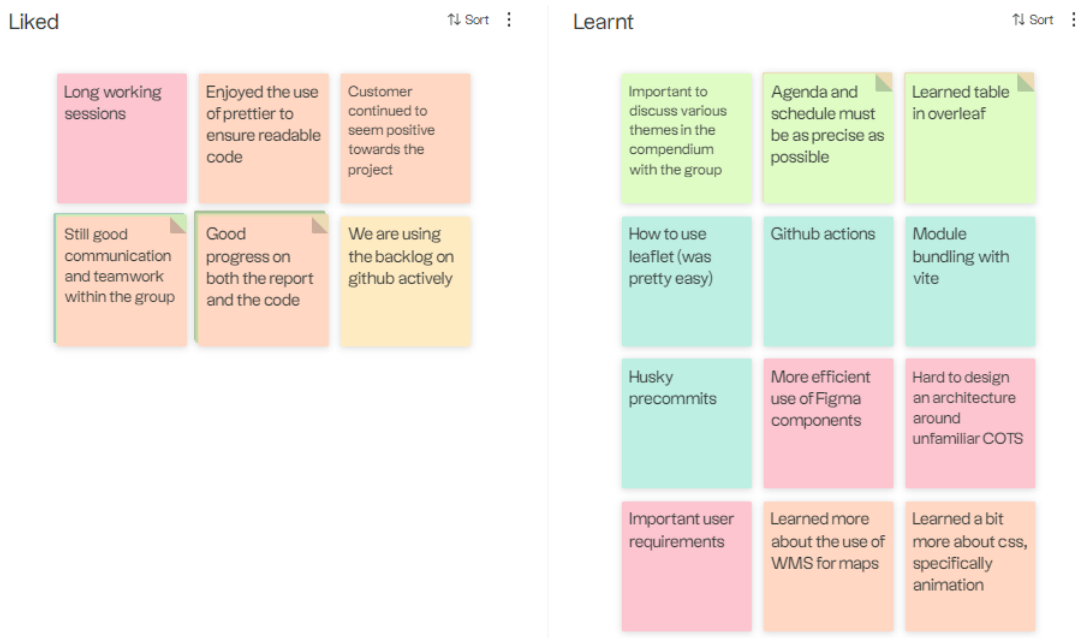


Figure 26: Results from Retrospective Sprint 2: Liked and Learnt

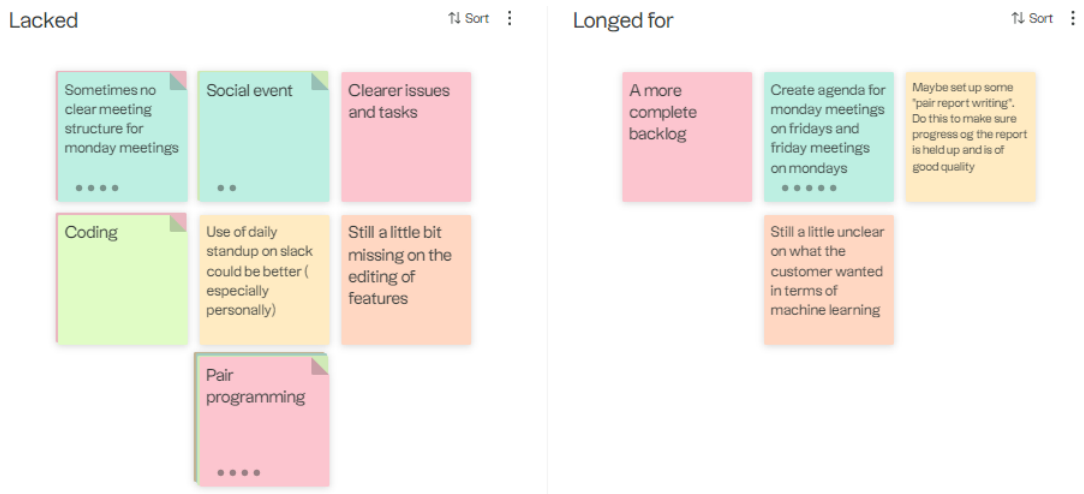


Figure 27: Results from Retrospective Sprint 2: Lacked and Longed for

H.3 Sprint 3

Figure 28: Results from Retrospective Sprint 3: Liked and Learnt

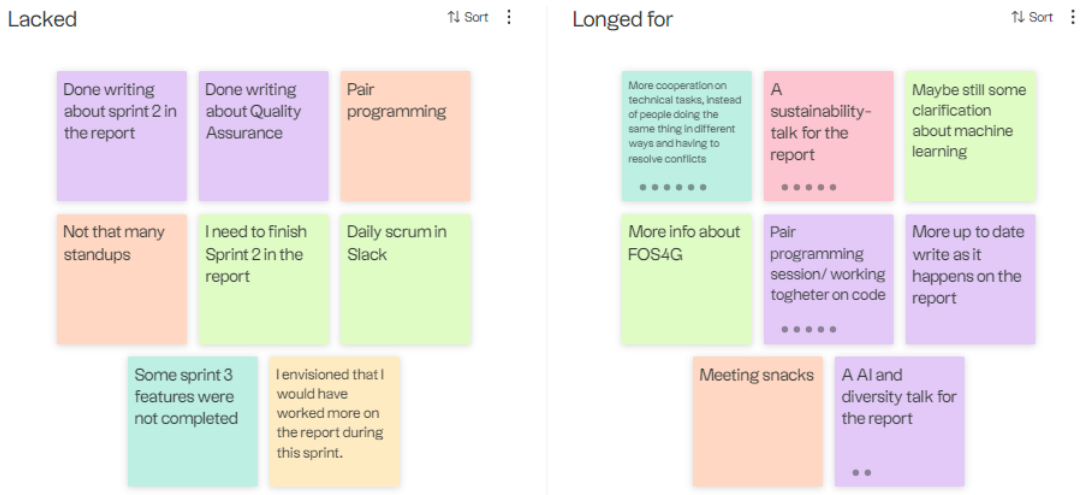


Figure 29: Results from Retrospective Sprint 3: Lacked and Longed for

H.4 Sprint 4

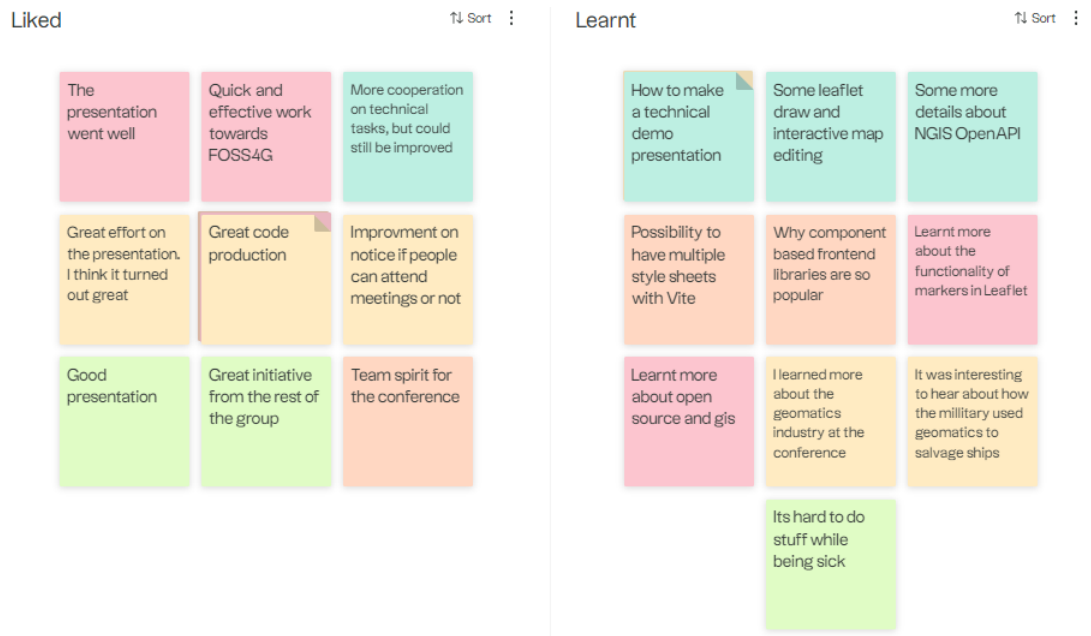


Figure 30: Results from Retrospective Sprint 4: Liked and Learnt

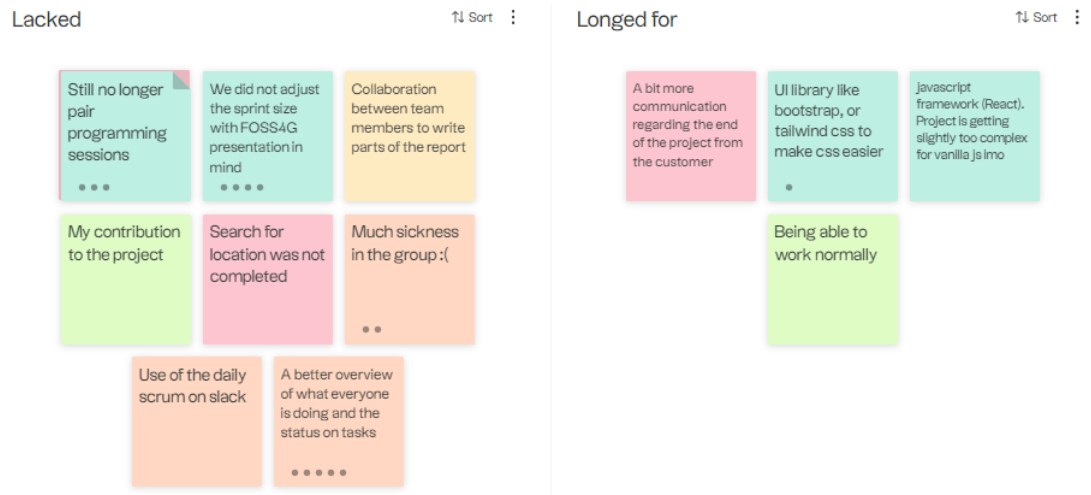


Figure 31: Results from Retrospective Sprint 4: Lacked and Longed for

H.5 Sprint 5

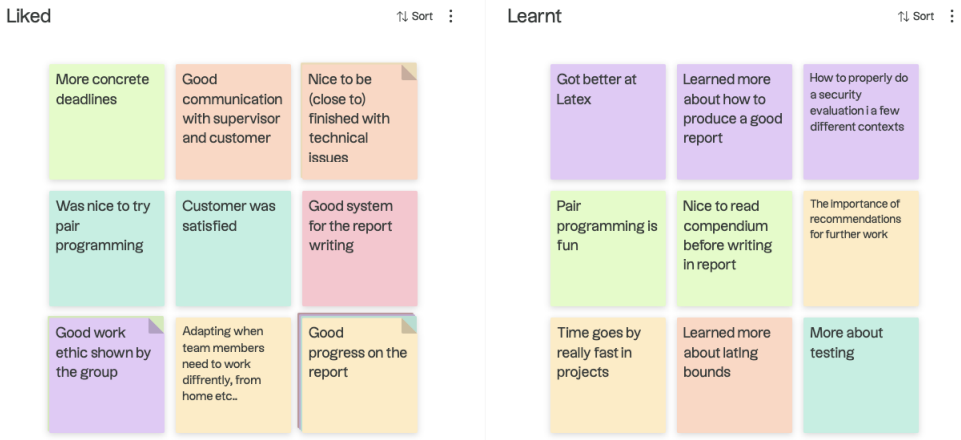


Figure 32: Results from Retrospective Sprint 5: Liked and Learnt

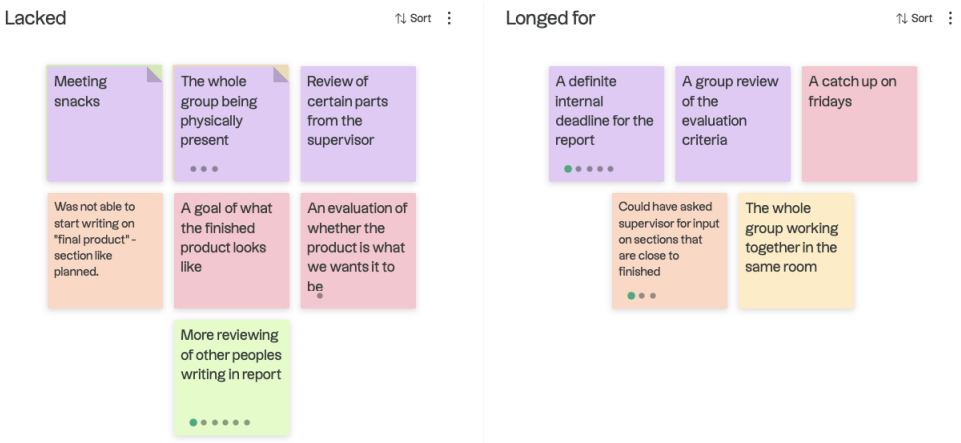


Figure 33: Results from Retrospective Sprint 5: Lacked and Longed for