

Gustavsen, Sjur  
Jørgensen, Mathias  
Nilsen, Sebastian

## Marine Traffic Portal

Website that shows current boats in  
Trondheimsfjorden

Bachelor's thesis in Computer Science  
Supervisor: Tomren, Kjell Inge  
Co-supervisor: Nymo, Christian  
May 2023



Gustavsen, Sjur  
Jørgensen, Mathias  
Nilsen, Sebastian

## **Marine Traffic Portal**

Website that shows current boats in  
Trondheimsfjorden

Bachelor's thesis in Computer Science  
Supervisor: Tomren, Kjell Inge  
Co-supervisor: Nymo, Christian  
May 2023

Norwegian University of Science and Technology  
Faculty of Information Technology and Electrical Engineering  
Department of ICT and Natural Sciences





## ABSTRACT

Accenture recently relocated their office to a new building by the docks in Trondheim. Thus, they experienced an increasing interest among the employees surrounding the subject of boats in the fjord. To compliment for this increasing interest Accenture wanted a modern website that presented all commercial marine traffic outside their office in Throndheimsfjorden. This website where to be showcased on all the different Info screens around the office.

This thesis offers a comprehensive understanding of the approach, development methods, and decisions made by the group throughout the project. It delves into the rationale behind selecting specific technologies and tools, as well as highlights the team's collaborative dynamics and their iterative problem-solving process. The document serves as a valuable resource for future projects and endeavors, showcasing the lessons learned and best practices established during this unique undertaking.

The main objective to the given problem is to create a publicly available web-based interactive map. This map is to display vessels in Trondheimsfjorden visible from Accenture's office at the docks.

During the project, the group employed agile development methodology. With Weekly sprints and meetings with the client. This agile development made the group adaptable to changes demanded by the client, or any other problems encountered. Which proved important for completing the product on time.

To conclude the project, a final and working product was delivered to the client. The product had all the functionality and specification the client requested. The client and students were very satisfied with the final solution provided, and client made the following statement:

"Accenture Trondheim is very satisfied with the final solution, this is not just the opinion of the supervisor, but also the impression I have from the other employees at the office."

## SAMMENDRAG

Accenture flyttet nylig kontorlokalene sine til bryggen i Trondheim. Grunnet dette opplevde de en økt interesse i båttrafikken bland de ansatte. Derav ønsket Accenture et moderne nettsted hvor all kommersiell maritim trafikk utenfor kontoret deres vises. Denne nettsiden skal vises på diverse infoskjermene rundt i kontorlokalene.

Denne rapporten vil gi en omfattende forståelse av tilnærmingen, utviklings metoder, og beslutninger tatt av gruppen gjennom hele prosjektet. Den vil utdype og begrunne valg gjort i forhold til teknologier og verktøy. Samt presentere gruppens samarbeidsmetodikk og prosessen gruppen gikk gjennom for å løse problemet. Dokumentet vil være en verdifull ressurs for fremtidige prosjekter. Ved å vise frem gruppens erfaringer opparbeidet gjennom denne oppgaven.

Oppgavens hovedmål er å lage en offentlig tilgjengelig nettside, hvor informasjon visualiseres på et interaktivt kart. Kartet skal vise fartøy i Trondheimsfjorden som er synlige fra kontorlokalene til Accenture.

Under arbeidet på prosjektet brukte gruppen Agile utviklingsmetodikk. Med ukentlige sprinter og møter med oppdragsgiver. Denne utviklingsmetoden gjorde at gruppen kunne tilpasse seg til endringer klienten ønsket, eller ta høyde for eventuelle problemer som oppsto. Dette hjalp gruppen å fullføre produktet i tide.

Etter prosjektets gang ble et endelig og fungerende produkt levert til oppdragsgiver. Produktet hadde all funksjonalitet og spesifikasjoner kunden etterspurte. Og både kunden og gruppen var fornøyd med det leverte produktet. Etter oppgaven ga kunden følgende tilbakemelding:

"Accenture Trondheim is very satisfied with the final solution, this is not just the opinion of the supervisor, but also the impression I have from the other employees at the office."

# PREFACE

## About

This Project was chosen as developing and deploying a complete fullstackk web application seemed like a good challenge to the group. The prosses of designing and fitting the application based on the customer's needs was a process the group found exiting and interesting. The idea that the final product where to be used and be publicly available was a huge motivation for the group during development. The development prosses consisted of three main phases, planning, creating and lastly publishing.

## Thanks to

The group would like to express their sincere thanks to:

- The client Accenture for providing the topic of this thesis. And a special thanks Christian Nymo for guidance and being broadly available to answer questions and give feedback.
- Our supervisor, Kjell Inge Tomren. Thanks for good advice on what to prioritize and guidance through all stages of development and report writing.

# ASSIGNMENT

## Assignment Text

Create a public website that shows current boats in Trondheimsfjorden. Provision a public cloud environment that hosts the marine traffic website and its functionality. The audience for the website is boat enthusiasts, as well as the Accenture employees. Gather data about traffic and present interesting data points. For example - when was the last time this boat was here?



# CONTENTS

<b>Abstract</b>	i
<b>Abstract</b>	ii
<b>Preface</b>	iii
<b>Assignment</b>	iv
<b>Contents</b>	ix
<b>List of Figures</b>	ix
<b>Code examples</b>	x
<b>Abbreviations</b>	xii
<b>Glossary</b>	xiii
<b>1 Introduction</b>	<b>1</b>
1.1 Background	1
1.2 Objectives	1
1.3 Limitations	2
1.4 Motivation	2
1.5 Goals	2
1.6 Result	2
1.7 Report structure	3
<b>2 Theory</b>	<b>5</b>
2.1 Domain specific theory	5
2.1.1 MMSI	5
2.2 Object-Oriented Programming	5
2.2.1 Coupling and cohesion	5
2.2.2 Coupling	5
2.2.3 Cohesion	5
2.3 Design patterns	6
2.3.1 Observer and observable pattern	6
2.3.2 Singleton pattern	6
2.3.3 State design pattern	6

2.4	Client server communication	6
2.4.1	HTTP	7
2.4.2	REST API	7
2.5	Web scraping	7
2.5.1	The robots.txt file	7
2.6	Accessibility	8
2.6.1	Color blindness	8
2.6.2	Screen readers	8
2.6.3	Alt text	9
2.7	Web application	9
2.7.1	Singe page application	9
2.8	Development	9
2.8.1	Agile development	9
2.8.2	Version control	10
2.8.3	Git	11
2.8.4	Cloud services	11
2.8.5	Containerization	11
2.8.6	Relational database	11
2.8.7	DRY principle	11
2.9	Algorithms	11
2.9.1	Point in Polygon algorithm	11
2.10	Quality assurance	12
2.10.1	Design principles	12
2.10.2	API testing	12
2.11	Technologies	13
2.11.1	SQL	13
2.11.2	Open-source	13
2.11.3	CSS	13
2.11.4	JavaScript	13
2.11.5	TypeScript	14
2.11.6	JSON	14
2.11.7	Geo.JSON	14
2.11.8	Artificial intelgence	14
2.12	CI/CD	14
2.12.1	Continuous Integration (CI)	14
2.12.2	Continuous Deployment (CD)	14
<b>3</b>	<b>Methods</b>	<b>17</b>
3.1	Project Planning and Design Process	17
3.1.1	Preliminary Project Plan	17
3.1.2	Responsibilities	17
3.1.3	Meeting with client	17
3.1.4	Design	18
3.1.5	Wireframe	18
3.2	Technologies	18
3.2.1	Java Spring Boot	18
3.2.2	NextJS	18
3.2.3	MapBox	19

3.2.4 Selenium	19
3.2.5 PostgreSQL	19
3.2.6 Docker	19
3.2.7 Microsoft Azure	21
3.3 Tools	21
3.3.1 Github Actions	21
3.3.2 Postman	22
3.3.3 Prettier	22
3.3.4 Figma	23
3.3.5 PgAdmin4	23
3.3.6 ChatGPT	23
3.3.7 IDE's	23
3.4 Collaboration tools	24
3.4.1 GitHub	24
3.4.2 Jira	24
3.4.3 Confluence	24
3.4.4 Communication	25
3.5 Agile development	25
3.5.1 Roles and work distribution	25
3.5.2 Sprint	25
3.6 Development process	26
3.6.1 User Experience	26
3.6.2 BarentsWatch	26
3.6.3 Backend server and Database	26
3.7 Testing	27
3.7.1 Endpoint testing	27
3.7.2 User testing	27
3.7.3 Lighthouse	28
<b>4 Results</b>	<b>29</b>
4.1 Final result	29
4.2 Gathering additional data for the vessels	29
4.2.1 Problem	29
4.2.2 Possibilities	29
4.2.3 Process	30
4.2.4 Result	32
4.3 Deployment of the backend	32
4.3.1 Problem	32
4.3.2 Possibilities	32
4.3.3 Process	33
4.3.4 Result	34
4.4 Designing the website	35
4.4.1 Problem	35
4.4.2 Possibilities	35
4.4.3 Process	35
4.4.4 Result	36
4.5 Handling the visitor counter for the vessels	37
4.5.1 Problem	37

4.5.2	Possibilities	38
4.5.3	Process	38
4.5.4	Result	39
4.6	Server application	39
4.6.1	Problem	39
4.6.2	Possibilities	39
4.6.3	Process	39
4.6.4	Result	41
4.7	Web application	41
4.7.1	Problem	41
4.7.2	Possibilities	41
4.7.3	Process	42
4.7.4	Result	43
4.8	Backtracking the vessels	43
4.8.1	Problem	43
4.8.2	Process	43
4.8.3	Result	45
4.9	TV Mode	45
4.9.1	Problem	45
4.9.2	Process	46
4.9.3	Result	46
4.10	User experience	47
4.10.1	Wireframes on screens	47
4.10.2	Survey	47
4.10.3	Interviews	47
4.11	Making a public site	47
4.11.1	Problem	47
4.11.2	Possibilities	48
4.11.3	Process	48
4.11.4	Result	48
<b>5</b>	<b>Discussion</b>	<b>49</b>
5.1	Communication	49
5.1.1	With Client	49
5.1.2	Within the group	49
5.2	Development process	50
5.2.1	The plan	50
5.2.2	Agile methodology	50
5.3	Collaboration with Jira and Confluence	50
5.4	Ethics of Web scraping	51
5.5	Social impact	52
5.6	Correct data	52
5.7	General Data Protection Regulation	53
5.8	Environmental impact	53
5.9	Economical impact	53

<b>6 Conclusions</b>	<b>55</b>
6.1 Conclusion . . . . .	55
6.2 Final Product . . . . .	55
6.3 Further work . . . . .	56
<b>References</b>	<b>57</b>
<b>Appendices:</b>	<b>61</b>
<b>A - Requirements Documentation</b>	<b>62</b>
<b>B - Project Plan</b>	<b>66</b>
<b>C - Collaboration Agreement</b>	<b>74</b>
<b>D - Wireframe</b>	<b>78</b>
<b>E - System Documentation</b>	<b>84</b>
<b>F - Survey</b>	<b>94</b>
<b>G - Lighthouse Result</b>	<b>97</b>
<b>H - Feedback</b>	<b>105</b>

## LIST OF FIGURES

3.5.1 Issueboard from sprint 10	26
3.7.1 Example of postman test. This test checks if inArea call has no vessels older than 1 hour	27
4.3.1 Process of deploying a new version of the backend	34
4.4.1 Displays how the site looks with different types of color blindness. Light mode on the left and darkmode on the right	37
4.7.1 What the finished web application looks like	43
4.8.1 Shows how the backtracking was on some vessels when the points was stored in a list of max 50 points	44
4.8.2 Shows how the backtracking is when the point removal was donw via a database query	45
4.11.1 Resources in Azure	48
5.4.1 The infocard showing the information scraped and the credits	52

## LISTINGS

3.1 Dockerfile	20
3.2 GitHub Actions	21
4.1 The scraping functions login method	30
4.2 Collecting the image URL	30
4.3 Example of Vessel object after scrapper has gotten data	32
4.4 Maven dependency causing error	33
4.5 Initial use of the webdrivermanager library	33
4.6 Revised implementation of the chromedriver	34
4.7 Code to check if coordinates are inside a GeoJSON area	38
4.8 Defining the query call for receiving all active vessels in the Area of interest	40
4.9 Update a vessel when new information is retrieved from the API	40
4.10 Gets a list of all	41

## ABBREVIATIONS

List of all abbreviations in alphabetic order:

- **MMSI** Maritime Mobile Service Identity
- **NTNU** Norwegian University of Science and Technology



## GLOSSARY

- **API** Application Programming Interface
- **CD** Continuous Deployment
- **CSS** Cascading Style Sheets
- **HTML** Hypertext Markup Language
- **HTTP** Hypertext Transfer Protocol
- **IDE** Integrated Development Environment
- **IP** Internet Protocol Address
- **JS** JavaScript
- **JSON** JavaScript Object Notation
- **MVP** Minimum Viable Product
- **REST** Representational State Transfer
- **TCP** Transfer Control Protocol
- **TS** TypeScript
- **SQL** Structured Query Language
- **AIS** AIS is a system that automates ship identification and tracks its movement. It consists of a transceiver, which is a combined transmitter and receiver, that sends out information about the ship's identification, position, speed, and course. Additionally, it can also transmit information about the vessel's type, cargo, destination, and other relevant details.
- **Backend** This term refers to the software infrastructure responsible for managing business logic and data storage. It is not directly accessible to the user.
- **DevOps** DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to streamline the process of delivering software and services, aiming to shorten development cycles, increase deployment frequency, and improve collaboration, efficiency, and overall quality.

- **Frontend** This term refers to the graphical user interface (GUI), which is the software stack that enables user interaction with a system.
- **Repository** This is a storage location for software code and related resources, such as documentation and configuration files. It's a centralized system that enables developers to collaboratively manage and track changes to code over time, making it easier to maintain and improve software projects.
- **Scrape/Crawl** Scraping and crawling are techniques used in web data extraction and retrieval. They involve automated processes to collect data from websites.
- **Wireframe** This is an illustration of a webpage's interface and layout. It is used to help visualize the concept.

## INTRODUCTION

### 1.1 Background

Accenture recently relocated their office premises to the waterfront area in Trondheim. This move sparked a growing fascination with nautical activity among the employees. Consequently, they sought a convenient and efficient method to access information about the vessels outside their office. They concluded that the best way for all to get this information was to get a custom website running on the info screens placed around in the office.

At present, employees rely on various websites on their computers or smartphones to gather vessel information. However, this process is inefficient, as different sites offer varying levels of detail, often requiring individuals to visit multiple sites to obtain the desired data. The client believes that featuring the custom website on the information screens would not only simplify access but also optimize the use of these screens, which are currently underutilized.

### 1.2 Objectives

The team collaborated with Accenture Trondheim, an IT service and consulting firm. Accenture has multiple TV screens throughout their office and sought to display engaging content on them. They determined that a website showcasing ships located in the fjord outside their office would be ideal.

Accenture tasked the group with developing this website and ensuring it was fully operational for public use. The team had the freedom to choose their approach for completing the task. Accenture's requirements specified that the site should be publicly accessible, with vessel updates occurring in real-time. Another crucial factor was that the site would be displayed on TV screens, necessitating an automated operation with no user interaction

### 1.3 Limitations

Throughout this semester, the students were concurrently enrolled in another subject alongside their bachelor thesis. This factor significantly limited the time available for the project. The parallel subject was intensive, spanning 10 weeks and occupying 3-4 days per week. Consequently, it was challenging for students to allocate time for their thesis during this period. As the subject concluded with an exam at the end of March, students had to dedicate their time to exam preparation, leaving no room for the thesis.

### 1.4 Motivation

The primary motivation for this project stemmed from the freedom provided in devising and developing the application. With this autonomy, the group felt compelled to deliver a functional product they could take pride in. Knowing that Accenture would implement the completed project in their office further boosted motivation. Accenture also assigned a supervisor to the students, who proved to be incredibly helpful and consistently positive about the project's development. This support fostered a sense of progress and bolstered motivation.

### 1.5 Goals

- Find a source for location data of the vessels within the fjord.
- Create a backend able to fetch the data of the vessels and save them to a database.
- Create endpoints in the backend where one can fetch data stored.
- Develop a web-based application that can display the vessels on a map.
- Be able to gather interesting data on the vessel. Like Images, county of origin, and so on.
- Deploy application in an cloud environment.

### 1.6 Result

At the end of the project a complete and finalized product was handed over to the client. The final application was a working site that had all the functionalities requested by the client. The client was satisfied with the result.

## 1.7 Report structure

This thesis report is split into extensive 6 parts

### **Chapter 1 Introduction**

Gives an overview of the problem domain and the vision for the project at the beginning

### **Chapter 2 Theory**

Describes Presents the required theoretical background of this thesis. Including definitions necessary to understand the thesis.

### **Chapter 3 Method**

Provides insight to the choices made throughout the project.

### **Chapter 4 Results**

Presents the results from the project

### **Chapter 5 Discussion**

Discussion points not touched in chapter 4

### **Chapter 6 Conclusion**

Provides a conclusion to the thesis and recommendations for future work



## 2.1 Domain specific theory

### 2.1.1 MMSI

An MMSI is a unique identification number temporarily assigned by the object's current flag state. It serves as the international telephone number for various types of maritime objects, such as vessels, fixed offshore installations, mobile units, maritime aircraft, coast stations, and more. [1]

## 2.2 Object-Oriented Programming

Object oriented programming is a programming model that organizes software design around objects, rather than functions and logic. An object can be defined as an instance of a class, encapsulating related data fields (attributes) and methods (functions or behaviors) within it. [2]

### 2.2.1 Coupling and cohesion

In object-oriented programming, it is important to think about coupling and cohesion when writing high quality code. High-quality code is characterized by several aspects, one of which is the ease with which features can be changed without causing unnecessary work. It can be achieved by writing code which has low coupling and high cohesion. [3]

### 2.2.2 Coupling

The level of interdependence between modules is referred to as coupling, and a well-designed software system should aim to minimize it. [3]

### 2.2.3 Cohesion

Cohesion measures the functional relationship between elements in a module, determining how well they work together to accomplish a specific task. Essentially,

cohesion acts as a glue that binds the modules elements together. A strong software design will prioritize high cohesion. [3]

## 2.3 Design patterns

A design pattern in software engineering refers to a frequently encountered problem in software design, for which a generalized, reusable solution is provided. Rather than a complete design that can be immediately translated into code, it serves as a guideline or blueprint for addressing the problem and can be adapted to various scenarios. [4]

### 2.3.1 Observer and observable pattern

The Observer pattern is classified as a behavioral design pattern that outlines how objects communicate with each other, specifically between observables and observers. An observable is an object that updates its observers about any changes made to its state. The Observer are objects that wishes to be notified when a state of another object changes. [5]

### 2.3.2 Singleton pattern

Singleton design pattern ensures that there is only one object of its kind that exists. The singleton object must provide a global access point to get the instance of the object. [6]

### 2.3.3 State design pattern

The intent of the state design pattern is to let an object alter its behavior when the object's state changes. In computer programming, the state pattern is utilized to encapsulate different behaviors for a single object, depending on its internal state. [7]

## 2.4 Client server communication

The client-server model is a widely used distributed computing architecture where servers provide services or resources to clients upon request. Servers are typically categorized into three types: application servers, database servers, and web servers. Application servers run applications, often connecting client requests to database servers. Database servers store, retrieve, and manage data in a database. Web servers deliver web pages to clients through the HTTP protocol.

Clients send requests to servers and receive responses back, using protocols such as TCP/IP, HTTP, or REST. Servers run on dedicated hardware or virtual machines and can handle multiple requests simultaneously, using advanced techniques such as threading or pooling.



Clients run on user devices and initiate communication sessions with servers by establishing network connections. The model offers several benefits, including scalability, reliability, security, and maintainability. Examples include email, network printing, and the World Wide Web. [8]

### 2.4.1 HTTP

HTTP or Hypertext Transfer Protocol is a protocol in the application layer. It is the foundation of communication on the World Wide Web, and it is mainly used for communication between the server and the client to transfer files. [9]

### 2.4.2 REST API

API or Application Programming Interface defines how devices or applications can connect and communicate with one another. REST or Representational State Transfer is an architecture style, therefore the REST API is an API that follows the design principles of REST. [10]

## 2.5 Web scraping

Web scraping is a technique used to extract data from websites. It involves using software to automatically collect information from web pages and save it in a structured format, such as a spreadsheet or a database. The data can be anything that is publicly available on the website, such as prices, product descriptions, reviews, or contact details. Web scraping can be done manually, by copying and pasting data from web pages, or automatically, by using specialized tools or scripts. It is often used for market research, competitive analysis, content aggregation, or data analysis. However, web scraping may be illegal or unethical if it violates the website's terms of use or privacy policy, or if it harms the website's performance or integrity. [11]

### 2.5.1 The robots.txt file

The robots.txt file is a text file that is placed on a website to provide instructions to web crawlers, such as search engine robots, on what pages or sections of the website should or should not be crawled or indexed. It is often used to prevent web scraping tools from accessing certain pages or data, in order to protect the website's content or resources, or to enforce legal or ethical restrictions. The robots.txt file can contain rules that specify which user agents are allowed or disallowed, which directories or files are allowed or disallowed, and how often or at what time the crawler can access the site. Web scrapers should always check the robots.txt file of a website before scraping it, and should comply with the rules and restrictions stated in the file, in order to avoid legal or ethical issues. [12]

## 2.6 Accessibility

Accessibility means treating everyone equally and giving them the same opportunities, regardless of their abilities or circumstances. Just like it's unacceptable to exclude someone in a wheelchair from a physical building, modern public buildings have wheelchair ramps or elevators to ensure equal access. Similarly, it's not right to exclude someone with a visual impairment from a website. We're all unique individuals, but we share the same human rights. This can open up new markets for your products and services, as you'll be able to serve customers who might not have been able to access your offerings otherwise. Having an accessible website benefit everyone, and it is important to think of this in every aspect of the development, from the heading structure to the formatting, layout, and other uses of visual media. [\[13\]](#) [\[14\]](#)

There are several rules to ensure sufficient accessibility on a public website. These are the most essential.

### 2.6.1 Color blindness

When designing a website for color blind users, there are several technical solutions that can be implemented to ensure accessibility. The first step is to use a color palette that takes into account the most common types of color blindness, such as red-green or blue-yellow deficiencies. This can be achieved by using color contrast analyzers to check the color combinations and ensure that they meet the Web Content Accessibility Guidelines (WCAG) criteria.

Another solution is to provide alternative text descriptions for non-textual content such as images and charts, which may convey important information that color-blind users might miss. In addition, it is important to avoid relying solely on color to convey information or to differentiate between elements, as this can cause confusion for users with color blindness. Instead, use other visual cues such as patterns, shapes, or icons to distinguish between different elements.

Finally, it is important to test the website with real color-blind users to ensure that the design is effective. There are online tools and browser extensions that simulate different types of color blindness, which can be used to test the website's accessibility. By following these technical solutions and testing with real users, designers can create websites that are accessible to all users, including those with color blindness. [\[15\]](#)

### 2.6.2 Screen readers

Screen readers are software programs used by people with visual impairments to access and navigate the web. To ensure web accessibility for screen reader users, several theories can be applied in web development. One such theory is the use of semantic HTML, which involves utilizing appropriate HTML tags to help screen readers navigate and interpret the content accurately. Another theory is the use of Accessible Rich Internet Applications (ARIA), which is a set of attributes that can

be added to HTML elements to provide additional information to screen readers. Additionally, focus management is crucial for keyboard navigation, and testing with screen readers and other assistive technologies is essential for identifying and resolving accessibility issues. Overall, these theories play a critical role in creating accessible web experiences for screen reader users. [16]

### 2.6.3 Alt text

Alternative text or alt text for short. Is a descriptive text that expresses the content/meaning of a visual item. Examples of such items can be images and videos. Alt text is commonly used on websites to express an items content either for screen readers or if the item fails to load. [17]

## 2.7 Web application

Web applications are software programs accessed through a web browser interface over the internet. They are designed for various purposes, including e-commerce, data management, and communication. These applications can be accessed through any browser, but some may have specific requirements for optimal performance.

Web apps play a crucial role in modern computing, providing users with a convenient and accessible way to interact with software over the internet. They are platform-independent and offer several advantages, such as easy accessibility and flexibility. However, they may also have certain limitations, such as browser compatibility issues or slower performance. Despite these limitations, web applications are essential for a wide range of use cases, from small-scale individual needs to large-scale organizational needs, and they continue to evolve to meet changing user demands. [18]

### 2.7.1 Single page application

A single page application (SPA) is a web application that operates within a single web page. The content is updated dynamically as the user interacts with the page, without loading new pages from the server. This creates a smoother user experience without interruptions. SPAs are commonly built with JavaScript frameworks like React, Angular, or Vue. [19]

## 2.8 Development

### 2.8.1 Agile development

Agile methodology is a project management and software development approach that emphasizes iterative processes to deliver customer value more efficiently. Unlike traditional "big bang" launches, an agile team breaks down work into small, easily consumable increments. The team continuously evaluates requirements, plans, and outcomes, allowing them to quickly adapt to change. This approach

enables teams to respond to feedback and changes in a more timely and efficient manner. [20]

### 2.8.1.1 Scrum

Scrum is a framework rooted in empiricism and lean thinking, and decisions are based on observations gained from experience. The latter focuses on eliminating waste and prioritizing essentials. Scrum is a heuristic framework that facilitates continuous learning and adjustment to fluctuating factors. It recognizes that teams don't have complete knowledge at the start of a project and are likely to evolve through experience. With built-in mechanisms for re-prioritization and short release cycles, Scrum enables teams to naturally adapt to changing conditions and user requirements while consistently improving their output.

The framework of scrum provides a comprehensive guide for scrum teams to create and deliver a product or service, encompassing a set of values, principles, and practices. It outlines the responsibilities of the scrum team members, defines the key "artifacts" used to create and refine the product, and details the scrum ceremonies that facilitate the team's progress through the work.

A scrum team is a small and agile group focused on delivering committed product increments. While its size is typically compact, it is still capable of completing a significant amount of work within a sprint. In order to function effectively, a scrum team must have three key roles: the product owner, the scrum master, and the development team.

The product owner's primary responsibility is to understand the business, customer, and market requirements, and then prioritize the work for the engineering team accordingly. The Scrum master serves as the champion of the Scrum framework within the team. They coach the team, the product owner, and the business on the Scrum process and continuously improve its implementation. The development team drives the plan for each sprint and consists of members with varying skill sets who cross-train each other to avoid any bottlenecks in the delivery of work.

A sprint refers to the designated period in which the scrum team collaborates to complete a product increment. While a two-week sprint is commonly used, some teams may opt for a shorter duration of one week for easier project scoping. [21]

### 2.8.2 Version control

Version control or source control is a method of keeping track and managing changes made to a file, though mostly used for software code. This assists development teams in managing changes to files over time. Helping the team collaborate quicker and easier. If a mistake were to happen. The developers can go back in time to a specific working version of the file and compare the two, or rollback to the working version completely.

### 2.8.3 Git

Git is an open-source distributed version control system that's both free and designed to manage projects of all sizes, from small to extremely large, with great speed and efficiency. [22]

### 2.8.4 Cloud services

Cloud services is a broad term used to describe a variety of on-demand services delivered over the internet to businesses and customers. These services are aimed at offering convenient and cost-effective access to resources and applications without requiring internal hardware or infrastructure. Whether or not employees are aware of it, most use cloud services during their workday, from email checking to collaborating on documents. Using a cloud server allows one to run an application without setting up and maintaining a physical server for the client. [23]

### 2.8.5 Containerization

Containerization refers to a deployment procedure for software that involves packaging an application's code together with all necessary files and libraries required for its operation on any infrastructure. In the past, it was necessary to install a software package that matched the operating system of your computer to use any application. With containerization, a single container can be developed that operates across all device types and operating systems. [24]

### 2.8.6 Relational database

A relational database is a database that enables storage and retrieval of related data points. The relational model forms the basis of these databases, offering a clear and intuitive method of representing data through tables. Each row in a relational database table represents a record, and it has a unique ID called the key. The table columns contain data attributes, and each record typically has a value for every attribute, facilitating the establishment of relationships among data points. [25]

### 2.8.7 DRY principle

DRY stand for don't repeat yourself. The principle is a practice in software development that recommends the engineers to not exactly that. Meaning one should write something once, and only once. [26]

## 2.9 Algorithms

### 2.9.1 Point in Polygon algorithm

The Point in Polygon algorithm is a method for determining whether a given point lies inside, outside, or on the boundary of a polygon. It works by tracing a line from the point to a point outside the polygon, and counting the number of times

the line intersects with the polygon's edges. If the number of intersections is odd, the point is inside the polygon; if it's even, the point is outside the polygon. This algorithm can be used in a variety of applications, such as geolocation and spatial analysis. [27]

## 2.10 Quality assurance

### 2.10.1 Design principles

Design principles are a set of guidelines on how to create accessible and pleasant design and user interfaces. These guidelines are often referred to as the UX principles. The principles provide a framework where designers can seek direction and lean against when in doubt. Some of key principles in web development are:

- **Consistency:** Maintain a consistent design across your website, including fonts, colors, and layout elements.
- **Hierarchy:** Organize information and elements in a clear hierarchy, prioritizing content and functionality based on their importance.
- **Context:** Consider the circumstances of which the product is to be used. Ask yourself what device the user use will and where might they be.
- **Simplicity:** Keep designs simple and avoid unnecessary complexity. Focus on the essential elements and functions, and eliminate distractions.
- **White space:** Use white space (empty space) strategically to create visual separation between elements, improve readability, and reduce cognitive load.
- **Navigation:** Create intuitive and easy-to-use navigation systems that help users find their way around your website.
- **Feedback:** Provide users with feedback on their interactions, such as indicating which buttons have been clicked or which form fields are required.
- **Accessibility:** Ensuring that the site is accessible to and usable for as many people as possible

By following these principles, one can create a user-friendly website that meets the needs of the users. [28]

### 2.10.2 API testing

API testing is a type of software testing that helps programmers verify and analyze an API. Making it easier to check the API's functionality, security, and performance. By making automated tests that calls the API's endpoints and verifies/presents the values returned by the API. This allows for an easy and quick check on all the endpoints whenever a change is made to the API's code [29]

## 2.11 Technologies

### 2.11.1 SQL

SQL is a programming language that facilitates storing and manipulating information within a relational database. This type of database organizes data in tabular format, using rows and columns to represent distinct data attributes and the interrelatedness of data values. By utilizing SQL statements, users can perform various operations such as storing, updating, removing, searching, and retrieving data from the database. [30]

#### 2.11.1.1 Hibernate

Hibernate is an object-relational mapping (ORM) framework for Java that simplifies the development of database-driven applications. It maps Java objects to database tables and provides a way to query and manipulate data from those tables using object-oriented programming techniques. Hibernate automates much of the repetitive database-related coding tasks, such as opening and closing database connections and converting data between Java and SQL data types. This makes it easier for developers to focus on the business logic of their application, rather than the low-level details of database access. [31]

### 2.11.2 Open-source

Open source software (OSS) refers to software that comes with its source code, allowing users to access, modify, and distribute it with the same rights as the original software. The source code represents the program's inner workings and the instructions that developers use to manipulate its behavior. Access to source code enables programmers to modify the software by adding new features, making changes, or fixing any issues with the program's functionality. Typically, OSS includes a license that grants programmers the freedom to customize the software to their specific requirements and determine how the software can be distributed. [32]

### 2.11.3 CSS

Cascading Style Sheets or CSS is a type of stylesheet language utilized to define the visual appearance of HTML or XML documents. It specifies how individual elements should be displayed on a range of media, such as computer screens, paper, audio devices, or other forms of output. [33]

### 2.11.4 JavaScript

JavaScript is a programming language that is popular among web developers for its ability to enhance user interaction and create more dynamic web pages, applications, servers, and games. Typically used in conjunction with HTML and CSS, JavaScript complements CSS by providing the ability to create interactive features that CSS cannot achieve on its own, while CSS focuses primarily on formatting

HTML elements. Despite being a lightweight language, JavaScript plays a vital role in web development due to its versatility and broad range of applications. [34]

### 2.11.5 TypeScript

TypeScript is a programming language that builds upon JavaScript. It allows users to add extra syntax(types), on top of JavaScript. [35]

### 2.11.6 JSON

JSON, or JavaScript Object Notation, is a lightweight data exchange format designed to be readable and writable by humans, while also being easily parsed and generated by machines. This text-based format is commonly utilized as an alternative to XML for transmitting data between a server and a web application. While JSON is based on a subset of the JavaScript programming language, it can be utilized with a variety of other programming languages as well. [36]

### 2.11.7 GeoJSON

GeoJSON is a JSON-based format used to encode geographic data structures, frequently utilized for the representation of geographic features, such as points, lines, polygons, and multi-geometries, along with their corresponding attributes. [37]

### 2.11.8 Artificial intelligence

#### 2.11.8.1 Prompt

In the context of AI chatbots, a prompt is the input message or question the user provides. The chatbot then analyzes and responds to it. [38]

## 2.12 CI/CD

### 2.12.1 Continuous Integration (CI)

Continuous Integration (CI) is a DevOps development practice that streamlines the process of integrating code changes from multiple contributors. It achieves this by consolidating code updates from various sources into a single software project. From this unified repository, builds or tests can be executed, enabling faster validation and deployment. This approach ultimately enhances collaboration and accelerates software development processes. [39]

### 2.12.2 Continuous Deployment (CD)

Continuous Deployment (CD) is a DevOps development practice that automates all stages following the completion of code writing. This process ensures that the code is automatically built and deployed to the production environment once specific requirements are met. By streamlining the software development lifecycle,



CD fosters efficiency and minimizes the time it takes to release new features and improvements. [40](#)



## **3.1 Project Planning and Design Process**

Throughout this phase of the project, the team conducted comprehensive planning and preparation efforts.

### **3.1.1 Preliminary Project Plan**

Prior to commencing the project, the team formulated a preliminary project plan. This plan supplied the group with well-defined routes, guidelines, and deadlines to adhere to throughout the project's development. The decision was made for the team to adopt a weekly sprint approach, ensuring a steady workflow and promoting consistent effort from the outset. The specifics of the preliminary project plan can be found in Attachment B.

### **3.1.2 Responsibilities**

The team established a collaborative agreement at the onset of the project. This agreement served to structure the group by assigning roles and delineating areas of responsibility for each member. These roles were designed to enhance organization within the team and provide a sense of belonging for all members. The roles given were team leader, document manager and quality assurance. Although each member had a designated area of responsibility, they actively supported one another across all aspects of the project. The collaborative agreement can be found in Attachment C.

### **3.1.3 Meeting with client**

The team engaged with the client early in the planning phase. The purpose of this meeting was to clarify essential requirements for the solution that the team was to develop. Throughout the meeting, uncertainties surrounding delivery and technology requirements were thoroughly discussed and addressed

### 3.1.4 Design

#### 3.1.4.1 Research

A significant component of the planning process involved conducting research. The team examined existing platforms, such as [MarineTraffic](#) and [Kystverket](#), to gain insights into their page layouts and methods of displaying vessels on maps. Additionally, the team explored other websites for further design inspiration.

#### 3.1.5 Wireframe

Following the research and collection of design inspiration, the team initiated the wireframe development. Collaboratively, the team utilized Figma to create the wireframes. Figma was chosen due to its cost-free availability and its status as one of the most widely employed tools for wireframe creation and high-level sketching. Furthermore, it allows multiple users to work on the same file concurrently, enabling seamless collaboration for the entire team. The primary design objective was to develop an aesthetically appealing site that also offered ease of use. To accomplish this, the team applied the principles of design referenced in section [2.10.1](#)

The web application's primary use case is for display on large TV screens within the office setting. Consequently, the design emphasized readability on larger screens and visibility from a considerable distance. The color themes were inspired by ocean hues and Kystverket's visual elements. The wireframes can be found in [appendix D](#)

## 3.2 Technologies

### 3.2.1 Java Spring Boot

Spring Boot was the chosen framework for developing a backend server for the application. It was chosen because of all the functionality it included. Another major reason was the developer experience. Java is a language all members of the team are highly comfortable using. Because of this experience, the team could rapidly start the development proses.

From a technical standpoint Java and Spring Boot was also a good choice since Java is a cross-platform language, meaning the code will work on MacOS, Windows and Linux despite being compiled on another operating system. This makes the application flexible and futureproofs the application by easily being deployed on different systems.

### 3.2.2 NextJS

The main requirement of the project was to create a web application. Next.js is a React framework that gives the possibility to create building blocks to create a web application. By using Next.js the group could create the site part by part with React, and use additional tools given by Next.js to make the process easier.

### 3.2.2.1 React

React is a free and open-source front-end JavaScript library. It is used for building user interfaces via the usage of components. React also has extensive support for TypeScript. With this functionality the team agreed that React was a good choice for a UI framework.

### 3.2.2.2 Typescript

The team wanted to use TypeScript in the project as it offers several advantages. Some of these advantages including improved code readability, maintainability, and error prevention. Typescript's has static typing this feature enables early detection of type-related issues. TypeScript's advanced language features, such as interfaces, generics, and decorators, allow for better structuring and abstraction, making it easier to scale and collaborate on projects.

## 3.2.3 MapBox

Upon researching various map APIs, the team ultimately selected MapBox GL JS as the map API for the project. MapBox offered exceptional customization options on their maps, such as adjusting map colors and detail level. This allowed for the removal of all unnecessary details and highlights from the map, ensuring a clean look for the application. MapBox also supplied all the requisite functionality the team sought in a map API, including the creation of markers and automatic camera panning to specific points on the map.

## 3.2.4 Selenium

Selenium, is an open source Java library, that was used to obtain vessel images by web scraping the [Ship-Info](#) website. Since the website's `/.robots.txt` endpoint was unspecified, the team reached out to the website to ask for permission to scrape it. Ship-info agreed to provide with login credentials and permission to crawl the site, on the condition that accreditation was given. They were also insistent that credit must be given to all the photographers when displaying their images.

## 3.2.5 PostgreSQL

A PostgreSQL database, herby referred to as Postgres, is an open source relational database management system. Its high compatibility with a wide range of operating systems, diverse database models, and programming languages. Postgres is both robust and reliable, and offering support for complex data types, full-text search, and geospatial data processing. This makes it an ideal choice for the project.

## 3.2.6 Docker

The utilization of Docker facilitated the process of containerizing the backend of the project. Selenium, a library used in the project (See sec. [3.2.4](#)), necessitates

the installation of chromedriver on the operating system. However, Docker circumvents this issue by ensuring that all the necessary project dependencies, including libraries and the chromedriver, are automatically and accurately installed.

**Listing 3.1:** Dockerfile

```

1 # Base image with Maven and JDK 17
2 FROM --platform=linux/amd64 maven:3.9.0-eclipse-temurin-17
3
4 # Google Chrome installation
5 #ARG CHROME_VERSION
6 RUN apt-get update -qqy \
7     && apt-get -qqy install gpg unzip curl \
8     && wget -q -O -
9     https://dl-ssl.google.com/linux/linux_signing_key.pub
10    | apt-key add - \
11    && echo "deb http://dl.google.com/linux/chrome/deb/
12    stable main" >
13    /etc/apt/sources.list.d/google-chrome.list \
14    && apt-get update -qqy \
15    && if [ -z "$CHROME_VERSION" ]; then \
16        apt-get -qqy install google-chrome-stable; \
17    else \
18        apt-get -qqy install
19        google-chrome-stable=$CHROME_VERSION; \
20    fi \
21    && rm /etc/apt/sources.list.d/google-chrome.list \
22    && rm -rf /var/lib/apt/lists/* /var/cache/apt/* \
23    && sed -i 's/"$HERE\/chrome"/"$HERE\/chrome"
24    --no-sandbox/g' /opt/google/chrome/google-chrome
25
26 # ChromeDriver installation
27 RUN CHROME_DRIVER_VERSION=$(curl -sS
28     https://chromedriver.storage.googleapis.com/
29     LATEST_RELEASE) \
30     && wget -q -O /tmp/chromedriver.zip
31     https://chromedriver.storage.googleapis.com/
32     $CHROME_DRIVER_VERSION/chromedriver_linux64.zip \
33     && unzip /tmp/chromedriver.zip -d /opt \
34     && rm /tmp/chromedriver.zip \
35     && mv /opt/chromedriver
36     /opt/chromedriver-$CHROME_DRIVER_VERSION \
37     && chmod 755 /opt/chromedriver-$CHROME_DRIVER_VERSION \
38     && ln -s /opt/chromedriver-$CHROME_DRIVER_VERSION
39     /usr/bin/chromedriver
40
41 # Expose port for the application
42 EXPOSE 8080
43
44 # Copy the JAR file and set the entrypoint
45 COPY target/MarineTraffic-0.0.1-SNAPSHOT.jar app.jar
46 ENTRYPOINT ["java", "-jar", "/app.jar"]

```

### 3.2.7 Microsoft Azure

Microsoft Azure is a cloud computing platform. The team chose azure as its deployment environment. For the project Azure is used for hosting all parts of the project (Database, Backend and frontend), this allowed for excellent communication and integration between all layers of the application.

## 3.3 Tools

### 3.3.1 Github Actions

GitHub Actions is GitHub's CI/CD (Continuous Integration and Continuous Deployment) service. Since the team had already chosen GitHub as their platform, GitHub Actions became a natural selection. GitHub Actions enables teams to automate their workflows for building, testing, and deploying software. Throughout the project, the team employed GitHub Actions to deploy the frontend to Azure whenever the main Git branch was updated. The code in listing [3.2](#) demonstrates this process. This action first runs 'npm install' and then 'build' to compile the application from the source code. Upon completion, it compresses the build for faster uploading and transfers the files. After uploading, the compressed files are deployed to the webapp platform in Azure.

Listing 3.2: GitHub Actions

```
1 name: Build and deploy Node.js app to Azure Web App -
   MarineTrafficFrontend
2
3 on:
4   push:
5     branches:
6       - main
7   workflow_dispatch:
8
9 jobs:
10  build:
11    runs-on: ubuntu-latest
12
13    steps:
14      - uses: actions/checkout@v2
15
16      - name: Set up Node.js version
17        uses: actions/setup-node@v1
18        with:
19          node-version: '18.x'
20
21      - name: npm install, build, and test
22        run: |
23          export NEXT_PUBLIC_MAPBOX_TOKEN=${{
24            secrets.NEXT_PUBLIC_MAPBOX_TOKEN }}
25          npm install
26          npm run build --if-present
27
28      - name: zip all files for upload between jobs
29        run: zip next.zip /* .next -qr
```

```
29
30   - name: Upload artifact for deployment job
31     uses: actions/upload-artifact@v2
32     with:
33       name: node-app
34       path: next.zip
35
36 deploy:
37   runs-on: ubuntu-latest
38   needs: build
39   environment:
40     name: 'Production'
41     url: ${ steps.deploy-to-webapp.outputs.webapp-url }
42
43   steps:
44     - name: Download artifact from build job
45       uses: actions/download-artifact@v2
46       with:
47         name: node-app
48
49     - name: 'Deploy to Azure Web App'
50       id: deploy-to-webapp
51       uses: azure/webapps-deploy@v2
52       with:
53         app-name: 'MarineTrafficFrontend'
54         slot-name: 'Production'
55         publish-profile: ${ secrets.AZUREAPPSERVICE_PUBLISHPROFILE_
56           D6D0780E31D34C8A84E636850018D1D7 }
57         package: next.zip
58
59     - name: Delete zip file
60       run: rm next.zip
```

### 3.3.2 Postman

Postman is a platform where developers can design, build, and test APIs. During the project the team used postman for creating tests on the API. In postman it is easy to create a set of tests and then run them all with a click. This assisted the team in checking all the endpoints of the backend part of the project. See figure [3.7.1](#) for example.

### 3.3.3 Prettier

Prettier was the teams chosen code formatter and was taken in use for the development of the Frontend part of the application. This was taken in use to enforce a consistent style through the project. In prettier the team defined a ruleset on how the project were to be formatted. This helped the team to maintain a more readable code and making it easier when merge conflicts happened.



### 3.3.4 Figma

The team selected Figma as their tool of choice for designing and creating wireframes. Figma was chosen for its accessibility and collaborative capabilities. As a cloud-based platform, Figma enabled all team members to access and edit wireframes using either the Figma application or directly within a browser. Figma also supports real-time collaboration, allowing multiple team members to work on and discuss the same file simultaneously, even when working remotely. Utilizing Figma, the team was able to develop everything from basic mockups to polished wireframes.

### 3.3.5 PgAdmin4

For project database management and administration, the team opted for pgAdmin4 as their preferred tool. pgAdmin4 is an open-source administration and management tool specifically designed for PostgreSQL databases [41]. It provided the team with an intuitive interface for managing the database. The tool enables the execution of SQL queries and visualizes the query results, simplifying the team's database analysis and management process. The team's database is relatively small, comprising only two tables. Consequently, using a visualization tool proved to be a fast and straightforward method for managing the tables.

### 3.3.6 ChatGPT

Throughout the development process, the team employed Chat GPT (Generative Pre-trained Transformer) as a tool to assist in debugging code. Chat GPT is an artificial intelligence-powered chatbot. When the team encountered difficulty understanding or resolving errors within the backend or frontend code, they utilized Chat GPT as a supportive tool to identify and address the issue. To accomplish this, the team provided the chatbot with the error message and created a prompt that included the problematic code and other relevant project information. The chatbot then responded with suggestions to rectify the error.

### 3.3.7 IDE's

#### 3.3.7.1 VSCode

Visual Studio Code (VS Code) is a lightweight yet powerful code editor. It was the preferred choice for two team members, thanks to its extensive selection of available extensions, which allows users to customize and enhance its functionality as desired. When working on large projects encompassing various file types, VS Code can swiftly open, view, and edit these files, while still supporting formatting and autocompletion features.

#### 3.3.7.2 Webstorm

JetBrains WebStorm is a highly regarded integrated development environment (IDE) for web development. With its intelligent code assistance and smart navigation features, WebStorm allows developers to efficiently code in HTML, CSS,

and JavaScript, among other languages. It also comes with an array of powerful features, such as integrated version control, advanced debugging tools, and support for popular frameworks like Angular and React. Thanks to its seamless integration with other JetBrains tools, WebStorm is an excellent choice for teams that want a comprehensive web development environment.

### 3.3.7.3 IntelliJ

JetBrains IntelliJ IDEA is a widely used integrated development environment (IDE) that supports a multitude of programming languages and frameworks. It is particularly popular among Java developers, thanks to its advanced code analysis and debugging features. With its extensive customization options and plugin ecosystem, IntelliJ IDEA is known for providing a highly tailored development experience, and therefore was a natural fit for our team. The team used several plugins to compliment the IDE's features such as Sonarlint, Docker and Prettier.

## 3.4 Collaboration tools

### 3.4.1 GitHub

GitHub was the team's chosen code collaboration tool. Two separate Git repositories were created on GitHub, one for frontend and one for backend. With the use of version control via GitHub the team could streamline their collaboration on the same code-basis.

### 3.4.2 Jira

The team selected Jira for project management, which assisted in tracking tasks that needed completion and those already accomplished. Within Jira, the team created a project containing a backlog of tasks to be completed. As new features or bugs were identified, they were added as issues in the backlog. Each week, a new sprint commenced, and the team chose tasks from the backlog to address during that week. Throughout the sprint, all relevant issues were displayed on an issue board. Team members could assign tasks to themselves (or others), and everyone maintained an overview of the ongoing work and pending tasks at any given time.

### 3.4.3 Confluence

Throughout the project, the team utilized Confluence in tandem with Jira for creating sprint retrospectives and documenting meeting notes. This seamless integration was facilitated by both platforms being developed by the same company, Atlassian. As a result, Confluence emerged as an optimal choice for composing sprint retrospectives, which served as the primary reason for its selection.

### 3.4.4 Communication

During the development process, the team employed a variety of communication methods to facilitate both internal and external interactions. For internal communication, Facebook Messenger was used for informal and quick exchanges, such as when a member was running late for a meeting. Microsoft Teams served as another platform for internal discussions, primarily for sharing simple files like Word documents and images.

For external communication, email was the primary channel, used to schedule meetings with clients and supervisors. A Microsoft Teams chat was also established, which included all group members and the external supervisor from Accenture. This chat enabled quicker communication between the team and the client, and was particularly useful when either party had questions that needed addressing before scheduled weekly meetings.

## 3.5 Agile development

Early in the project, the team chose to adopt an Agile work methodology. This approach allowed the team to adapt to unforeseen circumstances and maintain flexibility in response to changes.

### 3.5.1 Roles and work distribution

During the planning phase, it was decided that all members would contribute equally to every aspect of the project. However, to maintain organized, specific team roles were assigned. Sjur Gustavsen was designated as the team leader, Mathias Jørgensen took on the role of document manager, and Sebastian Nilsen was appointed as quality assurance. The team leader's responsibilities included serving as the primary point of contact with the client and supervisor. The document manager's primary responsibility was to oversee all project-related documentation. Meanwhile, the quality assurance role involved ensuring high code quality throughout the project.

### 3.5.2 Sprint

The team operated in one-week sprints, with each sprint commencing with a meeting on Monday. During these meetings, the team determined which tasks/issues to tackle during the sprint, allowing them to prioritize tasks based on urgency or client feedback. On the last day of the sprint (Friday), the team conducted a sprint retrospective. This review session summarized each member's work and progress during the sprint. The team also discussed what went well and areas for improvement. In this forum, members could express any dissatisfaction with the current working methods, allowing for swift resolution before issues could negatively impact the project.

By employing sprints in this manner, the team was able to break the project into manageable segments, facilitating focus and progress. The ability to consistently

see progress by completing tasks made for a more satisfying development process.

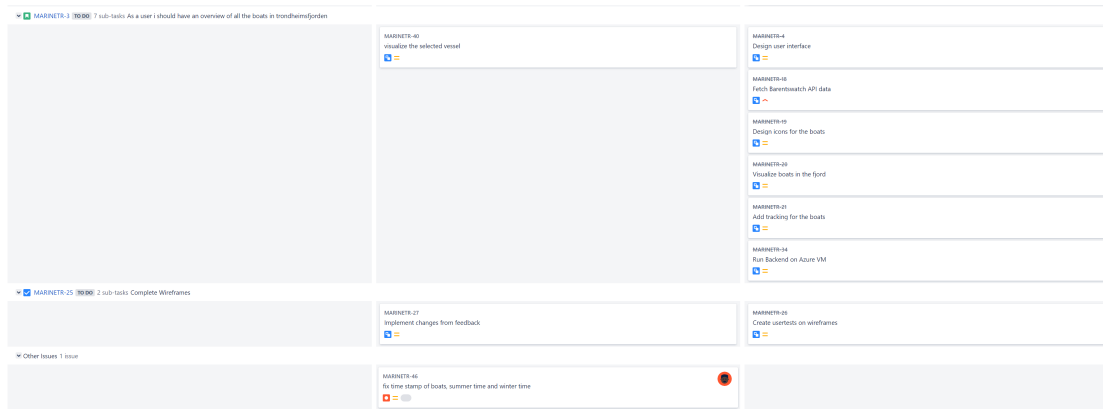


Figure 3.5.1: Issueboard from sprint 10

## 3.6 Development process

### 3.6.1 User Experience

The team initiated the application development by creating a static website, which functioned as a simple image gallery showcasing the wireframes in full screen. This site was presented on screens at Accenture to gather feedback. Concurrently, the team began searching for data sources on boats, determining how to set up the database, and developing the backend.

### 3.6.2 BarentsWatch

Ultimately, the team chose BarentsWatch’s AIS API for real-time vessel data within the fjord. This API provides a data stream containing up-to-date vessel positions, sourced from the reputable Kystverket. Furthermore, the API allows data filtering based on a position grid, enabling the team to focus exclusively on vessels within the fjord. This approach saves time and resources by eliminating the need for manual filtering.

### 3.6.3 Backend server and Database

#### Database Population

Upon choosing the BarentsWatch API as the data provider, the team began developing the Spring Boot server. They decided to prioritize creating functionality for obtaining data from the API before developing other backend features.

To store the data, the backend application is connected to a relational database, using hibernate, which consists of tables linked by primary keys. The data from the BarentsWatch API includes the vessels’ MMSI, a unique ID assigned to each vessel, which serves as an ideal key for storing vessel data.

## 3.7 Testing

### 3.7.1 Endpoint testing

The team created a test set in postman to test to test the endpoints of the RESTAPI. This helped the team quickly see if all the endpoints still worked as intended after changes was made to the backend server. If all tests where to pass the returned values where as expected. If some of the tests where to fail the team could check out that specific test to see what It returned to figure out what went wrong. This was a great help in maintaining quality and hindering bugs. Without it, it could have been hard to figure out if the error happened in the frontend or backend. The Figure [3.7.1](#) shows how one of the test is set up.

The screenshot shows the Postman interface for a GET request to `https://marinetrafficbackend.azurewebsites.net/inArea`. The 'Tests' tab is active, displaying a JavaScript test script. The script performs a second API call and checks if the response is successful and if the 'lastUpdated' field is not older than 1 hour. The test results at the bottom show two tests passing: 'Second API call is successful' and 'lastUpdated is not older than 1 hour'.

```

1  var jsonData = pm.response.json();
2
3  jsonData.forEach(function(item) {
4    pm.sendRequest({
5      url: "https://marinetrafficbackend.azurewebsites.net" + "/get/" + item.mmsi,
6      method: 'GET',
7    }, function(err, response){
8      if (err) {
9        console.log(err);
10     } else {
11       var secondApiResponse = response.json();
12
13       pm.test("Second API call is successful", function () {
14         pm.expect(response.code).to.equal(200);
15       });
16
17       var lastUpdated = new Date(secondApiResponse.lastUpdated);
18
19       var currentTime = new Date();
20       var timeDifference = currentTime - lastUpdated;
21       var oneHourInMilliseconds = 60 * 60 * 1000;
22
23       pm.test("lastUpdated is not older than 1 hour", function () {
24         pm.expect(timeDifference).to.be.at.most(oneHourInMilliseconds);
25       });
26     }
27   });
28 });
29 });

```

Test Results (76/76)

- PASS Second API call is successful
- PASS lastUpdated is not older than 1 hour

**Figure 3.7.1:** Example of postman test. This test checks if inArea call has no vessels older than 1 hour

### 3.7.2 User testing

A survey was created during the later stages of the development process to gather feedback on the significance of various available data for display. The findings were utilized to identify which data should be displayed on the website based on the users' level of interest.

The team also employed additional methods of user testing. They conducted interviews with employees, discussing functionality and providing a live demo of the application. Furthermore, a static demo site showcasing the wireframes was

created and sent to the client for feedback on the design. This allowed the team to gather valuable insights and make necessary adjustments based on the client's preferences.

### 3.7.3 Lighthouse

Google Lighthouse is an open-source tool for automatically measure the quality of a webpage. It gives a score on performance, accessibility, and search engine optimization. This testing tool was utilized to test the frontend webpage to get feedback on these factors. The use of Lighthouse gave the team a quick way to generate a general report or the sites performance and usability. This was a helpful tool alongside user feedback and testing. [42]

## RESULTS

### 4.1 Final result

At the end of the project, the team delivered a fully functional website with all the functionality and attributes requested by the client. The client was very pleased with the result, and the website is to be shown on the info-screens at the Accenture offices. The client cited in the feedback (see section [6.3](#)) that:

"the team had successfully captured the end-users needs and transformed these to functional code, something which is crucial for success". Further more they gave the following statement about the result:

"The final solution meets the requirements outlined in the assignment and does exactly what a marine traffic portal catered to a specific location is supposed to do. Provide real-time updates on the marine ship movements in a predefined grid in an easy-to-understand graphical interface primarily designed for large monitors."

### 4.2 Gathering additional data for the vessels

#### 4.2.1 Problem

Following a meeting with the client, the team was tasked with collecting additional information to present about the vessels. This included details such as nationality, origin, and images, among others. However, during the search for compelling data, the team encountered difficulties in obtaining vessel images and determining their nationality. The Barentswatch API did not include this information, necessitating the team to seek out an alternative source to acquire the data.

#### 4.2.2 Possibilities

##### 4.2.2.1 BarentsWatch API

In the search of an alternative source for vessel data, the team considered purchasing the Marine Traffic API. While this option would have provided the necessary

information, it proved to be quite costly. Unfortunately, due to limited spending, this approach was not feasible for the team to pursue.

#### 4.2.2.2 Web Scraping

A practical solution that emerged from the team's brainstorming efforts was to collect the required vessel data through web scraping the ship-info website. However, this approach required the team to first seek permission from the website owner to use this technique.

Web scraping involves extracting data from websites using automated software, making it a potentially valuable approach for collecting large amounts of information efficiently. However, it's important to note that web scraping can potentially infringe on the website owner's terms of service or even violate copyright laws. Hence, it was essential for the team to ensure that they had explicit permission from the website owner before proceeding with this strategy.

### 4.2.3 Process

**Listing 4.1:** The scraping functions login method

```

1  public void loginToWebsite(){
2      driver.get("https://www.ship-info.com/prog/login.htm");
3      // Insert the username to the field
4      driver.findElement(By.xpath("/html/body/table/tbody/tr
          /td[2]/table/tbody/tr[2]/td/table/tbody/tr/td/form/p/
          input[@id='theFieldID']"))
5          .sendKeys(username);
6      // Insert the password to the field
7      driver.findElement(By.xpath("/html/body/table/tbody/tr
          /td[2]/table/tbody/tr[2]/td/table/tbody/tr/td/form/p/
          input[2]"))
8          .sendKeys(password);
9      // Clicking the submit button
10     driver.findElement(By.xpath("/html/body/table/tbody/tr
          /td[2]/table/tbody/tr[2]/td/table/tbody/tr/td/form/p/
          input[3]"))
11         .click();
12 }

```

**Listing 4.2:** Collecting the image URL

```

1  // check if the vessel has an image.
2  List<WebElement> imageDOMElement =
3      driver.findElements(By.xpath("/html/body/table[1]/tbody/
          tr[2]/td/table[3]/tbody/tr[1]/td/table/tbody/tr/td[2]
          /table/tbody/tr/td[1]/img"));
4  if (imageDOMElement.size() != 0) {
5      // Vessel has an image.
6      // get the image url.
7      imageURL = imageDOMElement.get(0).getAttribute("src");
8  } else {
9      imageURL = "Unknown";

```



9 } \_\_\_\_\_

## 4.2.4 Result

In conclusion, the adoption of a scraping technique proved to be an effective solution for collecting supplementary information. The utilization of this technique offers an automated process that functions for a broad range of vessels worldwide.

**Listing 4.3:** Example of Vessel object after scrapper has gotten data

```
1  {
2  "mmsi": 257234800,
3  "name": "M S NIDARHOLM",
4  "longitude": 10.385262,
5  "latitude": 63.451208,
6  "type": 60,
7  "speed": 0.6,
8  "length": 19.0,
9  "destination": null,
10 "nationality": "Norway",
11 "course": 246.6,
12 "lastUpdated": "2023-05-08T13:30:11.414077",
13 "width": 5.0,
14 "lastVisited": null,
15 "visitedCounter": 0,
16 "image": "https://www.ship-info.com/Vessels/K123612.jpg",
17 "imageAuthor": "Photo: Per Kristian Rognes |
18   Published: 01.10.2008",
19 "portOfRegistry": "Trondheim",
20 "owner": "Valso, Lindis Pauline",
21 "yard": "B.K.V. Staalprodukter, Valsøybotn, Nordmore",
22 "buildYear": "1974",
23 "passengerCapacity": "98",
24 "backTrack": [...],
}
```

## 4.3 Deployment of the backend

### 4.3.1 Problem

During the deployment of the backend to Azure, the team faced an issue with the installation of the chromedriver, which is a prerequisite for the Selenium library. As App Service on Azure does not permit manual installation of packages and programs on the virtual machine, it became impossible to install the necessary component.

### 4.3.2 Possibilities

#### 4.3.2.1 Change scraping library

To resolve the problem of installing chromedriver during deployment to Azure App Service, the team can explore alternative solutions. One option is to use a different library for scraping the data, such as Jsoup etc. This would lead to refactoring the whole scraping code and may cause other similar problems.

#### 4.3.2.2 Dockerize

Alternatively, the team can consider using a containerization solution, such as Docker (Sec. 3.2.6), to package and deploy the application and its dependencies together. This approach would allow for the easy installation of chromedriver as part of the containerized application. Additionally, the team can explore the possibility of using a different library for web automation that does not require chromedriver.

#### 4.3.2.3 Change azure resource type

A third possible solution is to switch the resource in Azure to a standard virtual machine. However, this may raise security concerns since endpoints in an App Service can only be accessed through Azure using keys. If the team opts for a standard virtual machine, these endpoints will be open to all users without any additional security measures implemented in the application.

### 4.3.3 Process

The team researched the different possibilities mentioned above, and tried to figure out what was the best solution for this project. The solution to Dockerize the application seemed to the group to be the best and most flexible solution. This allowed the application to easily be deployed to other cloud hosting platforms, thus not locking the group to azure.

At the next meeting with the supervisor for Accenture. The team presented the problem and their idea for the solution. The supervisor strongly agreed to this fix and said that he would also have done the same.

The next step was to begin the Dockerization. During this phase the team faced significant challenges with getting the image build correctly. The biggest problem was still the chromedriver. The team figured out the problem was with driver installed through a maven dependency.

**Listing 4.4:** Maven dependency causing error

```
1 <dependency>
2     <groupId>io.github.bonigarcia</groupId>
3     <artifactId>webdrivermanager</artifactId>
4     <version>5.3.2</version>
5 </dependency>
```

**Listing 4.5:** Initial use of the webdrivermanager library

```
1 import io.github.bonigarcia.wdm.WebDriverManager;
2 :
3 WebDriverManager.chromedriver().setup();
```

To solve this the team removed the dependency completely and instead installed the chromedriver directly into the system of which the program was running. This

was done by specifying the image base as linux platform with maven and JDK 17 installed. Then linux terminal commands were run to install both Google Chrome and the Chrome driver onto the system. Thus, the application was able to directly access the systems chromedriver by locating the executable file as shown below.

**Listing 4.6:** Revised implementation of the chromedriver

```

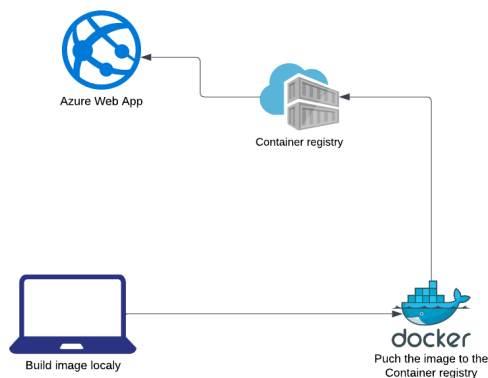
1 File file = new File("/usr/bin/chromedriver");
2 System.setProperty("webdriver.chrome.driver",
   file.getAbsolutePath());

```

Despite having implemented the Docker solution for our project, the team encountered a few minor problems. One of them was an incompatibility between the Chrome driver and the Chrome browser. Additionally, we faced an issue where the hard-coded version to be downloaded might have been deprecated or unavailable for download. To resolve this problem, we made the Docker image capable of automatically checking and downloading the latest versions available. This approach effectively resolved both issues.

#### 4.3.4 Result

As a result of the dockerization of the application. The team managed to deploy the application to an Azure Web App using a Container registry. In the deployment settings of the Web App it was specified to always use image `marine_traffic/backend` with the latest tag. Thus, when we pushed a new image to the Container registry, we always tagged it with “latest”. When the image was successfully pushed, the Web App automatically fetched the images and deployed the new version. Figure [4.3.1](#) underneath shows the process. All issues related to the chromedriver and deployment were successfully resolved by implementing this solution.



**Figure 4.3.1:** Process of deploying a new version of the backend

## 4.4 Designing the website

### 4.4.1 Problem

When design the website the team wanted to make sure the website was usable by everyone at the Accenture's office regardless of disabilities such as color blindness, weak sight and sensitive eyes. At the same time the site needed to be easy to use and understand. To do this the team should follow the design guidelines for designing a website. See section [2.10.1](#)

### 4.4.2 Possibilities

To make the site usable by people with color blindness. The site must make use colors that does not clash for different color blinded users and make use of animations or other forms of markings to display changes or critical information. Such that the site can be used without any colors at all.

For making an easy-to-use site the team should make the site easy to understand. This can be done by using pure text to display information. And when using icons, the icons should be easy to understand. The site also needs to take users with weak or no eyesight into account making the site optimized for screen readers. Another factor to consider is if the user does not have access to a mouse and just uses a keyboard. Then the site needs to be usable with just a keyboard.

The site needs to be optimized for TV screens as this is the primary usage area. Thus, the site needs to have big font so that the text is readable far away. This also applies to all factors of the site such as being able to clearly see which vessel is selected.

### 4.4.3 Process

The team made sure to follow all the design guidelines to their best ability given the time limit on the task. Firstly the team started by making the wireframes in Figma to decide on the layout and the colors of the page. Before deciding on design a user test was done on the wireframes. See section [4.10.1](#) for more information on how the test was done.

After agreeing on the wireframes, the team had a meeting with two of Accenture's design team to get further input on the design and how the team could improve the wireframes more. More about this can be found in section [4.10](#).

Following the design guidelines, the colors of the site was an important aspect for the team, as they wanted all users regardless of colorblindness to be able to use the site to. Thus, the team made sure that there was as little color clash as possible on the site. However, there are a lot of different vessel types. And the team could not use different colors on all of them and still regard optimization for color blindness. To work around this animation and enlargement was added to the selected vessel. Such that a user would be able to clearly see which vessel is selected regardless of colorblindness. The vessel type is also displayed within the

information box of the vessel. Making one able to find out about the type without needing to see the color. A dark color theme was also implemented to the site. This was implemented in case of viewing in a darker room. The dark mode would be more pleasant to look at and not as straining on the eyes in darker environment.

To optimize the site for multiple user inputs, multiple measures were set in place. Firstly, the team implemented alt text to images and buttons on the site. This was to ensure functionality with users using screen readers. Secondly, tab indexation was implemented onto the TV mode toggle, dark mode toggle and settings icon. This allows the user to use the tab-key to navigate between the elements. Then to activate the selected element the user can hit either return/enter or the space-key. This implementation was done as some users might not have or use a mouse. Thus, ensuring the capability of entering TV-mode and toggling settings when opening the site on the TV screens.

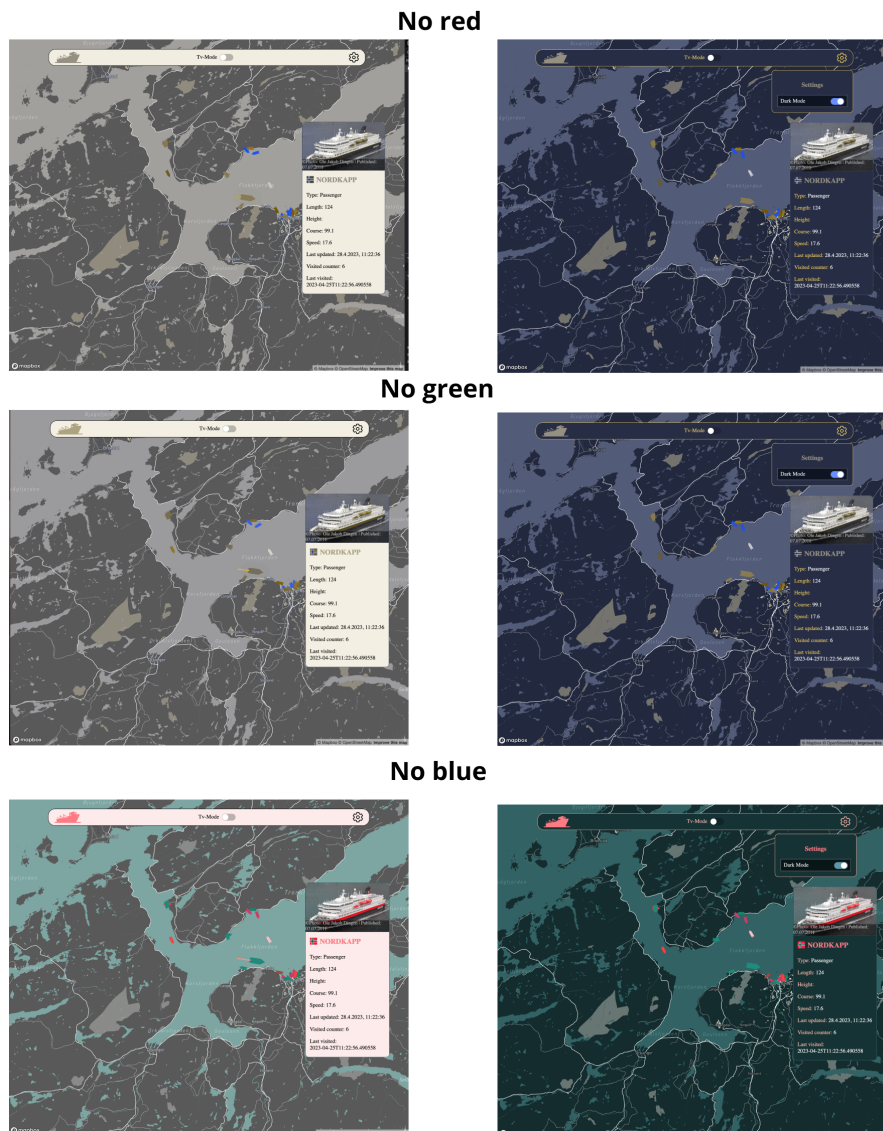
#### 4.4.4 Result

After taking these different forms of accessibility and usability into account. The team was able to create a usable site for the client. The team got great feedback from the client on the implementation of dark mode to the site. Saying it was good looking and appreciated. The team simulated how the site would look with different versions of color blindness. The figure [4.4.1](#) under shows how the results of this. As the results show the maps color difference between the ocean and land works well regardless of color blindness. The vessel icons, however, look similar. Regardless one is still able to see a difference between them and separate them from the ocean.

The icons used on the website are recognizable icons. The settings icon is the icon of a cog this is the generic and typical icon used for settings. This makes the icon easily recognizable by the user and the user will be able to understand its usage and functionality. Furthermore, the icon used for the vessels is in the shape of a boat looked down upon. Thus, making it easy to understand what it is. Lastly the icon used for the toggle switches is also a commonly used way of displaying a toggle switch. Again, making it easy for the user to understand its functionality. The switch also changes color and has an animation when clicked to give the user feedback on that something happened.

The team also ran a [Lighthouse](#) test on the site. This is an accessibility tester that scores the site on accessibility, performance, best practices, and search engine optimization. The result gathered from this was excellent and to the groups satisfaction, scoring almost 100/100 on accessibility, best practices, and search engine optimization. The results can be found in [attachment G](#).

### Colour blindness



**Figure 4.4.1:** Displays how the site looks with different types of color blindness. Light mode on the left and darkmode on the right

## 4.5 Handling the visitor counter for the vessels

### 4.5.1 Problem

The client requested a counter to be displayed indicating the number of times a vessel has visited the fjord, along with its last visited date and time. However, the team faced a dilemma on how to approach this task in order to accurately calculate the vessel's entrance and exit from the fjord.

## 4.5.2 Possibilities

### 4.5.2.1 Time check

Initially, the team proposed assuming that a vessel was outside the fjord if it hadn't received an update in the past hour. However, this approach proved impractical as some vessels would turn off their AIS while docked, leading to inaccurate data. Consequently, the team decided to repurpose this solution for hiding inactive vessels on the /inArea endpoint, which displays all vessels on the map.

### 4.5.2.2 Point in polygon algorithm

The team's second solution involved using two hexagons to determine the vessel's position: one to define the "Active area" and a slightly larger one to outline the region accurately. With the help of the Point in Polygon algorithm (Sec. [2.9.1](#)), the team could check if the vessel was moving from the active area into the "deadzone" or vice versa.

## 4.5.3 Process

A function called `checkIfInArea()` was added to the vessel class by the team. This function accepts a location point and a geoJSON area as arguments and evaluates if the point is contained within the area. The function returns a boolean value where `True` indicates the point lies inside the polygon. The following code snippet shows an example implementation.

**Listing 4.7:** Code to check if coordinates are inside a GeoJSON area

```

1  boolean inside = false;
2  for (int i = 0, j = vertices.size() - 1; i <
   vertices.size(); j = i++) {
3      VesselCoordinatePoint v1 = vertices.get(i);
4      VesselCoordinatePoint v2 = vertices.get(j);
5      if ((v1.getLatitude() > currentPoint.getLatitude())
        != (v2.getLatitude() >
6          currentPoint.getLatitude()) &&
           (currentPoint.getLongitude() <
7             (v2.getLongitude() - v1.getLongitude())
            * (currentPoint.getLatitude() -
8             v1.getLatitude()) /
           (v2.getLatitude() - v1.getLatitude()) +
9             v1.getLongitude())) {
10         inside = !inside;
    }
    }
    }
    return inside;

```

When determining whether a vessel is departing from the active area, the team utilizes this function. Their criteria for a vessel leaving the area is when the current position is located outside the active area while the previous position remains inside. If both conditions are met, the visitor counter will increment, and the last visited date will be updated.



#### 4.5.4 Result

The implemented solution allows the application to determine with a high degree of accuracy whether a vessel has visited the fjord. However, in some infrequent cases where the AIS data is slightly inaccurate, the counter may update multiple times for a single visit. To address this issue, a cooldown timer could be introduced to determine when the counter should be updated. This feature could be incorporated into future work.

### 4.6 Server application

#### 4.6.1 Problem

The team needed a server for storing data from different vessels, in addition, the team wanted to have calls to the BarentsWatch API in the backend to reduce the number of calls the frontend would make. Therefore, the team needed to choose a server application that could both store data easily and serve as a REST API.

#### 4.6.2 Possibilities

##### 4.6.2.1 Firebase

The teams' initial thought was to use Firebase as it is very easy to set up and implement. However, as we wrapped our heads around the problem the team figured the solution had to be a relational database. Implementing this in firebase would be more complicated as a result. In addition, the amount of calls/writes made to the database needed the pricing would extend above our budget.

##### 4.6.2.2 Spring Boot

The team has previous experience with the Java Spring Boot framework, so naturally it was the next option. Spring boot creates a Hibernate relational database, and it is also convenient for setting up a REST API, as it is built in the framework. There is no limit to calls/writes, which is a huge advantage, and we also have experience with deployment of a spring boot application.

#### 4.6.3 Process

##### 4.6.3.1 Models

Within the Spring Boot application, controllers, services, and repositories utilize models to present data and entities in a way that is easy to manipulate. These models also reflect the structure of the corresponding tables they are based on.

##### 4.6.3.2 Repositories

In the Spring Boot application, the repositories function as the data access layer and serve as interfaces for interacting with the persistence layer. They are responsible for creating and managing queries between the application and database. By

utilizing repositories, the service layer can easily retrieve data from the database as shown in code example 4.8.

**Listing 4.8:** Defining the query call for receiving all active vessels in the Area of interest

```

1 @Query(value = "SELECT vessel.mmsi, vessel.latitude,
    vessel.longitude, vessel.type, vessel.course from vessel
    where vessel.last_updated > NOW() - interval '5
    MINUTE'", nativeQuery = true)
2 List<Object> getActiveVessels();

```

#### 4.6.3.3 Services

The service layer handles the business logic. When the service is called from the controller, it handles the boundaries and then executes the call. The service often uses, alters, or updates data from the repository layer as shown below in 4.9.

**Listing 4.9:** Update a vessel when new information is retrieved from the API.

```

1 public boolean updateVessel(Vessel vesselToUpdate, Long
    mmsi, String name, double longitude, double latitude,
    int type, double speed, double length, String
    destination, String nationality, Double course, double
    width, LocalDateTime lastVisited, int visitedCounter,
    String image, String imageAuthor ) throws
    MalformedURLException {
2
3     vesselToUpdate.setMmsi(mmsi);
4     :
5     vesselToUpdate.setImageAuthor(imageAuthor);
6
7     try {
8         vesselRepository.save(vesselToUpdate);
9         return true;
10    } catch (Exception e){
11        return false;
12    }
13 }

```

#### 4.6.3.4 Controllers

The controller handles the endpoints of the server. The responsibility of the controller is to handle the requested data sent to the server, and then send back a response. This includes formatting the JSON from the request body, as shown in code example 4.10, or converting request parameters into arguments that can be used by the service layer. The controllers also call the appropriate methods in the service layer to generate the desired response to the request.

**Listing 4.10:** Gets a list of all .

```

1 @CrossOrigin
2   @GetMapping("/inArea")
3   public ResponseEntity<List<Object>>
4     getVesselsWithinActiveArea() throws
5     ExecutionException, InterruptedException {
6     List<Vessel> vessels =
7       vesselService.getVesselsWithinActiveArea();
8     JSONArray vesselInfo = new JSONArray();
9     for (Vessel vessel : vessels) {
10      JSONObject vesselJson = new JSONObject();
11      vesselJson.put("mmsi", vessel.getMmsi());
12      :
13      vesselJson.put("type", vessel.getType());
14      vesselInfo.put(vesselJson.toMap());
15    }
16    return ResponseEntity.ok(vesselInfo.toList());
17  }

```

#### 4.6.4 Result

The implemented solution is working well, as it is storing all the data received from the BarentsWatch API in the database, and the REST API is working well for sending data from the backend to the frontend. The code is separated into layers and in their own packages to make it more organized, more cohesive, and easier to maneuver – hence the Spring boots best practices [\[43\]](#) have been followed. [\[6.3\]](#)

## 4.7 Web application

### 4.7.1 Problem

The initial challenges faced by the team included determining the optimal organization of the codebase for ease of explanation and maintenance, as well as deciding on the most suitable framework, if any, to use. The next issue they had to address was figuring out how the website would display and retrieve information from the server.

### 4.7.2 Possibilities

#### 4.7.2.1 Organizing the codebase

A viable strategy for organizing code and minimizing duplication is to employ a framework that enables the creation of components. By developing a component once and using it in multiple locations within the code, maintenance becomes more manageable, as any changes made to the component will automatically apply to all its instances.

### 4.7.2.2 Deciding on framework

The team explored various framework options for the project. They first investigated Next.js, then considered Svelte, and finally examined the possibility of using standard React or Vue.

### 4.7.2.3 Displaying information

The first essential piece of information to display is the background map, as it provides context for the rest of the data. The team identified two main approaches for displaying the map: using a map API to present an interactive map in the background or using an image of a map as the background.

To display the vessels, the team discovered only one viable method – using an icon on the map to represent the position of each vessel. These icons need to be clickable and then the site is to show specific information about that vessel.

## 4.7.3 Process

The team initially focused on selecting a framework, as they wanted the ability to create components. They considered both React and Vue but ultimately chose React, as most team members had experience with it. After agreeing on React, they decided between using native React or another React-based framework. They selected Next.js for its appealing features and the opportunity to try something new. Next.js also allowed for both client-side and server-side rendering.

Once Next.js was chosen, the team explored map solutions, eventually settling on MapBox for its features (see section [3.2.3](#) for details). They quickly discarded the idea of using a static background image, as it would make positioning vessels difficult. Using the MapBox library, they were able to directly insert coordinates from the server as map markers. This approach enabled them to create a function that fetched all vessels from the backend, created markers with unique MMSI identifiers, assigned colors based on vessel type, and set rotation based on course.

After implementing the map with all vessels, the team focused on displaying information about specific vessels and their selection. They first needed to obtain the MMSI number of a particular vessel, achieved by assigning each marker a click function that returned the marker's ID. In the application's root, an MMSI state was created, which was then set to the returned marker ID.

Next, the team created the info card component, which contained all information fields and used conditional rendering to display only the fields relevant to the current vessel. A function to update the information on the card was then developed and placed inside a React `useEffect`. This function would run every time the root MMSI changed, effectively updating the info card for the selected vessel.

### 4.7.4 Result

The outcome of this process was a fully functional web application that the team deployed and made accessible to everyone. Refer to section [4.11](#) for details on how this was achieved. The code structure and organization made it easy for the team to modify and expand the code when necessary. The technologies chosen by the team worked seamlessly, and they felt confident in their choices. As of today (May 8, 2023), the application is ready for further expansion if the addition of new features is desired. Accenture provided positive feedback and expressed satisfaction with the results.

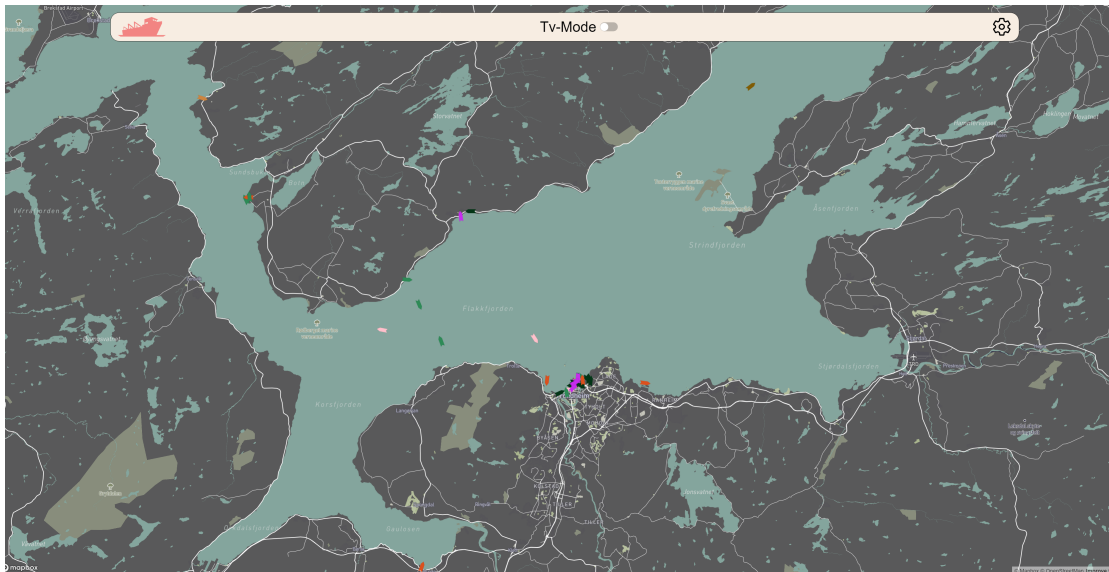


Figure 4.7.1: What the finished web application looks like

## 4.8 Backtracking the vessels

### 4.8.1 Problem

The client expressed a desire to view the past locations of specific vessels. As a result, the team began exploring the development of a backtracking feature. Upon investigation, they identified two primary challenges that needed to be addressed. First, they had to find a way to store the position data for all vessels. Second, they needed to effectively and logically display the stored data on the website, ensuring it was easy to understand and navigate.

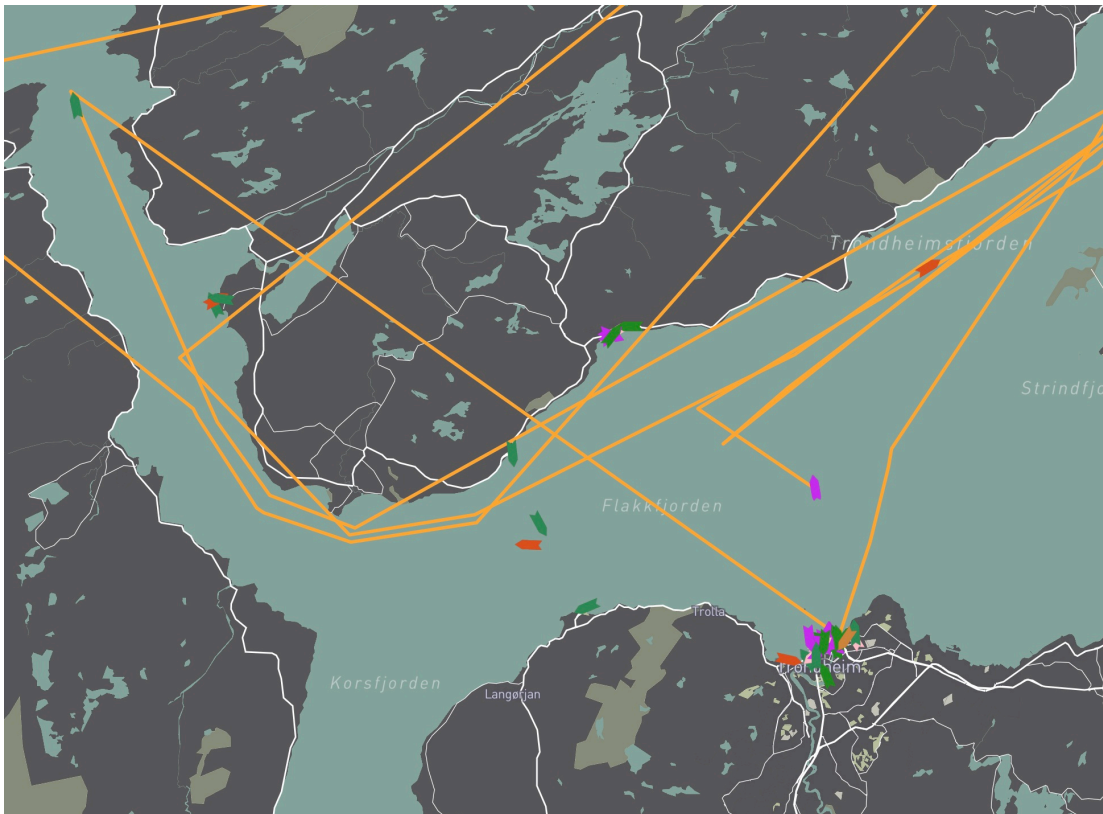
### 4.8.2 Process

The team initiated work on collecting data and storing it in the database. They began by implementing a function to save each vessel's current position in a list. This meant that when a vessel was updated, a new position would be stored in the list. The list was ordered with the oldest points first and the newest points last, and was integrated into the vessel class, ensuring that all vessels had a list of their previous positions stored. Initially, this solution seemed to work, so the

team started developing a way to visualize the points on the frontend.

To display the backtracking points, the team fetched information about the selected vessel, sorted it, and extracted the points. They utilized several MapBox functions to display the points on the map. First, a new canvas source was created, specifying that the source was of type GeoJSON and that the points should be connected with a line. Next, a new layer was added to the map using this source as a template, defining the line's styling. Lastly, the coordinate list was inserted, creating a pathed line between all the points.

Initially, this approach appeared to work, but the team soon discovered an issue with the order of the points. To address this, they limited the number of points a vessel could store, removing the first element from the list whenever a point was added at the end. While this improved the backtracking visualization for some time, they later observed incorrect backtracking for certain vessels. See figure [4.8.1](#).



**Figure 4.8.1:** Shows how the backtracking was on some vessels when the points was stored in a list of max 50 points

Upon investigation, the team discovered that the issue was related to the removal of points in the backend. They realized that their method of adding and removing objects from the list was not only inefficient but also resource-intensive. This was because each time a vessel received a new point, the program had to remove a point and shift all the points to their new positions. With approximately 40 vessels in the fjord this meant around 300 points received every minute, this shifting

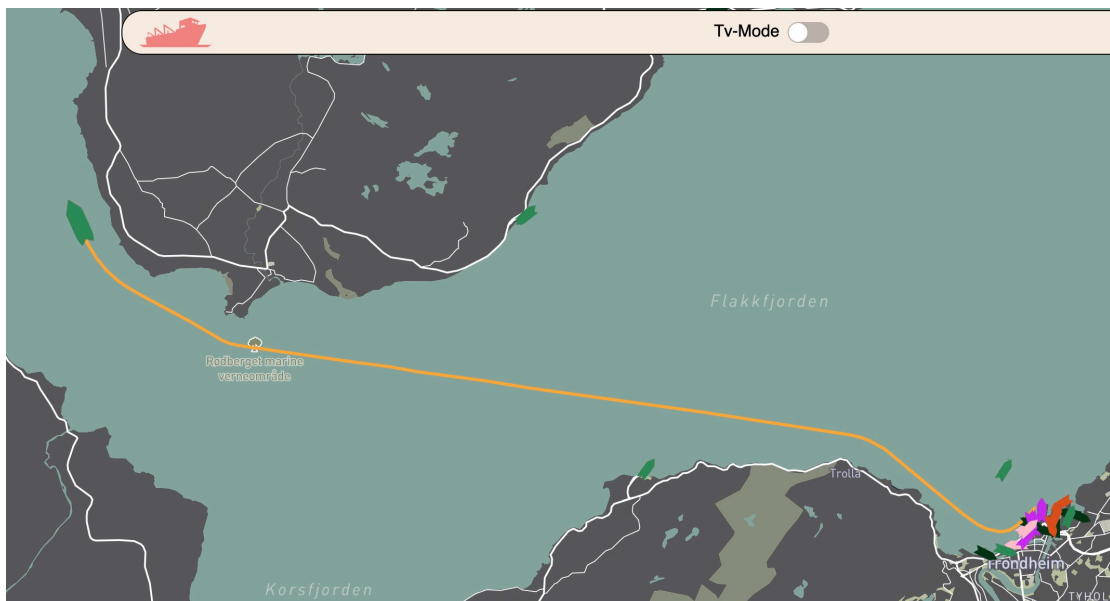
process was executed far too many times.

The team decided to revert to their original method of adding points to the list and implemented a deletion function that deleted all points older than one hour using a database query. They scheduled this function to run every 10 minutes. By doing so, the team shifted the load of iterating and shifting the list away from the backend, and by using query operations on the database, the process became both faster and more resource-efficient.

These changes resolved the backtracking issues, significantly improving its appearance and accuracy. See figure [4.8.2](#). Moreover, this modification enhanced the backend's performance, making the website feel more responsive and efficient overall.

### 4.8.3 Result

After addressing the challenges associated with creating the backtrack, the team achieved the desired results. The backtrack lines accurately reflect the position of the vessels, smoothly following their turns. No points are misplaced, and every vessel now has a backtrack displayed correctly. The figure [4.8.2](#) under shows how the backtrack ended up.



**Figure 4.8.2:** Shows how the backtracking is when the point removal was done via a database query

## 4.9 TV Mode

### 4.9.1 Problem

Upon receiving the assignment, the team swiftly recognized the need for a "presentation mode" on the website that could effectively exhibit the vessels on TV screens without depending on user input. To achieve this, the website would have

to be capable of automatically cycling through the vessels and displaying their relevant data on the screen.

## 4.9.2 Process

The implementation process for the TV mode involved both backend and frontend development to ensure a seamless and engaging experience for TV screen display.

### 4.9.2.1 Backend

The team aimed to develop an algorithm that generates a list of the most interesting and desired vessels to be displayed on the TV mode. In collaboration with the client, the team created a geojson area that served as a priority zone for vessels to be displayed on the TV screen. Subsequently, the backend would collect the IDs of all vessels located within this zone and return them, one by one, through the REST endpoint. This endpoint also employs the code in Listing [4.10](#) for the purpose of code reuse. Once a vessel was shown, it was removed from the list. If the list became empty, the system would check for vessels in the priority zone again. If no vessels were found in the priority zone, the system would randomly select a vessel from the entire fjord area. The algorithm performed adequately as an initial solution, but there is room for improvement in the future. Fine-tuning can be achieved by incorporating metrics such as a visited counter and other relevant factors to enhance the algorithm's effectiveness. By implementing these enhancements, the algorithm can be optimized for even better results.

### 4.9.2.2 Frontend

The frontend component for the TV mode heavily relies on the functionality provided by the interactive modes' React states and hooks. It is designed to dynamically change the selected vessel at regular time intervals. As the interval triggers, the browser sends a request to fetch an MMSI for display. Once the MMSI is received, the corresponding React state, called MMSI, is updated automatically, resulting in the display of the information card and the selected vessel. To ensure a smooth transition between vessel changes, a brief zoom and move animation is employed when the TV mode is active. This animation enhances the seamless experience of switching between vessels.

## 4.9.3 Result

The implementation of the TV mode was successful. The backend algorithm generated a curated list of vessels within the priority zone, ensuring captivating display on the TV screen. The frontend seamlessly updated the selected vessel at regular intervals, providing a smooth transition between changes. Overall, the TV mode achieved its objectives, delivering an immersive and enjoyable viewing experience.



## 4.10 User experience

Throughout the project, the team presented all decisions made to the client for their review and approval. Since the client would be an end-user of the application, they were used actively in the development process for testing and review. This approach ensured that all design and functionality aspects were in line with the client's expectations and provided a proper user experience. In addition, the team conducted three targeted test cases to gather feedback directly from end-users. This feedback was then incorporated into the development process to enhance the overall user experience.

### 4.10.1 Wireframes on screens

Before deciding on a design for the website, the team made a static website consisting of the different wireframes created. This website was a simple image gallery displaying each wireframe in full screen mode. The purpose of this site was for Accenture to open it on their TVs to view how the site would look on the intended screens. Then they were to send feedback on which design, they liked the most. However, the team never got any feedback from this, thus they with agreement from the supervisor decided on the design they thought was the best. The static website codebase can be accessed from the following github repository: <https://github.com/GustavsenSj/MarineTrafficStatic>

### 4.10.2 Survey

During the project, the team conducted a survey for the end-users to gather their feedback on the most important data points. The primary objective of the survey was to obtain insights into the users' interests. The survey was distributed to end-users, and seven responses were received. The survey results are provided in [appendix F](#). Based on the survey results, the team identified that the users were highly interested in images, country of origin, and vessel type. Additionally, they were interested in the vessel's backtrack and passenger capacity. The team took these suggestions into consideration and implemented them into the website.

### 4.10.3 Interviews

In the later stages of the development, the team asked the client to provide end-users to interview for user testing. The purpose of the interview was for the team to identify any flaws when the end users interacted with the website. The meetings were held online. The format of the interview was that the test subjects shared their screen, while the

## 4.11 Making a public site

### 4.11.1 Problem

A part of the assignment from the client was to make a public website, which was in a public cloud environment that hosts our website. Therefore, the team needed

to find a way to host the website and create a cloud environment.









### 4.11.2 Possibilities

Since Accenture uses Microsoft Azure in their own applications, the client wanted us to use Azure in our development, to make it easier for them to take over the website. Microsoft Azure is a cloud computing platform operated by Microsoft that provides global access to data centers for the management, development, and utilization of applications and services.

### 4.11.3 Process

The team received an Azure subscription from NTNU and initially experimented with it, as none of the members had prior experience using the platform. After some trial and error, the team successfully created a Resource Group to house all necessary resources and consolidate various components in one location. An Azure PostgreSQL database was established to host the database, an identity key, an App configuration, and a Container registry to hold the backend Docker image. Subsequently, two Application services were created for the backend and frontend.

When a new image is pushed to the Container registry, it automatically deploys the updates to the App service. Continuous deployment for the frontend was set up through GitHub Actions, so updates pushed to the "main" branch are deployed to the website. The creation of an App service also generates a domain, eliminating the need for the team to create a new one.

Name	Type
 MarineTraffic	Resource group
 MarineTrafficBackend	App Service
 marine-traffic-ntnu	Azure Database for PostgreSQL single server
 MarineTrafficConf	App Configuration
 MarineTrafficFrontend	App Service
 ASP-MarineTraffic-bf22	App Service plan
 MarineTrafficContainer	Container registry
 MarineTrafficKey	Managed Identity

**Figure 4.11.1:** Resources in Azure

### 4.11.4 Result

The application has been transformed into a public website, which is now hosted in the Azure cloud environment. This setup allows for a seamless transition for Accenture to take control of the website once the assignment is completed. The backend domain can be accessed through the endpoint <https://marinetrafficbackend.azurewebsites.net> to retrieve data from the backend. Additionally, the frontend of the website can be viewed at this link: <https://marinetrafficfrontend.azurewebsites.net/>.

DISCUSSION

## **5.1 Communication**

### **5.1.1 With Client**

To gather feedback from end-users, our team conducted a survey among co-workers at Accenture to determine what kind of information they would like to see displayed about the vessels. Additionally, we conducted interviews with two employees from Accenture later in the development process, which yielded valuable insights on the design, strengths, weaknesses, and potential improvements. During these interviews, the employees also provided us with helpful suggestions on some minor details that could enhance the user experience.

The team has found that maintaining regular communication with the client throughout the project is incredibly beneficial. By holding frequent meetings, any minor issues or questions that arise can be addressed promptly, which has prevented the need for any major rework or delays. As a result, the project has progressed smoothly and efficiently, without any significant setbacks.

### **5.1.2 Within the group**

To facilitate an agile development process, the team implemented daily stand-up meetings lasting 15 minutes at 10 AM. These meetings served as a useful tool for structuring the workday and keeping team members informed about each other's progress. The physical proximity of team members working on campus was also essential in maintaining a sense of structure and morale. By working in close proximity, team members could easily collaborate and seek help from one another, simplifying the process of problem-solving and promoting a more cohesive working environment.

## 5.2 Development process

### 5.2.1 The plan

The team's diligent planning efforts were evident in the ease of following the plan. However, the team's optimistic scheduling approach, which involved adding pressure to the team, and not accounting for potential challenges or delays, led to deviations from the original milestone plan. Some user tests had to be rescheduled due to unforeseen issues on the client's side, resulting in delays. Furthermore, the completed website was published one week later than planned, as the initial target was to complete development by the end of April. Despite these setbacks, the team ultimately delivered a high-quality end product.

### 5.2.2 Agile methodology

Since the team had prior experience working in the agile methodology, and it is the industry standard for development projects, it was a natural choice for the team to adopt agile practices. One of the key benefits of working in an agile environment is the ability to prioritize tasks based on their importance, rather than following a set plan. This approach allowed the team to remain adaptable to changing circumstances throughout the project. Additionally, the agile framework facilitated continuous client involvement, enabling the client to stay updated of the progress and provide feedback along the way.

## 5.3 Collaboration with Jira and Confluence

The team's exceptional collaboration was a highlight of the project. Having worked together on previous projects, the team members were familiar with one another's working habits, which facilitated seamless teamwork. Most workdays were spent together in the same room, with all team members present on most days. The team thinks Jira is a more useful tool for larger teams where direct communication is more difficult. Since the team found it easier to ask each other directly when working in close proximity, rather than logging bugs and assigning them to specific team members. Despite initial challenges in adapting to Jira, the team gradually used it more actively and found it helpful in keeping track of bugs and tasks.

Given the client's relatively vague product specifications, creating a comprehensive product specification in Confluence from the outset was difficult. Instead, the team added details gradually as progress was made and new possibilities and desired features was discovered. Confluence was also utilized to store meeting notes and retrospectives from sprints, providing a useful reference for the team to ensure that all feedback from the client had been incorporated into the project. Additionally, the retrospectives helped the team to reflect and improve upon their teamwork, a critical aspect of successful development.

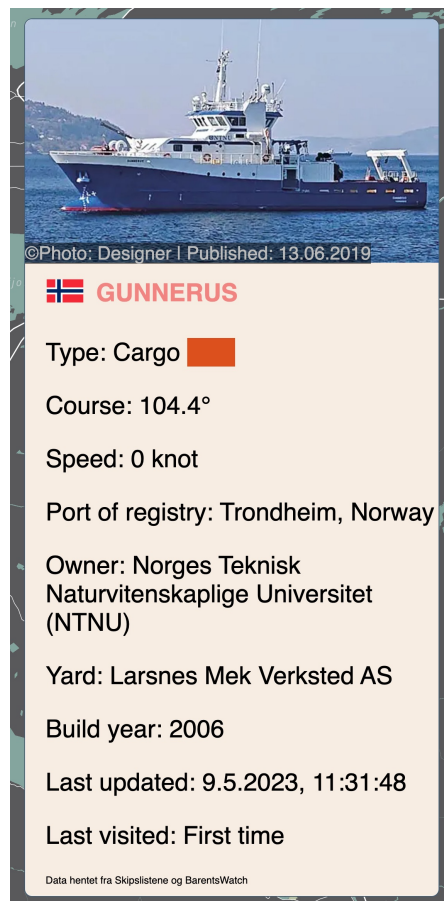
## 5.4 Ethics of Web scraping

To get the images and the information we needed the team needed to scrape the ship-info webpage, as they didn't have an API. According to [44], there are a few steps needed to be followed for the scraping process to stay transparent and ethical:

- Whenever possible, use a Public API instead of scraping data. If the API provides the data you need, it's best to avoid scraping altogether.
- Use a user agent string to identify yourself when sending requests to the website. This helps the website owner know who is accessing their data.
- Limit the rate of your data scraping and control the number of requests per second. Avoid overwhelming the website with too many requests that could be perceived as a DDoS attack.
- Only collect and save the data that is necessary for your business purposes. Avoid saving any unnecessary data.
- Respect the website's robots.txt and analytics requirements to avoid scraping private data from sensitive areas.
- Establish a formal Data Collection Policy that outlines the rules and procedures for data scraping within your organization.

The web scraping process used to collect ship-info data does not store any personal or private information. Moreover, the process is triggered only once for each new vessel that enters the area of interest. Since the website did not have a robots.txt page, the team proceeded with the scraping process. Furthermore, the website owner has provided login information and granted permission to use web scraping to gather data, on the condition that the team gives proper credit to both the website and the photographer for the photos, [5.4.1]

Based on these factors, the team believes that they have adhered to all ethical considerations while scraping the ship-info webpage. However, it would have been preferable if the website had an API to facilitate data collection. The website owner expressed satisfaction that someone was interested in their data and viewed it as a free advertisement.



**Figure 5.4.1:** The infocard showing the information scraped and the credits

## 5.5 Social impact

By keeping the application open on the office info screens, it becomes an interactive component within the workspace. This helps foster a better working environment by giving employees a shared point of interest, sparking conversations and discussions in the office. Such an environment can make the office feel more welcoming and engaging for both new and existing employees.

## 5.6 Correct data

The team prioritized the collection and display of accurate data, as it was crucial not to provide users with false information. Therefore, only trustworthy sources were used for data gathering. BarentsWatch, a company with a decade of experience in gathering and distributing data about Norwegian coastal and marine areas [45], was chosen for this purpose. Additionally, Skipslistene.no, which has been publishing and collecting vessel information since 1982 [46], was used to gather further information.

In order to maintain the accuracy of the data, the team ensured that only up-to-date information is displayed on the map. Vessels that have not been updated

recently are removed from the map, ensuring that no outdated or incorrect information is shown. This commitment to data accuracy and timeliness adds to the reliability and usefulness of the application.

## 5.7 General Data Protection Regulation

While developing the application, the team took the General Data Protection Regulation (GDPR) into account. The application is only using vessels through the AIS stream. AIS messages are considered in the public domain and freely available<sup>[47]</sup>, and therefore does not violate any GDP regulations.

Furthermore, the site does not store any data about the users accessing it. The team saw no need to collect user data, as the site functions completely without needing this information. This approach ensures the application is fully compliant with GDPR and respects the privacy of all its users.

## 5.8 Environmental impact

The website's provision of real-time vessel information in the fjord contributes to environmental awareness within Accenture. Employees can actively monitor and understand the impact of maritime activities on the local ecosystem, promoting a sense of environmental responsibility and stewardship.

With access to real-time data, employees can monitor vessel movements, traffic patterns, and interactions with the environment. This monitoring enables them to assess the potential ecological impact, such as pollution, disturbance to marine life, or habitat degradation. Armed with this knowledge, employees can make informed decisions and take proactive measures to mitigate environmental risks.

## 5.9 Economical impact

The inclusion of real-time vessel data on the information screens significantly enhances employee productivity within Accenture. By providing employees with readily available information about the vessels in real-time, the website eliminates the need for individual research efforts, thereby saving valuable time and resources. Employees no longer have to spend time manually gathering information about vessels that pique their interest.

The time saved through this streamlined access to data directly translates into increased productivity. Employees can now allocate their time and efforts more efficiently towards their core responsibilities, without the need for extensive vessel research. This improved productivity can have a positive impact on the overall operational efficiency of Accenture's workforce.





## CONCLUSIONS

### 6.1 Conclusion

The primary goal of this bachelor project was to develop a comprehensive, publicly accessible web application that displays vessels in Trondheimsfjorden. The project involved creating a backend with a REST API for gathering and delivering data, a database for storing the data, and a frontend for visualizing the data to users. This report has provided insights into the decision-making process and the team's collaboration and work throughout the project.

The team successfully created a backend REST API using Spring Boot in Java, which gathers live ship data from BarentsWatch and additional information from ShipsListene through web scraping. The data is organized in a PostgreSQL database, and the REST API is deployed to an Azure web app using a Docker container image, allowing for quick deployment and updates. The frontend was developed using React through the Next.js framework, with the MapBox library for a customizable interactive map, fetching vessel data from the REST API server.

The learning curve for this project was steep, as most team members had no prior experience with React or web scraping. However, the learning outcomes were excellent, with all members gaining a good understanding of these technologies and learning about developing products in unfamiliar domains, such as marine traffic and related concepts.

A significant takeaway from this project is the importance of user testing. Meetings with the client and user tests conducted by the client were instrumental in shaping the project's development. Without these tests and interviews, the team would not have known which information to display or how users would interact with the site.

### 6.2 Final Product

The final product from the project successfully met all the criteria provided by the client. Although the project description was open-ended and did not have

many detailed specifications, the team collaborated effectively with the client to tailor the project to their desires and requirements. As a result, a fully functional website was created, complete with a database and back-end. Overall, both the team members and the client were pleased with the outcomes.

### 6.3 Further work

As mentioned earlier, both the front-end and back-end of the project were developed and structured with an emphasis on high-quality and maintainable code. This meant using descriptive variable and function names and adhering to the DRY principles.

The team had several features and improvements in mind that they would have liked to implement if more time were available. First, they wanted to improve the TV-mode selection algorithm by creating a prioritized queue of vessels, with those most likely to be of interest being moved to the front. For example, if a new vessel that had not been in the fjord before were to enter, it would be prioritized and displayed more frequently than vessels docked in the harbor for an extended period.

Another feature the team considered implementing was map filtering, allowing users to display only vessels of a specific type. This feature was not prioritized, as there were not many vessels in the fjord and the map did not appear overcrowded. However, if more vessels were to enter the fjord in the future, this feature could be a valuable addition.

The website's loose coupling between the frontend and backend enables easy adaptation for various fjords. By simply modifying the geojson area in the backend, the website can be customized to accommodate different locations. This flexibility makes the solution highly versatile and applicable to diverse requirements.

## REFERENCES

- [1] *Maritime Mobile Service Identity*. en. Page Version ID: 1148641562. Apr. 2023. URL: [https://en.wikipedia.org/w/index.php?title=Maritime\\_Mobile\\_Service\\_Identity&oldid=1148641562](https://en.wikipedia.org/w/index.php?title=Maritime_Mobile_Service_Identity&oldid=1148641562) (visited on 04/26/2023).
- [2] *What is Object-Oriented Programming (OOP)?* en. URL: <https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP> (visited on 04/20/2023).
- [3] *Software Engineering | Coupling and Cohesion*. en-us. July 2018. URL: <https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/> (visited on 04/20/2023).
- [4] *Design Patterns and Refactoring*. en. URL: [https://sourcemaking.com/design\\_patterns](https://sourcemaking.com/design_patterns) (visited on 04/20/2023).
- [5] *Observer pattern*. en. Page Version ID: 1149928597. Apr. 2023. URL: [https://en.wikipedia.org/w/index.php?title=Observer\\_pattern&oldid=1149928597](https://en.wikipedia.org/w/index.php?title=Observer_pattern&oldid=1149928597) (visited on 04/20/2023).
- [6] *Java Singleton Design Pattern Best Practices with Examples | DigitalOcean*. en. URL: <https://www.digitalocean.com/community/tutorials/java-singleton-design-pattern-best-practices-examples> (visited on 04/20/2023).
- [7] *State pattern*. en. Page Version ID: 1119678476. Nov. 2022. URL: [https://en.wikipedia.org/w/index.php?title=State\\_pattern&oldid=1119678476](https://en.wikipedia.org/w/index.php?title=State_pattern&oldid=1119678476) (visited on 04/20/2023).
- [8] Wikipedia contributors. *Client-server model* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 17-April-2023]. 2023. URL: [https://en.wikipedia.org/w/index.php?title=Client%E2%80%93server\\_model&oldid=1146978876](https://en.wikipedia.org/w/index.php?title=Client%E2%80%93server_model&oldid=1146978876).
- [9] *HTTP*. en. Page Version ID: 1149611086. Apr. 2023. URL: <https://en.wikipedia.org/w/index.php?title=HTTP&oldid=1149611086> (visited on 04/20/2023).
- [10] *What is a REST API? | IBM*. en-us. URL: <https://www.ibm.com/topics/rest-apis> (visited on 04/20/2023).
- [11] Wikipedia contributors. *Web scraping* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 17-April-2023]. 2023. URL: [https://en.wikipedia.org/w/index.php?title=Web\\_scraping&oldid=1147883001](https://en.wikipedia.org/w/index.php?title=Web_scraping&oldid=1147883001).

- [12] Wikipedia contributors. *Robots.txt* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 17-April-2023]. 2023. URL: <https://en.wikipedia.org/w/index.php?title=Robots.txt&oldid=1150227804>.
- [13] *What is Accessibility: An Introduction* | *SeeWriteHear*. en-US. Oct. 2020. URL: <https://www.seewritehear.com/learn/what-is-accessibility/> (visited on 04/20/2023).
- [14] *What is accessibility?* - *Learn web development* | *MDN*. en-US. Mar. 2023. URL: [https://developer.mozilla.org/en-US/docs/Learn/Accessibility/What\\_is\\_accessibility](https://developer.mozilla.org/en-US/docs/Learn/Accessibility/What_is_accessibility) (visited on 04/20/2023).
- [15] *8 Ways to Design a Color Blind Friendly Website*. en. URL: <https://www.audioeye.com/post/8-ways-to-design-a-color-blind-friendly-website/> (visited on 05/02/2023).
- [16] *How to Design Your Website for Screen Reader Accessibility*. en. URL: <https://blog.hubspot.com/website/screen-reader-accessibility> (visited on 05/02/2023).
- [17] *Everything you need to know to write effective alt text* - *Microsoft Support*. [Online; accessed 8. May 2023]. May 2023. URL: <https://support.microsoft.com/en-us/office/everything-you-need-to-know-to-write-effective-alt-text-df98f884-ca3d-456c-807b-1a1fa82f5dc2>.
- [18] TechTarget Contributor. *web application (web app)*. [Online; accessed 20-April-2023]. 2023. URL: <https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app>.
- [19] bloomreach.com. *What Is a Single Page Application?* en. URL: <https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application> (visited on 04/20/2023).
- [20] Atlassian. *What is Agile?* en. URL: <https://www.atlassian.com/agile> (visited on 04/20/2023).
- [21] Atlassian. *Scrum - what it is, how it works, and why it's awesome*. en. URL: <https://www.atlassian.com/agile/scrum> (visited on 04/20/2023).
- [22] *Git*. URL: <https://git-scm.com/> (visited on 04/20/2023).
- [23] *What is a Cloud Service?* – *Cloud Services Solutions* - *Citrix*. en. URL: <https://www.citrix.com/solutions/digital-workspace/what-is-a-cloud-service.html> (visited on 04/20/2023).
- [24] *What is Containerization?* - *Containerization Explained* - *AWS*. en-US. URL: <https://aws.amazon.com/what-is/containerization/> (visited on 04/21/2023).
- [25] *What is a relational database?* en-US. URL: <https://www.oracle.com/database/what-is-a-relational-database/> (visited on 04/21/2023).
- [26] Laura Fitzgibbons. “DRY principle”. In: *WhatIs.com* (June 2018). URL: <https://www.techtarget.com/whatis/definition/DRY-principle>.
- [27] *How to check if a given point lies inside or outside a polygon?* Mar. 2023. URL: <https://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon/>.

- [28] Uxdesigninstitute. *7 fundamental UX design principles all designers should know - UX Design Institute*. en-US. June 2022. URL: <https://www.uxdesigninstitute.com/blog/ux-design-principles/> (visited on 04/20/2023).
- [29] Alexander S Gillis. *What Is API Testing? | Definition from TechTarget*. en. Mar. 2023. URL: <https://www.techtarget.com/searchapparchitecture/definition/API-testing> (visited on 04/24/2023).
- [30] *What is SQL? - Structured Query Language (SQL) Explained - AWS*. en-US. URL: <https://aws.amazon.com/what-is/sql/> (visited on 04/24/2023).
- [31] Wikipedia contributors. *Hibernate (framework) — Wikipedia, The Free Encyclopedia*. [Online; accessed 2-May-2023]. 2023. URL: [https://en.wikipedia.org/w/index.php?title=Hibernate\\_\(framework\)&oldid=1146534247](https://en.wikipedia.org/w/index.php?title=Hibernate_(framework)&oldid=1146534247).
- [32] *What Is Open Source Software and How Does It Work? | Synopsys*. en. URL: <https://www.synopsys.com/glossary/what-is-open-source-software.html> (visited on 04/24/2023).
- [33] *CSS: Cascading Style Sheets | MDN*. en-US. Apr. 2023. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS> (visited on 04/24/2023).
- [34] Jordana A. *What Is JavaScript? A Basic Introduction to JS for Beginners*. en-US. Aug. 2021. URL: <https://www.hostinger.com/tutorials/what-is-javascript> (visited on 04/24/2023).
- [35] *TypeScript Introduction*. en-US. URL: [https://www.w3schools.com/typescript/typescript\\_intro.php](https://www.w3schools.com/typescript/typescript_intro.php) (visited on 04/25/2023).
- [36] *An Introduction to JSON | DigitalOcean*. en. URL: <https://www.digitalocean.com/community/tutorials/an-introduction-to-json> (visited on 05/08/2023).
- [37] *GeoJSON*. en. Page Version ID: 1147050991. Mar. 2023. URL: <https://en.wikipedia.org/w/index.php?title=GeoJSON&oldid=1147050991> (visited on 05/08/2023).
- [38] Margaret Rouse. *Prompt-Based Learning*. en-US. Feb. 2023. URL: <https://www.techopedia.com/definition/34832/prompt-based-learning> (visited on 04/24/2023).
- [39] Amazon Web Services. *What is Continuous Integration? - Amazon Web Services*. en-US. URL: <https://aws.amazon.com/devops/continuous-integration/> (visited on 04/25/2023).
- [40] IBM. *Continuous Deployment: An Essential Guide | IBM*. en-us. URL: <https://www.ibm.com/topics/continuous-deployment> (visited on 04/25/2023).
- [41] pgAdmin. *pgAdmin - PostgreSQL Tools*. en. URL: <https://www.pgadmin.org/> (visited on 04/24/2023).
- [42] *Lighthouse overview*. en. URL: <https://developer.chrome.com/docs/lighthouse/overview/> (visited on 05/03/2023).
- [43] *Spring Boot - Code Structure*. en-us. July 2021. URL: <https://www.geeksforgeeks.org/spring-boot-code-structure/> (visited on 04/27/2023).
- [44] *Web Scraping Done Right: Best Practices to ensure Ethical Data Collection & Web Scraping - Merit Data Tech*. en-US. Aug. 2021. URL: <https://www.meritdata-tech.com/resources/blog/data/web-scraping-best-practices-ethical-data-collection/> (visited on 05/09/2023).

- [45] *About us*. [Online; accessed 12. May 2023]. May 2023. URL: <https://www.barentswatch.no/en/about>.
- [46] *Shipping Publications AS*. [Online; accessed 12. May 2023]. May 2023. URL: <http://www.skipslistene.no/prog/reder.asp?vq=GEHKL>.
- [47] *Overview of AIS dataset - AIS Handbook - UN Statistics Wiki*. [Online; accessed 12. May 2023]. May 2023. URL: <https://unstats.un.org/wiki/display/AIS/Overview%2bof%2bAIS%2bdataset>.

# APPENDICES

## A - REQUIREMENTS DOCUMENTATION



# Requirements Documentation

## Introduction

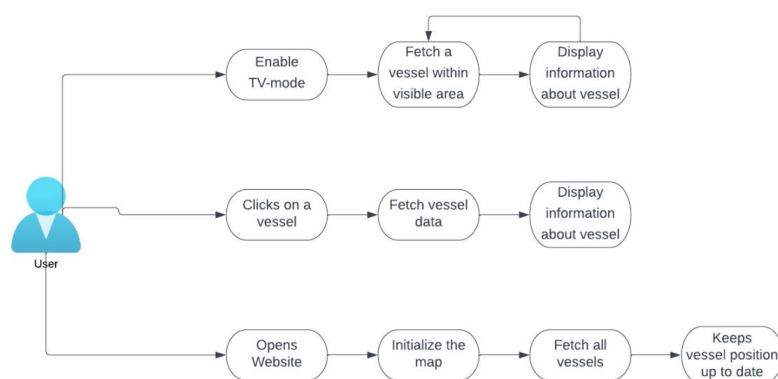
The purpose of this document is to elaborate on the requirements specification on the scope and content of the bachelor thesis “Marine Traffic Portal”. This thesis is a collaboration between Accenture and three NTNU students of NTNU Ålesund spring 2023.

## Objectives of the project

The users of the website will be employees at Accenture Trondheim as well as boat enthusiasts in Trondheim. After discussion with the client at Accenture the following needs and wishes were given for the project:

- To be able to get an overview of all the boats in the fjord and their location.
- To be able to click on a vessel and view information specific to that vessel.
- To be able to enter “TV-mode” were the application selects and views a vessel automatically.
- To have the application publicly available
- To show interesting and useful information about the vessels

## Use case diagram

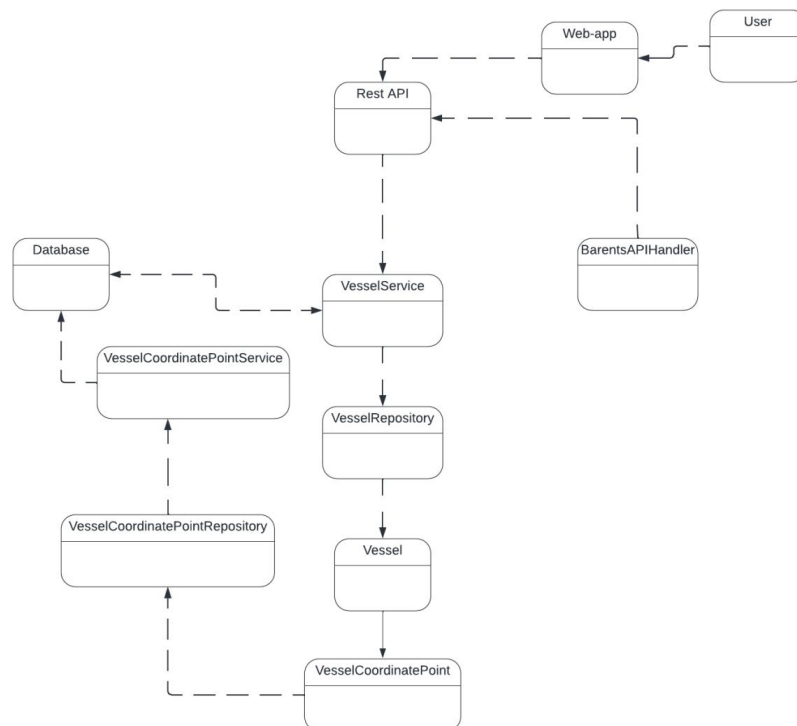


## User Stories

The following stories is from the user's perspective:

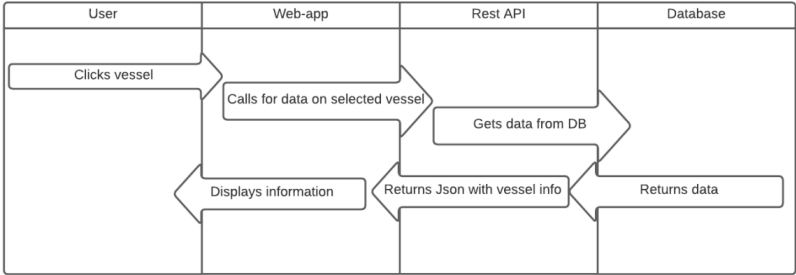
- As a user, I should be able to access the site form my browser.
- As a user, I should be able to click on a vessel to see information about the vessel.
- As a user, I should be able to set the site in TV-mode, and it should display vessels outside the office and change vessels within a time interval.
- As a user, I should be able to see all the vessels in the fjord and the location should be live.

## Domain Model



### Sequence Diagram

Get the information from a selected vessel.



## B - PROJECT PLAN

**Marine Traffic Portal  
Forprosjektplan**

**Versjon <1.0>**

**Revisjonshistorie**

<b>Dato</b>	<b>Versjon</b>	<b>Beskrivelse</b>	<b>Forfatter</b>
16/01/23	0.1	Oppstart	Mathias, Sjur, Sebastian
19/01/23	0.2	Ferdigstilt prosjektplan, venter på signaturer.	Mathias, Sjur, Sebastian
12/04/23	1.0	La til kontrakter	Sjur

---

## Innholdsfortegnelse

1. Mål og rammer	4
1.1 Orientering	4
1.2 Problemstilling / prosjektbeskrivelse og resultatmål	4
1.3 Effektmål	4
1.4 Rammer	4
2. Organisering	4
3. Gjennomføring	4
3.1. Hovedaktiviteter	4
3.2. Milepæler	5
4. Oppfølging og kvalitetssikring	5
4.1 Kvalitetssikring	5
4.2 Rapportering	5
5. Risikovurdering	5
6. Vedlegg	5
6.1 Tidsplan	6
6.2 Adresseliste	6
6.3 Avtaledokumenter	6
6.3.1 Arbeidskontrakt for bachelor-gruppen	6
6.3.2 3-partsavtale	7

---

## 1. Mål og rammer

### 1.1 Orientering

Oppgaven ble tildelt etter ønske fra gruppen. På bakgrunn av generell interesse hos gruppen for oppgaven.

### 1.2 Problemstilling / prosjektbeskrivelse og resultatmål

**Problemstilling:**

Accenture har et ønske om å kunne overvåke skipstrafikken utenfor sine kontorlokaler i Trondheim. Hvordan kan vi best visualisere båttrafikken via skjermene på kontoret?

**Oppgavebeskrivelse:**

Undersøk og lag en prototype på hvordan automatisere sporing av kommersiell marin trafikk i Trondheimsfjorden, og presenter det i en moderne nettside.

Lage en offentlig nettside som viser båter i Trondheimsfjorden. Tilby et offentlig cloud environment som hoster nettsiden og dens funksjonaliteter. Nettsidens brukere er båt entusiaster og ansatte i Accenture. Hent inn og lagre relevant informasjon om båttrafikken, for eksempel – når var denne båten sist inne i fjorden?

**Resultatmål:**

En nettside som skal kunne deles opp i to funksjoner. En inaktiv syklus modus som visualiserer båter som kan sees fra kontorvinduene uten input fra bruker. Og en interaktiv modus hvor bruker kan bruke et kart over Trondheimsfjorden aktivt og velge selv hvilke båter en ønsker å se nærmere på.

### 1.3 Effektmål

Målet for gruppen er å lage en konkurransedyktig nettside som står i henhold til moderne standarder med tanke på design, funksjonalitet og brukervennlighet.

Vi ønsker å lage en oversiktlig og brukervennlig nettside. Hvor de ansatte kan lett få oversikt over båter i fjorden.

### 1.4 Rammer

Under prosjektets gang har gruppen behov for skytjeneste til å hoste nettsiden. Microsoft Azure er foretrukket dersom det er mulig.

## 2. Organisering

Accenture er oppdragsgiver for prosjektet. NTNU bidrar med veileder til prosjektet.

## 3. Gjennomføring

### 3.1. Hovedaktiviteter

Gruppen vil jobbe med SCRUM arbeidsmetodikk gjennom prosjektet. Av den grunn vil arbeidsoppgaver tildeles underveis. Det vil bli tatt en vurdering hver uke for hvilke arbeidsoppgaver som skall utføres. Alle på gruppen er pliktet til å føre dokumentasjon under arbeidsprosessen. Dette føres i Confluence. Resultat av fullførte oppgaver deles mellom gruppemedlemmene med bruk av versjonskontroll i GitHub.



### 3.2. Milepæler

- 16.01.23 – Oppstart
- 19.01.23 – Første møte med oppdragsgiver
- Sprintmøter gjennomføres i slutten hver uke
- 22.05.23 - Innlevering

## 4. Oppfølging og kvalitetssikring

### 4.1 Kvalitetssikring

Alle tidskritiske milepæler blir gjennomgått i internmøter ved planlagt ferdigstillelse dato.

### 4.2 Rapportering

Gruppen vil lage sprint-rapporter etter hver sprint. Hver uke. Rapportene sendes til veileder i bedrift og NTNU.

Møte med oppdragsgiver og veileder vil i starten foregå ukentlig, med mulighet for endring senere ved behov.

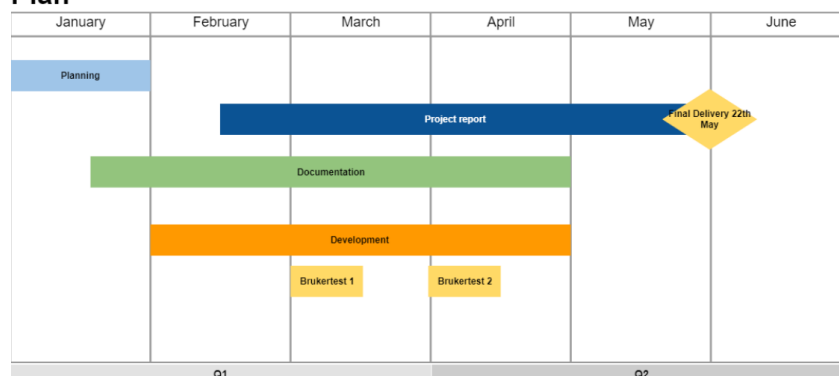
## 5. Risikovurdering

Problem	Sannsynlighet	Tiltak
En av gruppelemmene blir fraværende over lengre tid	Lav	Ha et godt og sunt arbeidsforhold
Konflikt i gruppen	Moderat/Lav	Ukentlige møter og klare forventninger til arbeidsinnsats og resultater.
Arbeidsmengde blir for stor	Lav	Fordele arbeidet jevnt over semesteret, samt dele opp prosjektet i mindre deler med spesifiserte datoer og jobbe målrettet mot disse.

## 6. Vedlegg

Forprosjektplan, Poster, Arbeidskontrakt

## 6.1 Tidsplan

**Marine Traffic Development Plan**

## 6.2 Adresseliste

Sebastian Nilsen, Student NTNU  
 +47 941 98 166  
[sebasn@stud.ntnu.no](mailto:sebasn@stud.ntnu.no)  
 Kirkegata 39, 6005 Ålesund

Mathias Jørgensen, Student NTNU  
 +47 466 34 708  
[mathjo@stud.ntnu.no](mailto:mathjo@stud.ntnu.no)  
 Kirkegata 39, 6005 Ålesund

Sjur Gustavsen, Student NTNU  
 +47 960 42 029  
[sjurgu@stud.ntnu.no](mailto:sjurgu@stud.ntnu.no)  
 Lihauggata 8, 6002 Ålesund

## 6.3 Avtaledokumenter

## 6.3.1 Arbeidskontrakt for bachelor-gruppen

Ligger vedlagt som «Arbeidskontrakt»

6.3.2 3-partsavtale

Ligger vedlagt som «NTNU Standardavtale Mathias Jørgensen, NTNU Standardavtale Sebastian Nilse og NTNU Standardavtale Sjur Gustavsen»

## C - COLLABORATION AGREEMENT

# Arbeidskontrakt for Bacheloroppgave Marine Traffic Portal

Medlemmer: Sjur Gustavsen, Mathias Jørgensen og Sebastian Nilsen

## Formål

Denne samarbeidsavtalen regulerer partenes ansvar, roller og rettigheter i forbindelse med gjennomføring av Bacheloroppgave Marine Traffic Portal i henhold til prosjektbeskrivelse som inngår som vedlegg 1 til denne avtale (heretter kalt «Prosjektet»).

Samarbeidsavtalen skal sikre at Prosjektet gjennomføres og dokumenteres i henhold til relevant regelverk og anerkjente etiske normer.

## Roller og oppgavefordeling

Partene har et selvstendig ansvar for organisering og utførelse av den del av Prosjektet som gjennomføres i egen institusjon, og at dette skjer i henhold til relevant regelverk og formelle godkjenninger.

## Rollefordelinger

### **Teamleder:**

Teamleder har et overordnet ansvar for prosjektmedarbeiderne, møteinnkallinger og kommunikasjon.

Kontaktinformasjon: Sjur Gustavsen, [sjurgu@stud.ntnu.no](mailto:sjurgu@stud.ntnu.no), +4796042029

### **Dokumentansvarlig:**

Dokumentansvarlig ansvarsområde omhandler alle dokumenter og vedlegg.

Kontaktinformasjon: Mathias Jørgensen, [mathjo@stud.ntnu.no](mailto:mathjo@stud.ntnu.no), +47 46634708

### **Kvalitetssikring:**

Kvalitetssikring sitt overordnede ansvarsområde er å sikre at alle dokumenter og loggføring blir gjennomført. Samt sørge for at alle frister blir overholdt.

Sebastian Nilsen, [sebasn@stud.ntnu.no](mailto:sebasn@stud.ntnu.no), +47 941 98 166

Eksempler på roller: Teamledelse, dokumentansvarlig, Kvalitetssikring

Hva innebærer de ulike rollene, hvordan ivaretas de, hvem har ansvar for hva.

## Prosedyrer (hvordan gjør man ting?)

### A. Møteinnkalling

Gruppen har internt møte i slutten av hver sprint (ca. hver 2. uke). Møte innkalles med 4 dagers forvarsel.

Tidspunkt for møter med arbeidsgiver/kunde avtales nærmere etter første møte med arbeidsgiver/kunde.

Når skal man ha møter. Hvordan innkalles det.

### B. Varsling ved fravær eller andre hendelser

Gruppen kommuniserer internt via Messenger. Dersom man har planlagt fravær skal dette informeres om i god tid. Hvis noe uanmeldt hender skal man varsle umiddelbart dersom man ikke har mulighet å møte.

### C. Dokumenthåndtering

Dokumenter blir i all hovedsak lagret og samskrevet i Microsoft Teams. Jira brukes som issue tracking. Og Git brukes som versjonshåndtering. Dokumentansvarlig har overordnet ansvar for at dette blir opprettholdt.

### D. Innleveringer av gruppearbeider

Alle filer skal leveres i henhold til frister. Alle filer som skal leveres blir felles kvalitets kontrollert og ferdigstilt av alle på gruppen før de leveres.

## Interaksjon (Hvordan opptrer man sammen?)

### A. Oppmøte og forberedelse

Alle gruppe-medlemmer skal møte til avtalt tid. Kommer noe i veien for avtalt møte tidspunkt skal medlemmet varsle de andre. Det forventes at alle har gjort kjent med innholdet som skal gjennomgås i møter, og stiller forberedt.

Gruppen har faste arbeidstider alle hverdager fra 09-15, om ikke annet er avtalt.

### B. Tilstedeværelse og engasjement

Det forventes at alle er til stedet og bidrar mens gruppen arbeider. Pauser avtales

fortløpende under arbeidet.

C. *Hvordan støtte hverandre*

Vår filosofi, er at ingen ideer er dumme. Alle skal ha mulighet til å bli hørt, og samværet i gruppen er en viktig del av prosjektet. Derfor vil trivsel bli høyt prioritert, og alle medlemmer har ansvar for å bidra til dette.

D. *Uenighet, avtalebrudd*

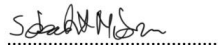
Når uenighet oppstår i gruppen, bestemmer flertallet. Eventuelt kan veileder bidra. Om avtalebrudd oppstår vil veileder kontaktes.

Gustavsen, Sjur  
11.01.2023

Nilsen, Sebastian  
11.01.2023

Jørgensen, Mathias  
11.01.2023







## D - WIREFRAME







## Info card component

---

### Chosen design



### Other design versions



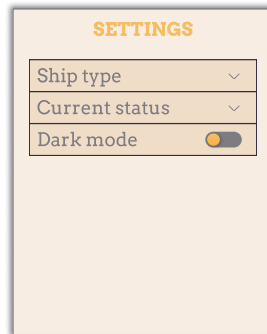
## Selected vessel design

---

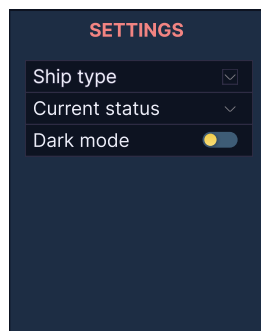
## Settings card design

---

Light mode



Dark mode



## E - SYSTEM DOCUMENTATION

# System Documentation

## Introduction

This document details the technical aspects of the system developed during the project. It covers topics such as the project's code structure, classes utilized in the application and server, as well as the configuration of the database.

## Architecture

The system's architecture is divided into three sections. The backend, the database, and the frontend. They are all hosted in an Azure cloud environment. The flowchart below shows the structure.

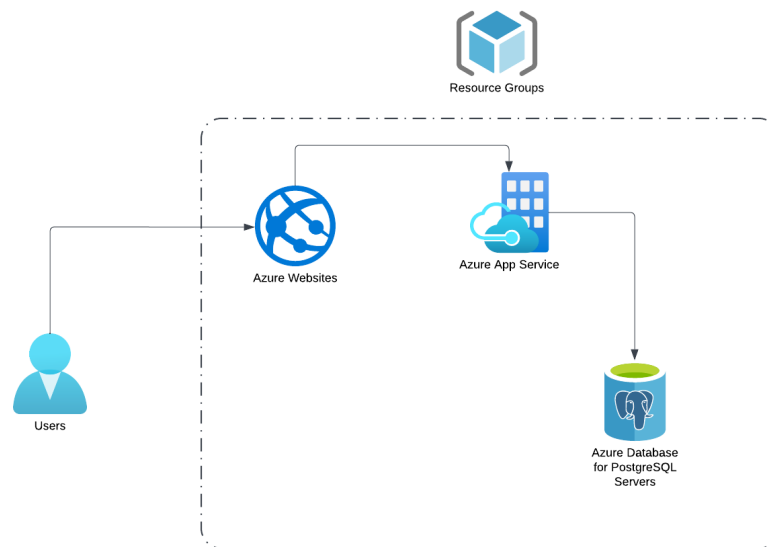


Figure 1: Shows the systems architecture.

## Project Structure

### Frontend

The tree structure shown below visualizes all folders and config files starting from the root of the workspace.



Figure 2: Shows the tree structure in the frontend.

- app / (contains the main .tsx and .css files for the application. Controls the logic and layout)
- components / (contains all the react components for the application)
- env.local (contains the token for accessing MapBox)
- .gitignore (Specifies files that Git should ignore)
- prettierrc.json (Configuration file for Prettier code formatting tool,)
- next.config.js (Used to customize various aspects of the build process and runtime behavior of the application)
- package-lock.json (Provides version information of all npm packages installed)
- package.json (Configure npm package dependencies for the project)



- tsconfig.json (Configuration file used in TypeScript. Specifies compiler options and project settings)

## Backend

The Spring Boot server is divided into three main folders. The controller folder, the service folder, and the repositories folder. The tree diagram below shows what these folders contain. The project also had other folders containing logic and/or functionality needed for the application to work.

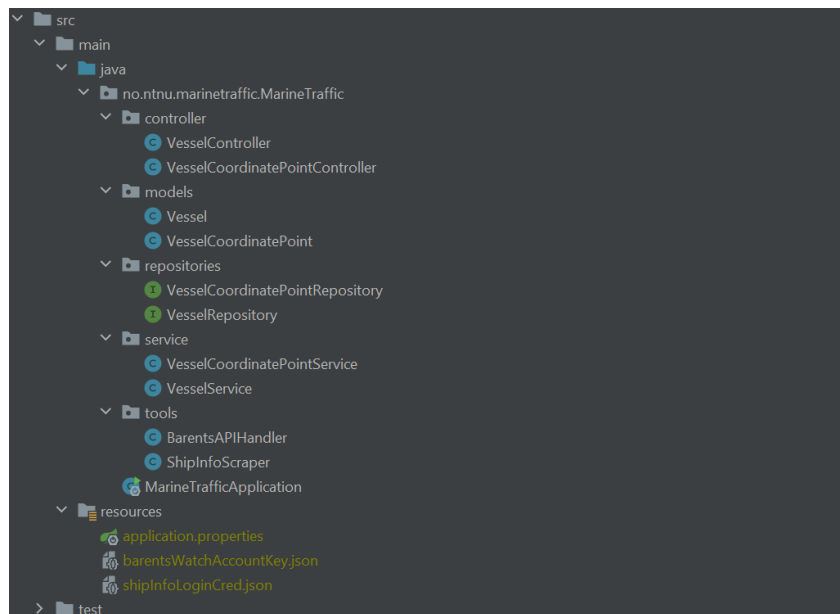


Figure 3: Shows the tree structure in the backend.

## Class diagram

The document below shows the class diagram for the backend server spring boot server. It contains 8 classes, 1 main class (MarineTrafficApplication) that starts the application, 2 model classes (Vessel and VesselCoordinatePoint) that defines a vessel and its position. 2 tools classes (BarrentsAPIHandler and ShipScrapper). The first handles the data stream from the BarentsWatch API and creates the vessels. The second ShipInfoScrapper is a web scrapper gathering information about a specific vessel. The last 3 classes are the Controller, Service and Repository classes. These classes handle the logic, endpoints, and database communication of the application.

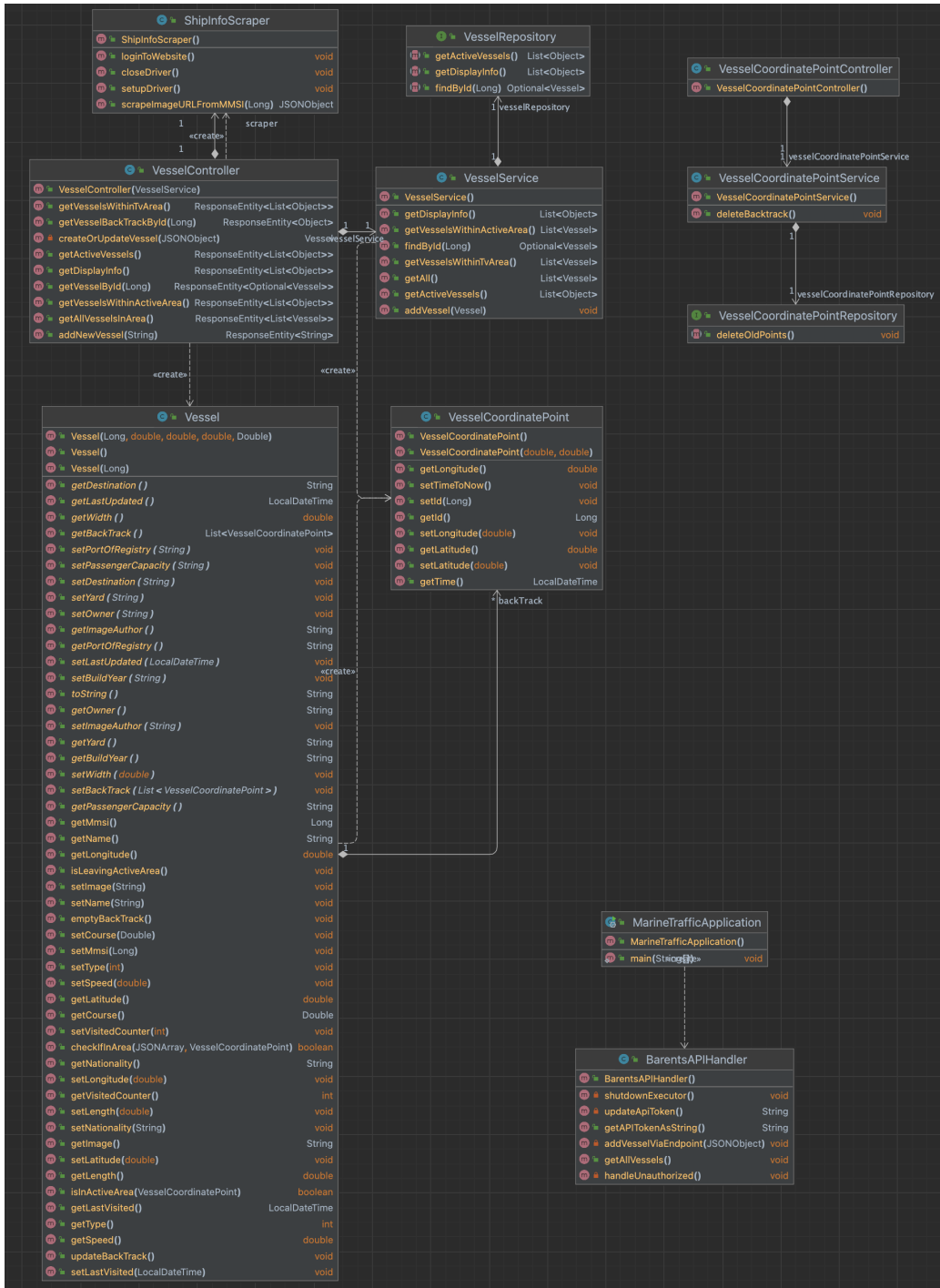


Figure 4: Shows the class diagram in the backend.

## Database Model

Figure 3 below shows the relation and table structure of the database. It consists of two tables, and there is a one-to-many relation from the vessel to the backtrack table, and many-to-one relation from backtrack to the vessel table.

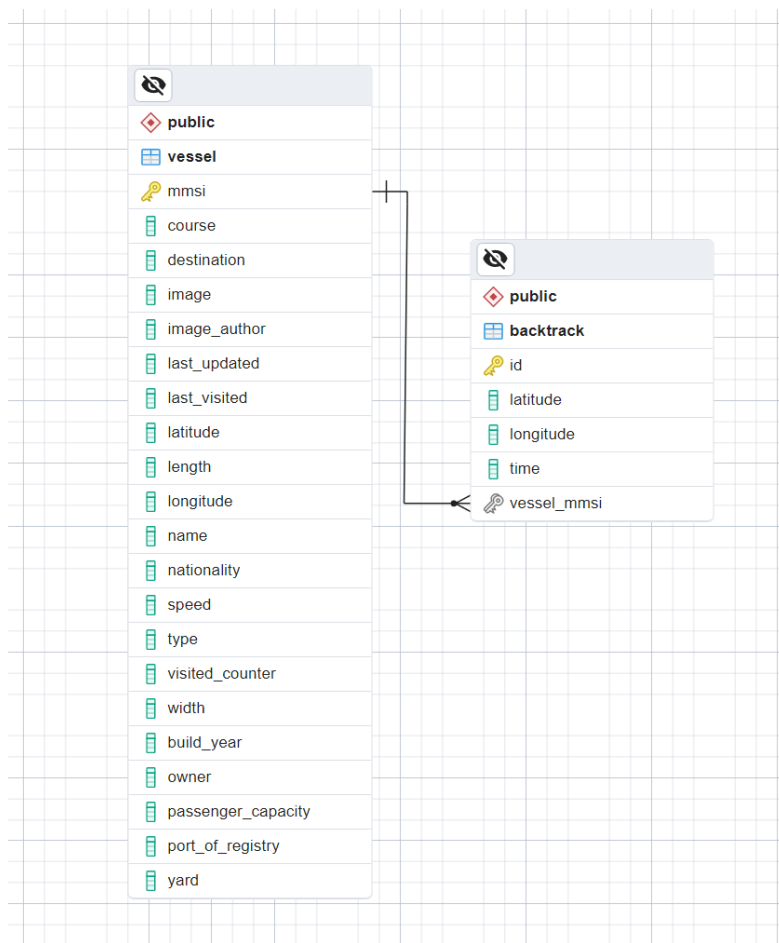


Figure 5: Diagram showing relation between the two tables in the database.

## Server Service

### Vessel resources

#### List of all vessels

*/getAll* – Gets a list of all the boats in the system.

#### Get Active boats

*/getActive* – Gets a list of all the active boats in the area of interest.

#### Add

*/add* – Adds a new boat to the system

#### Get boat

*/get{mmsi}* – Requires the boat ID (mmsi) and returns the boat information from the system.

#### Get backtracking

*/getBackTrack/{mmsi}* - Requires the boat ID (mmsi) and returns the last 50 stored positions as backtrack.

#### Get boats for TV view

*/tvArea* – get the boats inside the area of vision from the Accenture offices.

## Deployment

### Frontend

The website is available for everyone on the internet. The application is deployed on an Azure web app service, which autogenerates a domain. Continuous deployment is set up by GitHub actions, so whenever someone pushes something to the main branch in GitHub it deploys the new updates to the website.

The following code libraries are used in the app:

- **Mapbox** – A map library used for the display of Trondheimsfjorden. (Mapbox, 2023)

## Backend

The backend is deployed at Azure, using an Azure App Service. The team have set up a PostgreSQL database in Azure, as well as a container registry. In Docker one can create a compose file (docker image) where it is possible to create a spring boot container and a MySQL database. The image is then pushed to the container registry in Azure, and updates are then deployed to the backend.

Other libraries used in the backend:

- **Selenium** – Webdriver used for scrapping from ship-info.com. (Selenium, 2023)

## Documentation of source code

### Frontend

The code in the frontend is documented in-line with how to document in React. Each method is given a describing name, as well as using the “/” notation to give smaller comments to specify how things work.

### Backend

The whole backend is documented in detail using Javadoc. The team has used a built-in feature in IntelliJ to generate and show the documentation. This creates an HTML file containing all the documentation of the backend.

## Testing

Postman is mainly used to test the backend, and its endpoints. The website is tested publicly as it is available for everyone. The backend has a configured docker container, and the frontend is set up with GitHub actions, which makes it easy to push any updates.

## References

(2023, 04 28). Retrieved from Selenium:

<https://www.selenium.dev/documentation/webdriver/>

(2023, 04 28). Retrieved from Mapbox: <https://docs.mapbox.com/>

## F - SURVEY



## Accenture båtinno

7

Responses

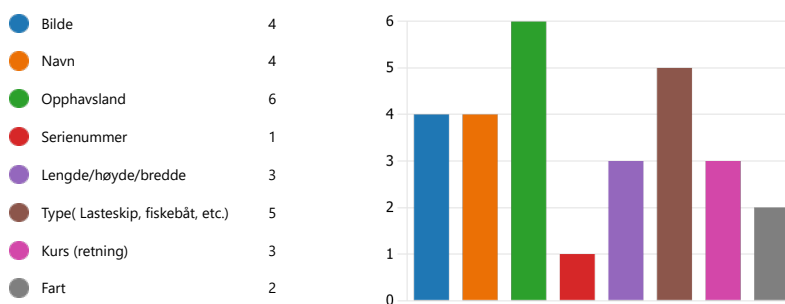
05:32

Average time to complete

Active

Status

1. Det finnes mye generell informasjon om båtene. Hvilke av disse synes du hadde vært mest nyttig?



2. Hvilken annen informasjon om en båt synes du det hadde vært interessant/gøy å se på en slik side? Vennligst oppgi ting i prioritert rekkefølge

7

Responses

Latest Responses

"Seilingshistorikk"

"Ruten til fartøyet - hvor den har vært og hvor den skal "

"Hvor mange kan være på båten (f. eks hvor mange passasjere på et ...

3. Hvilke funksjoner ønsker du deg på en slik side?

7

Responses

Latest Responses

"Klikkbare valg"

"Hvor mange som er om bord(?)"

"Se oversikt over antall/typer båter på et område. Se hvor båten ko...

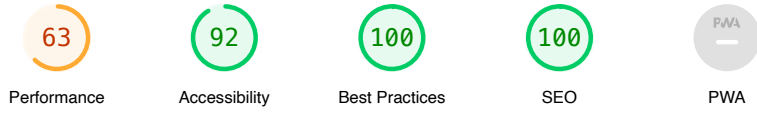
4. Noen tanker/inns spill du ønsker å komme med til siden?

2  
Responses

Latest Responses

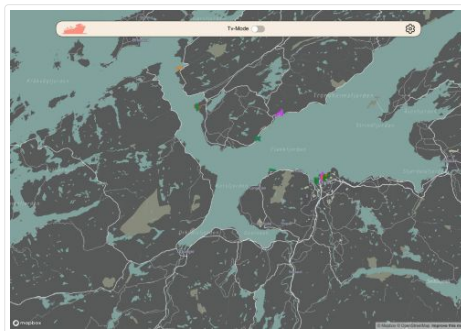
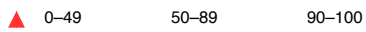
---

## G - LIGHTHOUSE RESULT



### Performance

Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. [See calculator.](#)



#### METRICS

[Expand view](#)

First Contentful Paint

**0.4 s**

Largest Contentful Paint

**1.4 s**

▲ Total Blocking Time

**1,650 ms**

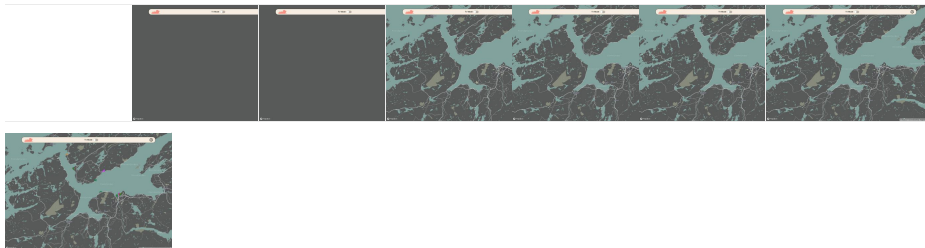
Cumulative Layout Shift

**0**

Speed Index

**1.9 s**

[View Treemap](#)



Show audits relevant to: [All](#) [FCP](#) [LCP](#) [TBT](#) [CLS](#)

## OPPORTUNITIES

Opportunity	Estimated Savings
Eliminate render-blocking resources	0.14s <span>▼</span>

These suggestions can help your page load faster. They don't [directly affect](#) the Performance score.

## DIAGNOSTICS

Minimize main-thread work — 2.8 s	<span>▼</span>
<input type="radio"/> Avoid chaining critical requests — 2 chains found	<span>▼</span>
<input type="radio"/> User Timing marks and measures — 2 user timings	<span>▼</span>
<input type="radio"/> Keep request counts low and transfer sizes small — 20 requests • 524 KiB	<span>▼</span>
<input type="radio"/> Largest Contentful Paint element — 1 element found	<span>▼</span>
<input type="radio"/> Avoid long main-thread tasks — 12 long tasks found	<span>▼</span>

More information about the performance of your application. These numbers don't [directly affect](#) the Performance score.

## PASSED AUDITS (31)

Hide

Properly size images	<span>▼</span>
Defer offscreen images	<span>▼</span>
Minify CSS	<span>▼</span>
Minify JavaScript	<span>▼</span>
Reduce unused CSS	<span>▼</span>
Reduce unused JavaScript — Potential savings of 169 KiB	<span>▼</span>
Efficiently encode images	<span>▼</span>
Serve images in next-gen formats	<span>▼</span>
Enable text compression — Potential savings of 2 KiB	<span>▼</span>

Preconnect to required origins	▼
Initial server response time was short — Root document took 170 ms	▼
Avoid multiple page redirects	▼
<input type="radio"/> Preload key requests	▼
Use video formats for animated content	▼
Remove duplicate modules in JavaScript bundles	▼
Avoid serving legacy JavaScript to modern browsers	▼
<input type="radio"/> Preload Largest Contentful Paint image	▼
Avoids enormous network payloads — Total size was 524 KiB	▼
Uses efficient cache policy on static assets — 0 resources found	▼
Avoids an excessive DOM size — 179 elements	▼
JavaScript execution time — 0.8 s	▼
All text remains visible during webfont loads	▼
Minimize third-party usage — Third-party code blocked the main thread for 0 ms	▼
<input type="radio"/> Lazy load third-party resources with facades	▼
<input type="radio"/> Largest Contentful Paint image was not lazily loaded	▼
<input type="radio"/> Avoid large layout shifts	▼
Uses passive listeners to improve scrolling performance	▼
Avoids <code>document.write()</code>	▼
<input type="radio"/> Avoid non-composited animations	▼
<input type="radio"/> Image elements have explicit <code>width</code> and <code>height</code>	▼
Has a <code>&lt;meta name="viewport"&gt;</code> tag with <code>width</code> or <code>initial-scale</code>	▼



## Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

### ARIA

▲ [\[aria-\\*\] attributes do not match their roles](#) ▼

These are opportunities to improve the usage of ARIA in your application which may enhance the experience for users of assistive technology, like a screen reader.

### NAVIGATION

[The page contains a heading, skip link, or landmark region](#) ▼

These are opportunities to improve keyboard navigation in your application.

### ADDITIONAL ITEMS TO MANUALLY CHECK (10)

Show

These items address areas which an automated testing tool cannot cover. Learn more in our guide on [conducting an accessibility review](#).

### PASSED AUDITS (17)

Hide

[\[aria-hidden="true"\]](#) is not present on the document `<body>` ▼

[\[role\]](#)s have all required [\[aria-\\*\]](#) attributes ▼

Elements with an ARIA [\[role\]](#) that require children to contain a specific [\[role\]](#) have all required children. ▼

[\[role\]](#)s are contained by their required parent element ▼

<code>[role]</code> values are valid	▼
<code>[aria-*]</code> attributes have valid values	▼
<code>[aria-*]</code> attributes are valid and not misspelled	▼
ARIA IDs are unique	▼
<code>[user-scalable="no"]</code> is not used in the <code>&lt;meta name="viewport"&gt;</code> element and the <code>[maximum-scale]</code> attribute is not less than 5.	▼
<code>[aria-hidden="true"]</code> elements do not contain focusable descendents	▼
Background and foreground colors have a sufficient contrast ratio	▼
Document has a <code>&lt;title&gt;</code> element	▼
<code>[id]</code> attributes on active, focusable elements are unique	▼
<code>&lt;html&gt;</code> element has a <code>[lang]</code> attribute	▼
<code>&lt;html&gt;</code> element has a valid value for its <code>[lang]</code> attribute	▼
Links have a discernible name	▼
No element has a <code>[tabindex]</code> value greater than 0	▼
NOT APPLICABLE (25)	Show



Best Practices

TRUST AND SAFETY



---

Ensure CSP is effective against XSS attacks ▼

---

GENERAL

---

▲ Missing source maps for large first-party JavaScript ▼

---

PASSED AUDITS (12)

Hide

---

Uses HTTPS ▼

---

Avoids requesting the geolocation permission on page load ▼

---

Avoids requesting the notification permission on page load ▼

---

Allows users to paste into input fields ▼

---

Displays images with correct aspect ratio ▼

---

Serves images with appropriate resolution ▼

---

Page has the HTML doctype ▼

---

Properly defines charset ▼

---

Avoids [unload](#) event listeners ▼

---

Avoids deprecated APIs ▼

---

No browser errors logged to the console ▼

---

No issues in the [Issues](#) panel in Chrome Devtools ▼

---

NOT APPLICABLE (2)

Show



## SEO

These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on [Core Web Vitals](#). [Learn more about Google Search Essentials](#).

### ADDITIONAL ITEMS TO MANUALLY CHECK (1)

Show

Run these additional validators on your site to check additional SEO best practices.

### PASSED AUDITS (9)

Hide

Has a <code>&lt;meta name="viewport"&gt;</code> tag with <code>width</code> or <code>initial-scale</code>	▼
Document has a <code>&lt;title&gt;</code> element	▼
Document has a meta description	▼
Page has successful HTTP status code	▼
Links have descriptive text	▼
Links are crawlable	▼
Page isn't blocked from indexing	▼
Document has a valid <code>hreflang</code>	▼
Document avoids plugins	▼

### NOT APPLICABLE (5)

Show

## H - FEEDBACK

# Marine traffic portal



**Bachelor thesis feedback**



## Practical information

### Client

Accenture AS  
Havnegata 9  
7010 – Trondheim

### Supervisor from Accenture

Christian Nymo

### Supervisor from NTNU

Kjell Inge Tomren

### Students

Sjur Gustavsen  
Sebastian Nilsen  
Mathias Jørgensen



## Background

Accenture Trondheim is very fortunate to be located in beautiful office spaces at Pirsenteret in Trondheim with an amazing panoramic view of Trondheimsfjorden and its surroundings. This has sparked a curiosity and a desire from the employees to learn more about the ships we see both passing by and docking right outside our windows. To help nurture this newly found interest and to hopefully create points of social interactions at the many wall-mounted monitors placed around the offices, Accenture Trondheim has submitted the following bachelor thesis assignment to NTNU.

## Assignment

Create a public website that shows current boats in Trondheimsfjorden. Provision a public cloud environment that hosts the marine traffic website and its functionality. The audience for the website is boat enthusiasts, as well as the Accenture employees. Gather data about traffic and present interesting data points. For example - when was the last time this boat was here?

## Feedback on the finished solution and general observations

Accenture Trondheim is very satisfied with the final solution, this is not just the opinion of the supervisor, but also the impression I have from the other employees at the office. The scope of this bachelor thesis assignment is both wide and narrow in nature, meaning that the expected outcome, i.e., the finished solution is well understood by the target audience, however the technologies involved and the road to get there is very much up to the students to decide. This is intentional from Accenture's side to reflect certain aspects that may be encountered in some client engagements.

It has solely been up to the group to decide on any frameworks, programming languages, development frameworks, final design decisions and hosting platforms etc. used in the target solution as well as provide reasoning for those decisions.

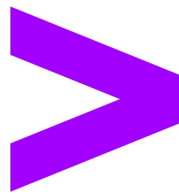
The students have successfully captured the end-user needs and transformed these to functional code, something which is crucial for success.

As a supervisor I have been very impressed with the drive that the group has demonstrated and their ability to incorporate any feedback from both Accenture and NTNU. Their willingness to learn and ability to adapt is also something I would like to point out as a positive.

The final solution meets the requirements outlined in the assignment and does exactly what a marine traffic portal catered to a specific location is supposed to do. Provide real-time updates on the marine ship movements in a predefined grid in an easy-to-understand graphical interface primarily designed for large monitors.

In addition, I would like to add that out of the three groups under my supervision working on the same assignment, this is the only group that found a solution for providing non-static pictures of the ships in question using a scraper in agreement with Skipslistene. This really gives the final solution that final touch.

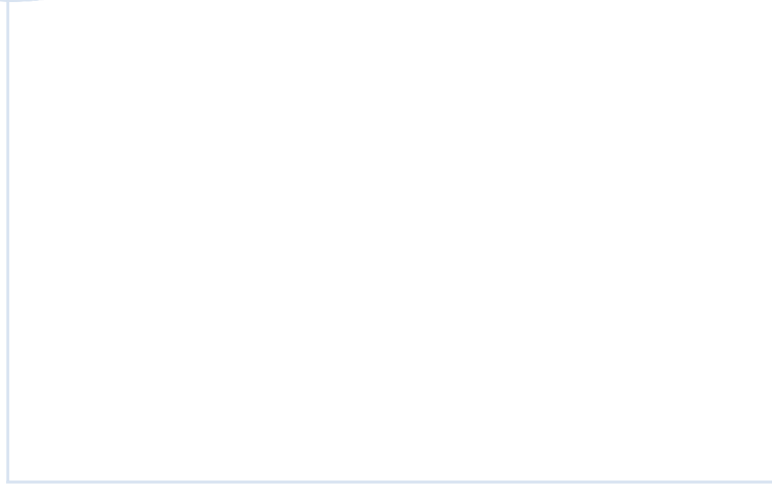
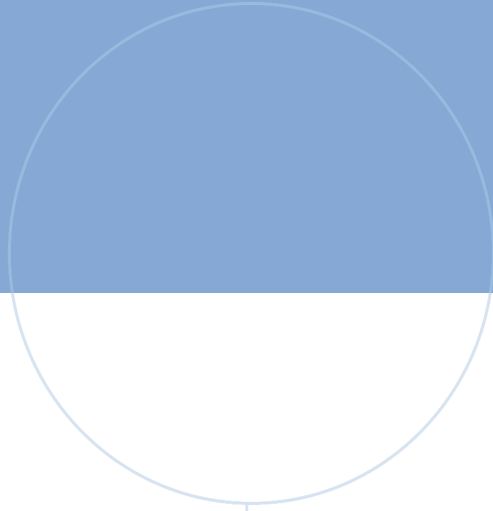
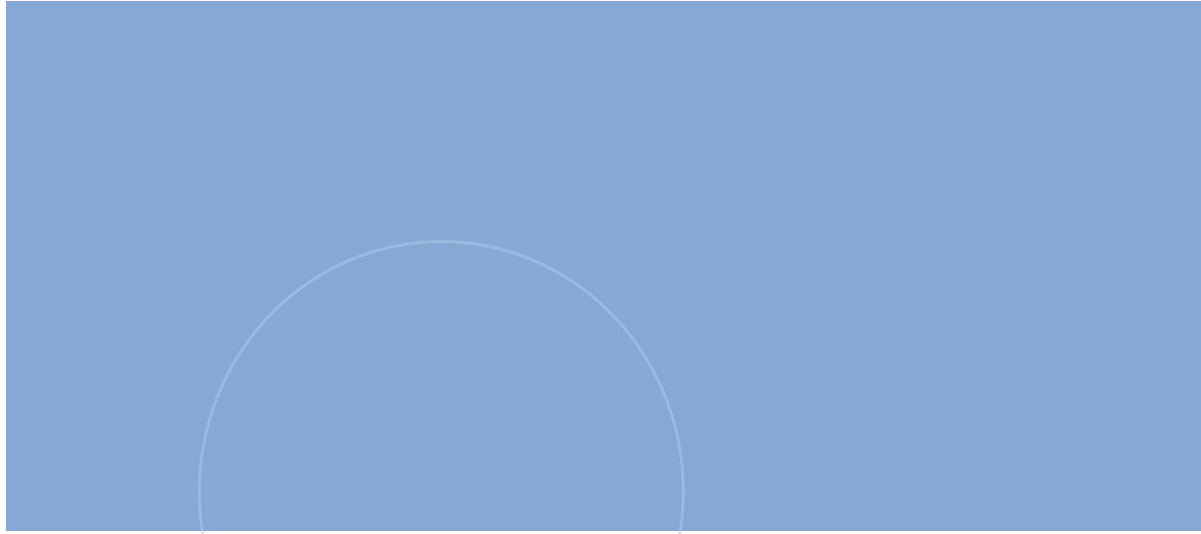
On a personal note, I would like to point out that it has been a joy to work with Sjur, Sebastian and Mathias and I am very proud of the result that they have produced. I wish them all the best in future endeavors after graduation.



Copyright © 2023 Accenture  
All rights reserved.  
Accenture and its logo are trademarks of Accenture.







 **NTNU**

Norwegian University of  
Science and Technology