

Didrik Eilertsen

Eirik Dahle

Jones Maaroufi

# Interactive Visualization Platform for Fish Health

Bachelor thesis in Computer Science

Supervisor: Di Wu

May 2023

NTNU (Norges Teknisk Naturvitenskapelige Universitet)

Norwegian University of Science and Technology

Faculty of Information Technology and Electrical Engineering

Department of ICT and Natural Sciences





## Abstract

PatoGen, a leading company specializing in fish health and disease diagnostics, plays a pivotal role in the aquaculture industry by providing fish farms with invaluable data through comprehensive test analysis. However, managing and comprehending the vast amount of generated data poses a significant challenge for fish farm managers and other stakeholders.

To address this issue, this bachelor thesis aimed to develop a user-friendly web-based application that visualizes fish health and disease risk using a traffic light system, leveraging the data provided by PatoGen. The traffic light system, a widely recognized method for representing risk levels, was chosen for its intuitive ability to convey complex information in a simple and understandable manner. By utilizing this application, fish farm managers and stakeholders can easily assess the risk levels associated with various diseases, enabling them to make informed decisions regarding farm management practices.

The developed application is seamlessly integrated with the PatoGen database, ensuring real-time and accurate information for users. The project employed Java and Springboot for efficient communication between the React frontend and the database, resulting in a well-integrated system. To visualize the color-coded risks, a customizable map platform using the Google Maps API was implemented.

Throughout the development process, an agile methodology was followed to ensure the delivery of a fully functional product. Clear goals and individual tasks were established using Jira and Confluence, aiding in progress tracking and adherence to time constraints. Regular and effective communication with the customer and supervisor proved crucial in adjusting the project trajectory when faced with challenges or unclear requirements.

In conclusion, the result stands as a highly advantageous tool for the end users. By leveraging the power of advanced technology, it greatly simplifies the challenging task of explaining intricate data.

## Sammendrag

PatoGen, et ledende selskap som spesialiserer seg på diagnostikk av fiskehelse og sykdommer, spiller en avgjørende rolle i akvakulturindustrien ved å gi oppdrettsanlegg uvurderlige data gjennom omfattende test-analyser. Imidlertid utgjør håndtering og forståelse av den store mengden genererte data en betydelig utfordring for eiere av oppdrettsanlegg og andre interessenter.

For å løse dette problemet hadde denne bacheloroppgaven som mål å utvikle en brukervennlig nettbasert applikasjon som visualiserer fiskehelse og sykdomsrisiko ved hjelp av et trafikklys-system, ved å dra nytte av dataene som PatoGen leverer. Trafikklys-systemet, en anerkjent metode for å representere risikonivåer, ble valgt for sin intuitive evne til å formidle kompleks informasjon på en enkel og forståelig måte. Ved å bruke denne applikasjonen kan fiskefarmeiere og interessenter enkelt vurdere risikonivåene forbundet med ulike sykdommer, slik at de kan ta informerte beslutninger om driftspraksis på gården.

Den utviklede applikasjonen er sømløst integrert med PatoGen-databasen, slik at brukerne får sanntidsinformasjon som er nøyaktig. Prosjektet benyttet Java og Springboot for effektiv kommunikasjon mellom React-frontenden og databasen, noe som resulterte i et velfungerende system. Fargekodede risikoer vises deretter på et kart hentet fra Google Maps API.

I løpet av utviklingsprosessen ble det fulgt en streng smidig metodikk for å sikre levering av et fullt funksjonelt produkt. Tydelige mål og individuelle oppgaver ble etablert ved hjelp av Jira og Confluence, noe som hjalp med å spore fremgangen og overholde tidsbegrensninger. Regelmessig og effektiv kommunikasjon med kunden og veilederen viste seg å være avgjørende for å tilpasse prosjektets kurs når det oppstod utfordringer eller uklare krav.

Til slutt står resultatet som et svært fordelaktig verktøy for sluttbrukerne. Ved å utnytte kraften av avansert teknologi, forenkler det betydelig den utfordrende oppgaven med å visualisere komplisert data.

## Preface

### Why did we choose this assignment?

The task "Risk model and web app" was the obvious candidate for our choice of bachelor thesis. This is primarily because we wanted to create a simple, visual, and user-friendly website, as well as the group members being both familiar and interested with the technologies used in web development. Also, fish farming is an industry with interesting challenges and issues. Everyone in the group wanted to gain experience from the ocean and fishing industry. All members of the group also live in Ålesund and are familiar with the fact that the city has a large market share within the fish industry.

### Supporters

Thanks to the people who have helped the group complete the process:

- Client: PatoGen AS.
- Client Supervisor/Reference person: Noralf Gamlem (Thank you for always being available and providing us with necessary support when needed).
- Project Supervisor: Di Wu (Thank you Di for Giving helpful feedback and supportive suggestions throughout the entire process).

# Contents

<b>Abstract</b> .....	<b>iii</b>
<b>Sammendrag</b> .....	<b>iv</b>
<b>Preface</b> .....	<b>v</b>
<i>Why did we choose this assignment?</i> .....	v
<i>Supporters</i> .....	v
<b>Contents</b> .....	<b>vi</b>
<b>List of code examples</b> .....	<b>x</b>
<b>List of figures</b> .....	<b>x</b>
<b>Before reading</b> .....	<b>1</b>
<i>Glossary and Phrases</i> .....	1
<i>Acronyms and jargons</i> .....	4
<b>1 Introduction</b> .....	<b>5</b>
1.1 <i>Background</i> .....	5
1.2 <i>Problem</i> .....	5
1.3 <i>Requirements</i> .....	6
1.3.1 <i>Confidentiality</i> .....	6
1.3.2 <i>User authentication</i> .....	6
1.3.3 <i>Data visualization</i> .....	6
1.3.4 <i>Graphical interface</i> .....	7
1.4 <i>Traffic Light Labeling</i> .....	7
<b>2 Theory</b> .....	<b>8</b>
2.1 <i>Client-server communication</i> .....	8
2.2 <i>Relational database</i> .....	8
2.3 <i>Agile development</i> .....	8
2.4 <i>Coupling and cohesion</i> .....	9
2.5 <i>Testing</i> .....	9
2.5.1 <i>Unit Tests</i> .....	9
2.5.2 <i>API Testing</i> .....	9
2.6 <i>Continuous integration</i> .....	10
2.7 <i>Client-side processing</i> .....	10
2.8 <i>Server-side processing</i> .....	10

2.9 Polymerase Chain Reaction.....	11
2.10 Cycle Threshold Values .....	11
2.11 Human-Computer Interaction.....	11
<b>3 Methodology and tools.....</b>	<b>12</b>
3.1 Planning phase.....	12
3.1.1 Pre-project plan .....	12
3.1.2 Use cases .....	12
3.1.3 Wireframes & Design .....	14
3.1.4 Research .....	14
3.2 Tools .....	14
3.2.1 SQL .....	15
3.2.2 React .....	15
3.2.3 Material UI & Recharts .....	15
3.2.4 Redux .....	15
3.2.5 Google maps API.....	17
3.2.6 Spring Boot.....	17
3.2.7 Version Control .....	17
3.2.8 OpenAI (ChatGPT, GPT-4).....	18
3.2.9 Postman .....	18
3.2.10 Microsoft Authentication Library .....	18
3.2.11 Deployment tools .....	19
3.3 Collaboration .....	19
3.3.1 Jira .....	19
3.3.2 Confluence.....	19
3.4 Development .....	20
3.4.1 Agile Methods.....	20
3.4.2 Minimal Viable Product & User tests.....	20
3.4.3 Connecting to Azure Database .....	21
3.4.4 Spring Boot.....	22
3.4.5 Visualizing data .....	22
3.4.6 Improving availability .....	24
3.4.7 Authenticated access .....	25
3.4.8 Postman .....	25
3.4.9 Version control .....	25
3.4.10 Collaboration .....	26
3.4.11 Processing and optimalization .....	26
<b>4 Results .....</b>	<b>29</b>
4.1 General.....	29
4.2 Engineering results .....	29
4.2.1 Map integration with polygons .....	30
4.2.2 Design principles.....	32
4.2.3 Authorizing accounts.....	34
4.2.4 Filtered search and date picker .....	35

4.2.5 API development.....	36
4.2.6 Assessing disease impact.....	40
4.2.7 Deployment .....	40
<b>4.3 Administrative results.....</b>	<b>41</b>
4.3.1 Using Jira & Confluence .....	41
4.3.2 Git.....	41
4.3.3 Milestones .....	42
4.3.4 Quality insurance .....	42
<b>4.4 Performance and Optimization Results.....</b>	<b>43</b>
4.4.1 Application performance .....	43
4.4.2 Code optimization .....	43
4.4.3 Accessibility and usability .....	43
4.4.4 Code reliability.....	43
<b>4.5 Integration and Compatibility Results .....</b>	<b>45</b>
4.5.1 Integration with external systems .....	45
4.5.2 Cross-platform and browser compatibility .....	46
<b>4.6 Learning Outcomes and Skill Development .....</b>	<b>46</b>
<b>5 Discussion.....</b>	<b>47</b>
5.1 Choice of technologies .....	47
5.2 Team collaboration and communication .....	47
5.2.1 Version control .....	47
5.2.2 Testing.....	48
5.3 Time constraints and milestones.....	48
5.4 Quality assurance .....	48
5.5 Alternative approaches .....	49
5.6 Agile Methods .....	49
5.6.1 Sprints .....	49
5.6.2 Daily standups.....	49
5.6.3 Jira .....	50
5.6.4 Confluence.....	50
5.6.5 Client and supervisor .....	50
5.7 Fulfillment of Client Requirements.....	51
5.7.1 Confidentiality .....	51
5.7.2 User Authentication .....	51
5.7.3 Data visualization.....	51
5.7.4 Graphical interface .....	52
<b>6 Conclusion.....</b>	<b>52</b>
6.1 Summary of Achievements .....	52
6.1.1 Technology Selection .....	52
6.1.2 Team Collaboration and Project Management .....	53
6.1.3 Application Design and Functionality .....	53
6.1.4 Overcoming Time Constraints.....	53



6.1.5 Quality Assurance .....	53
6.2 <i>Limitations and Future Research</i> .....	54
6.2.1 Alternative Approaches .....	54
6.2.2 Additional Features and Improvements .....	54
6.3 <i>Identified problems</i> .....	54
6.3.1 Spring Boot in deployment.....	54
6.4 <i>Final Remarks</i> .....	55
<b>7 Societal Impact</b> .....	<b>56</b>
7.1 <i>Benefits for Aquaculture Industry</i> .....	56
7.1.1 Economic implications .....	56
7.2 <i>Environmental Impact</i> .....	56
7.3 <i>Public health and food industry</i> .....	57
7.4 <i>Future Research and Development</i> .....	57
<b>Bibliography</b> .....	<b>58</b>

## List of code examples

CODE 1 (GOOGLE MAPS CONTAINER COMPONENT) .....	30
CODE 2 (CODE FOR EXTRACTING LATLNG-COORDINATES) .....	31
CODE 3 (IMPLEMENTATION OF THE GENERATEPATHS-FUNCTION) .....	31
CODE 4 (CUSTOM LOGIN-SCREEN COMPONENT) .....	34
CODE 5 (CUSTOM LOGIN-SCREEN COMPONENT) .....	35
CODE 6 (EXAMPLE OF REPOSITORY INTERFACE) .....	37
CODE 7 (EXAMPLE OF CONTROLLER CLASS) .....	37
CODE 8 (EXAMPLE OF SERVICE CLASS) .....	37
CODE 9 (EXAMPLE OF A NATIVE QUERY USING JpaREPOSITORY) .....	38
CODE 10 (EXAMPLE OF THE SAMPLEDATA ENTITY CLASS BEING ASSOCIATED TO REPOSITORY INTERFACE) ..	38
CODE 11 (EXAMPLE OF A QUERY USING JDBCTEMPLATE) .....	39
CODE 12 (CI BUILD BACKED JOB - ALL PARAMETERS IN MVN CLEAN VERIFY NOT DEPICTED) .....	44
CODE 13 (BUILD FRONTEND JOB) .....	44
CODE 14 (CI AUTOMATED POSTMAN TEST) .....	45

## List of figures

FIGURE 1 (USE-CASE DIAGRAM) .....	13
FIGURE 2 (REDUX DEVTOOLS EXAMPLE, SHOWING THE DIFFERENT STATES MANAGED) .....	16
FIGURE 3 (HISTORICAL DATA GRAPH FROM "RECHARTS") .....	23
FIGURE 4 (GENERAL INFORMATION GRAPH) .....	23
FIGURE 5 (DROPDOWN HINT BUTTONS (ON HOVER)) .....	24
FIGURE 6 (DESCRIPTIVE TEXT ABOUT COLOR VALUES) .....	24
FIGURE 7 (NUMBER OF ROWS IN A QUERY) .....	27
FIGURE 8 (PERFORMANCE OF FINAL ENDPOINT) .....	28
FIGURE 9 (SIZE AND PERFORMANCE OF QUERY) .....	28
FIGURE 10 (EXPLANATORY PICTURE OF THE RESULT) .....	30
FIGURE 11 (RESULTING MAP WITH POLYGONS) .....	32
FIGURE 12 (ON CLICK/ON HOVER POLYGON POPUP) .....	32
FIGURE 13 (SIDEWINDOW PC-SCREEN) .....	33
FIGURE 15 (SIDEWINDOW TABLET-SCREEN) .....	33
FIGURE 14 (SIDEWINDOW MOBILE-SCREEN) .....	33
FIGURE 16 (AGD-PERU 2022) .....	35
FIGURE 17 (AGD-PERU 2021) .....	35
FIGURE 18(PACKAGE/CLASS HIERARCHY, INCLUDING EXAMPLE OF ENTITY CLASS) .....	36
FIGURE 19 (A SINGLE REQUEST TAKING OVER 5.6 SECONDS) .....	37
FIGURE 20 (JDBC, UPDATED NTNU-VIEW FILE-STRUCTURE) .....	39
FIGURE 21 (JPA, NTNU-VIEW FILE-STRUCTURE) .....	39
FIGURE 22 (INITIAL FILE-STRUCTURE, SERVICE AND CONTROLLER CLASSES NOT FULLY IMPLEMENTED) .....	39
FIGURE 23 (THE URL OF THE WEBSITE) .....	40
FIGURE 24 (EXAMPLE OF COMMUNICATIONS WITH CLIENT USING DISCORD) .....	50



## Before reading

This report includes technical language and acronyms which can be hard to understand. Be sure to glance at the following and keep them in your mind while reading.

## Glossary and Phrases

**Frontend:** The part of a website that the user directly interacts with, among the likes of styles, layouts, buttons, forms, etc.

**Backend:** The part of a website that runs behind the scenes, which handles data, requests, database management, etc.

**Framework:** A framework is a reusable software structure that provides a foundation for developing applications, simplifying the development process by offering pre-built functionality and enforcing best practices.

**Library:** A library is a collection of pre-written code, functions, or routines that developers can use to avoid writing repetitive or commonly used code, saving time, and improving code quality.

**Web-based:** Web-based applications or systems run in a web browser and are accessible through the internet, allowing users to access and interact with them from any device with an internet connection.

**Cloud-based:** Cloud-based solutions are hosted on remote servers and delivered over the internet, providing scalability, flexibility, and accessibility from any location, as opposed to being hosted on local servers or machines.

**Wireframe:** A wireframe is a visual representation or blueprint of a web page or application's layout, used to plan and organize the user interface elements, content, and overall structure before development.

**Development:** Development refers to the process of designing, creating, and testing software applications or systems, involving various stages such as planning, coding, debugging, and deployment.

**Version control:** Version control is a system that tracks and manages changes to source code or other files in a project, allowing developers to work on different versions, merge changes, and revert to previous states if needed.

**Agile:** Agile is a project management and product development approach that emphasizes flexibility, collaboration, and iterative progress, allowing teams to quickly adapt to changing requirements and deliver incremental value.

**Hooks:** React Hooks are functions that let you "hook into" React state and lifecycle features from functional components, allowing you to manage state and side effects without needing to write a class component.

**Devtools:** Short for developer tools, devtools are an essential tool for web developers, providing a suite of features that assist with debugging and optimizing web pages and applications.

**Agens:** In physics, chemistry, and medicine, the term "agens" is used to refer to a substance or material that causes a biological, chemical, or physical change (effect).

**View:** A database view is a virtual table created by combining data from one or more tables in a database. It does not store any data itself but retrieves data dynamically from the underlying tables. A database view can simplify data access and management by providing a single, consistent, and secure interface to a subset of the data in the database.

**Query:** Queries are requests for specific information or data from a database or system. They are used to retrieve, manipulate, and analyze data by specifying conditions and parameters. Queries enable users to search, filter, and extract relevant information based on their needs and criteria.

**Stack trace:** A stack trace is a report of the active function calls at a specific point in a program's execution. It provides a detailed list of function calls that led to the point where the program crashed or encountered an error. A stack trace can be a useful tool for debugging and identifying the root cause of errors in software applications.

**JpaRepository:** An interface in Spring Data JPA that provides methods for all the CRUD (Create, Read, Update, Delete) operations on the entity for which it is created. It simplifies the data access layer by eliminating boilerplate code, and it can be extended to create custom queries and methods for interacting with the database.

**Java Database Connectivity:** Spring Boot JDBC is a part of the Spring Framework that simplifies database interaction in Java applications, providing an abstraction over Java Database Connectivity (JDBC). It offers a `JdbcTemplate` class that automates common tasks such as connection management, statement preparation, result set handling, and exception handling, enabling developers to focus more on business logic rather than low-level database operations.

**TCP/IP:** TCP/IP stands for Transmission Control Protocol/Internet Protocol. It is a set of protocols that allows computers and other devices to communicate and share information over networks, such as the Internet.

**HTTP/HTTPS:** HTTP, short for Hypertext Transfer Protocol, is a protocol used for transferring hypertext requests and information between servers and browsers, forming the basis of any data exchange on the Web. HTTPS is an extension of HTTP where the data transferred is encrypted.

**FTP:** FTP, short for File Transfer Protocol, is a commonly utilized network protocol that facilitates the transmission of computer files from one system to another via a network built on the TCP protocol, like the internet.

**WebSocket:** WebSockets are a communication protocol that enables real-time, bidirectional data transfer between a client (such as a web browser) and a server.

**Prop Drilling:** Prop drilling refers to the process in React where props are passed from a parent component down through the component tree to child components, which can become complex and unwieldy in large applications.

**Mock testing:** Mock testing is a methodology in software testing where individual units of source code are tested in isolation using simulated responses (mock objects) in place of actual dependencies to verify the functionality and behavior of the unit.

## Acronyms and jargons

API.....	iii; iv; 9; 14; 17; 18; 22; 31; 36; 46; 48; 53
CSS.....	10; 20; 21
DTO.....	39
HTML.....	10; 20; 21
JS .....	47; 48; 50; 53; 55
MVP.....	13; 20; 21

API: Application Programming Interface

CSS: Cascading Style Sheets

DTO: Data Transfer Object

HTML: Hypertext Markup Language

JS: JavaScript

MVP: Minimum Viable Product

# 1 Introduction

## 1.1 Background

In recent years, fish farming has grown rapidly, making it more important than ever to monitor and manage fish health effectively. Good fish health is essential for the success of fish farms, as diseases can cause serious problems for both the fish and the overall business. Reportedly, over 58 million salmon were killed in Norwegian farms alone, mainly due to lice and other diseases in 2022 [1]. Having accurate and timely information about fish health and different diseases helps fish farm managers make better decisions about how to run their farms.

PatoGen is looking for a way to keep their customers updated on the status of fish farms in Norway. PatoGen is responsible for testing the fish in these fish farms for diseases. Currently, they do not have an efficient way to convey this information to their customers and have concluded that a web-based application would be suitable for this.

It is to be expected that the customers of PatoGen have various levels of competence in terms of using digital solutions, therefore the application should be simple to use, and not require any digital prerequisites.

## 1.2 Problem

Currently, all the information regarding the process of testing fish-samples from a customer is stored in a relational database controlled by PatoGen. These fish-samples are sent by mail, which PatoGen calls "orders". An order has numerous properties to it, among the likes of; the disease, if the test was positive, where the test was taken, and so on.

The problem consists of displaying all this information in a graphical interface, in a simple, comprehensive way that people of all digital competencies can understand. This requires well thought out solutions, and a close, continuous dialogue with PatoGen.

The solution shall strive to be scalable, secure, optimized for speed, be well documented, and use the principles of high cohesion and loose coupling.



## 1.3 Requirements

### 1.3.1 Confidentiality

All the data provided by PatoGen contains sensitive and confidential data about their customers. In a theoretical use-case, the data should be retrieved and displayed in such a way that a customer cannot access information about another customer.

The group had to sign a confidentiality agreement and the team received several specific demands for the application to maintain the confidentiality. One of these demands was that there should be data available from at least three customers at the same time before any data is displayed. This demand should be implemented in such a way that if a user changes the application's settings (what month to see data from, what diseases to analyze, etc.), it should still be at least three customers before any data is displayed.

If this demand was not met, the following problem could occur: if two customers were in a zone with no other competitors, and they both submitted tests the same month, a customer could analyze the data and subtract their own results. What would remain is the data of the competitor. This way one competitor can effectively gather information about the competitor's progress.

### 1.3.2 User authentication

As previously stated, a lot of the client's data is confidential. As an extra security measure, they want to authenticate users before they gain access to the website. The client has already registered their customers in a list in a cloud-system and implemented authentication towards that list in their other websites. They wanted us to implement a similar design for the authentication in our project.

### 1.3.3 Data visualization

The client had specific requirements regarding the visualization of their data. They wanted their users to be provided with an effortless way to compare data from different production areas and to filter data with specific search parameters, all in a simple and intuitive interface. Even though the data provided was well known to PatoGen, they did not have any clear intentions on how to visualize them. The development team discussed and figured out visual solutions on their own and lastly insured these with the client.

### 1.3.4 Graphical interface

PatoGen gave the group full freedom to design the initial graphical interface for the web-application, but they provided a sheet with fonts and colors they wanted implemented. These colors and fonts are already used in other technology-services offered by PatoGen, so implementing their fonts and colors will help strengthen their brand and make the application more recognizable for the users.

After the initial design, there was still room for creative freedom, but the group also received a lot of feedback and guidance along the way.

## 1.4 Traffic Light Labeling

The Norwegian Ministry of Industry and Fisheries has decided to incorporate a traffic light system to determine the potential risk of diseases for salmon and trout production along the Norwegian coastline. This is a tactical measure that could result in a growth in production of 23 000 tons [2]. However, their system only analyzes the data from fish-lice.

Companies want to maximize their profits. If their data is analyzed correctly, the farms can maximize their production without having critical diseases spread to the point where they must eradicate all the fish. PatoGen does tests for about 50 diseases, which are also important to take into consideration when the fish farmers are making administrative decisions about their production. Therefore, PatoGen and their customers have a need for analyzing and displaying data about the status of different diseases, and not just fish-lice.

## 2 Theory

### 2.1 Client-server communication

Client-server communication is a fundamental concept in modern computing. It refers to the exchange of data between two entities: a client and a server. The client is a device or application that requests data or services from the server, while the server is a device or application that provides data or services to the client. The communication between the client and the server occurs over a network, which can be either a local network or the Internet.

This communication can be implemented using various protocols and technologies. Some common protocols include TCP/IP, HTTP, FTP, and WebSocket. These protocols define the rules for how data is exchanged between the client and the server. Different protocols have different strengths and weaknesses in terms of performance, security, and reliability, and the choice of protocol depends on the specific requirements of the application [3].

### 2.2 Relational database

A relational database is used as a store of information. It uses tables with columns and rows to store the data, whereas each row represents a unique record, and each column represents the attribute of the column. Each table also has a primary key for each row used as a unique identifier for each object in a table. This primary key can then be used on different tables as a foreign key, which will relate data from one table to another. This provides the ability to create connections between tables and allows for data manipulation [4].

### 2.3 Agile development

The agile development methodology is a strategy for software creation which prioritizes adaptability, teamwork, and the fulfillment of customer needs. It is based on iterative and incremental progress. Agile methodologies encourage adaptive planning, evolutionary development, continuous improvement, with a focus on delivering high-quality software that meets the needs of the end-users. This approach makes the development process easily adaptive to unexpected changes [5].

## 2.4 Coupling and cohesion

Coupling refers to the degree of interdependence between software modules or components. When code is highly coupled, a change in one module may negatively impact other modules, leading to difficulties in maintaining the software, whether it is for extending functionality or making modifications. As a result, it is essential to strive for "loose coupling" to ensure a more maintainable, flexible, and scalable system [6]. Cohesion refers to the relationship between the responsibilities or functionalities within a software module. High cohesion means that the module is focused on a single task, while low cohesion indicates that the responsibilities within the module are unrelated and serve different purposes. In general, modules should strive to adhere to the Single Responsibility Principle (SRP) and serve a single purpose. High cohesion is highly desirable, as it makes the module easier to understand, maintain, and modify [6].

Cohesion refers to the relationship between the responsibilities or functionalities within a software module. High cohesion means that the module is focused on a single task, while low cohesion indicates that the responsibilities within the module are unrelated and serve different purposes. In general, modules should strive to adhere to the Single Responsibility Principle (SRP) and serve a single purpose. High cohesion is highly desirable, as it makes the module easier to understand, maintain, and modify [6].

## 2.5 Testing

Testing is a critical process in the development of scientific applications, ensuring their accuracy, reliability, and overall quality. A comprehensive testing strategy helps identify potential issues early in the development process, reducing the likelihood of errors causing larger problems or jeopardizing the project's success [7].

### 2.5.1 Unit Tests

A unit test is a type of software testing that focuses on verifying the functionality of individual components or modules within an application. By isolating these components, unit tests can effectively detect errors, inconsistencies, or unintended behaviors. This approach allows for the prompt resolution of issues and minimizes the risk of bugs propagating to other parts of the system [8].

### 2.5.2 API Testing

API tests, also known as integration tests, are designed to verify the correct functioning of interfaces between different components or systems. In the context of scientific applications, these tests ensure that data is accurately transferred and processed between various

components, such as databases, external APIs, and user interfaces. This type of testing helps maintain data accuracy and consistency across the entire application, contributing to its reliability and stability.

## 2.6 Continuous integration

Continuous integration is a practice within agile methodology that serves as a safety measure when working with multiple developers. Often used alongside version control, continuous integration comes into play when a developer merges their code with a shared codebase. To ensure that the newly implemented changes do not cause issues, continuous integration performs checks on the codebase with the new code, verifying its compatibility. In cases where the changes fail these checks, the standard practice is to reject the changes and notify the developer to resolve the issues [9].

## 2.7 Client-side processing

Client-side processing refers to processing data and logic on the user's device, typically using programming languages like JavaScript, HTML, and CSS. This approach can provide a faster and more responsive user experience, as the processing is done locally on the user's device. Client-side processing has several advantages, including faster response times, reduced server load, and offline functionality. However, it also has several disadvantages, such as security risks, limited resources on the user's device, and browser compatibility issues [10].

## 2.8 Server-side processing

Server-side processing refers to the processing of data and requests on the server side of a client-server architecture. In this approach, the application's code and data are stored on the server, and the server processes the data and responds to client requests. Server-side processing can involve a wide range of tasks, including database queries, calculations, and application logic. This approach is often used in web development and other client-server applications, where the server handles the heavy lifting of processing data and serving up content to the client [10].

Server-side processing has several advantages, including better security, greater scalability, and access to more resources on the server. However, it also has several disadvantages, such as slower response times due to communication with the server, higher server load, and limited offline functionality. The choice between client-side and server-side processing depends on the specific requirements of the application, including factors such as performance, security, and scalability. Careful consideration of these factors is necessary to determine which approach is best suited for a particular application.

## 2.9 Polymerase Chain Reaction

PCR (Polymerase Chain Reaction) is a laboratory technique used to amplify specific segments of DNA or RNA, which is the main technique used by PatoGen to analyze their tests. It is a sensitive and efficient method that allows for the amplification of a single or a few copies of a target DNA or RNA sequence to millions or billions of copies. PCR has revolutionized many fields of biological research and has a wide range of applications, including disease diagnosis, genetic analysis, forensics, and biotechnology [11].

## 2.10 Cycle Threshold Values

CT-Values (Cycle Threshold Values) are a measurement of the amount of PCR cycles required for a sample to produce a detectable signal in a real-time PCR assay. In other words, it is a measure of the amount of amplification required to detect a target sequence in a sample [12].

CT-Values are often used to quantify the amount of a target sequence in a sample, with lower CT-Values indicating higher levels of the target sequence. They are also used to determine whether a sample is positive or negative for a particular target sequence, with CT-Values below a certain threshold indicating a positive result. While analyzing test results for PatoGen, CT-Values below 36.9 are treated as a positive.

## 2.11 Human-Computer Interaction

Human-computer interaction (HCI) is a study aimed at optimizing the way humans interact with computers by designing interactive user interfaces that makes communication between humans and computers easier, while still satisfying the user and their needs [13].

## 3 Methodology and tools

### 3.1 Planning phase

During the first weeks of 2023 the group started to evaluate the project and had multiple meetings with both client and supervisor to determine what was needed of the group to be able to deliver a successful bachelor thesis. This includes everything from working hours to regular meetings and updates. Working in sprints and setting clear goals for every sprint also states an important requirement which would be followed strictly during the coming months to guarantee progress. Having multiple meetings with the client helped with the overall understanding of their desires and expectations. After the client showed some examples of similar applications, the group was able to discuss and sketch up some possible solutions which the client considered very doable.

#### 3.1.1 Pre-project plan

After some planning the group wrote everything down in a Pre-project plan. This document contains the instructions for completing this project, including responsibilities, constraints, milestones, and various guidelines for preventing harmful situations. As deadlines and milestones were becoming clearer, a roadmap/Gantt-chart was created for visualizing the important sections and dates during the development process.

#### 3.1.2 Use cases

Creating use-cases and gathering user stories from user-tests can help a team reach a better understanding of the task at hand. The group needed to understand the workflow and created a simple use-case diagram visualizing the probable situations when using the intended application. Also listening to the needs of users during user-tests helps tremendously.

### 3.1.2.1 Use case diagram

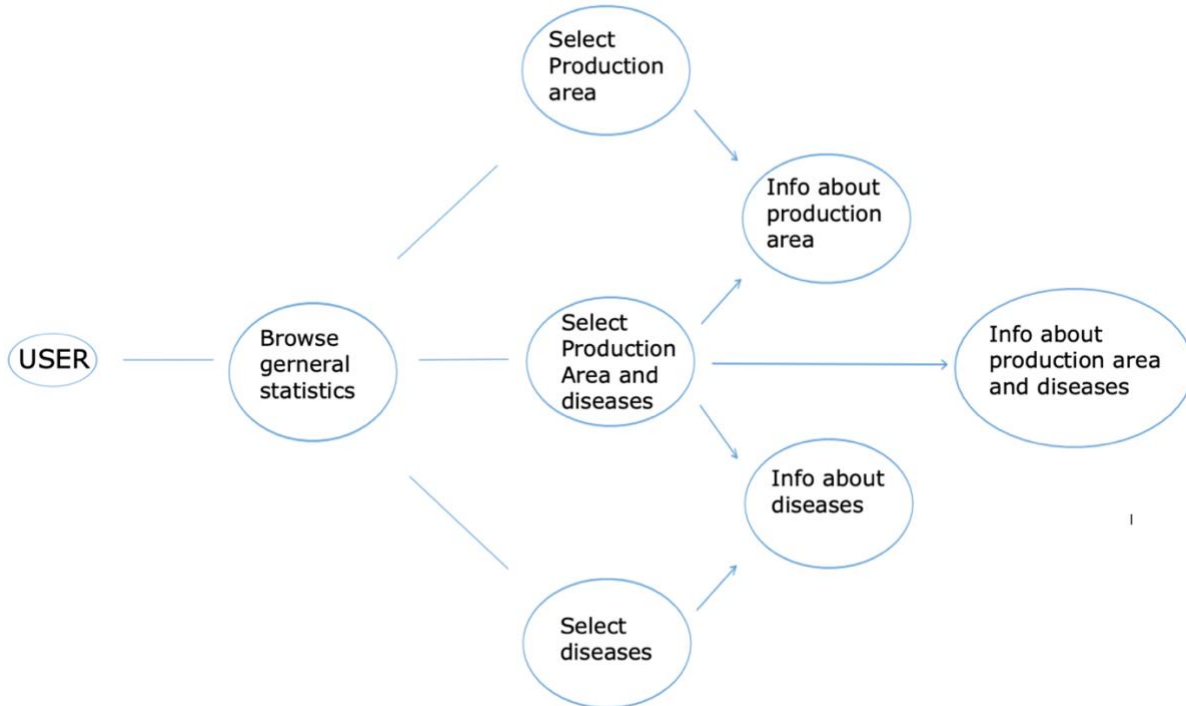


Figure 1 (Use-case diagram)

### 3.1.2.2 User stories

User stories created based on the wants and needs of users during tests, which the team has conducted along the way. (Stories 5 and 6 are not from the initial user-tests with wireframe, but MVP)

User story 1: "As a user I want to navigate the different zones in the map."

User story 2: "As a user I want to see the status of diseases in my zone."

User story 3: "As a user I want to be able to see historic data and choose specific dates."

User story 4: "As a user I want to choose a specific production zone and see the relevant data."

User story 5: "As a user I want to compare diseases to each other".

User story 6: "As a user I want the map to zoom in on my location, when I give access to my position".



### 3.1.3 Wireframes & Design

Deciding on a final design is an important milestone in every project, after thoroughly discussing potential application designs, the group managed to agree on a solution. The group used Figma, a cloud-based design tool great for collaboration [14]. Using this tool allowed the group to simultaneously work on the same wireframe sketches and see alterations in real time. Having the ability to work side by side while creating the application layout helped create a clean and useful webapp.

The applications usage and design are tailored to fit the needs of fish health experts, primarily PatoGen, and their customers will have the accessibility to review detailed information about the different production areas.

### 3.1.4 Research

Before starting a development process, having the right tools is exceptionally important. Initially PatoGen wrote in the project description that they would like the application to be written in Angular, a development platform, built on TypeScript. [15]. However, as the team had previous experience using the JavaScript library React, the team asked permission to use this, as the development of the application would be more effective.

Further research included finding different APIs for map services. The group finally decided on using the Google Maps API, as this is one of the most used map API's, which is highly documented and contains a lot of easy-to-use functions.

It was found that as a backend Java framework, Spring Boot delivers an easy integration and data funnel between PatoGen's database, and the frontend React application. In comparison Spring oot offers both a more developer-friendly experience and more well-known concepts/documentation, than many of the newer competitive solutions, like Micronaut, Quarkus or Javalin [16]. Previous experiences using Spring and their frameworks also made this decision easier as the group already knew the basics of these technologies.

Generally, the group had to perform individual research on the topic at hand to discover the true meaning and basic terminology within the fish farming industry. With significant help from the PatoGen representative Noralf, the group has managed to garner a decent understanding of the importance of keeping our fish healthy.

## 3.2 Tools

After carefully researching the possibilities, some tools and technologies stood out from the rest. The group's decisions are made primarily on quality and experience as the members all have previous experience with most of these.

### 3.2.1 SQL

SQL, commonly known as structured query language, is a programming language for manipulating data in relational databases. The main ability of the language is to create, read, update, and delete data. This is the language used to communicate with the database of the project. Not only used by database administrators, but also developers and analysts for writing integrated scripts or generally write analytic queries [17].

### 3.2.2 React

For the frontend portion of this project, the group decided on using React, a JavaScript library for building user interfaces. React lets you create individual components and combine them to easily create complex applications [18]. Some components, however, are already made. External libraries containing can easily be integrated within any application including. These libraries make it possible for the developer to focus on more important tasks, instead of spending time on creating components that already exist in an external library.

### 3.2.3 Material UI & Recharts

Material UI is a library of tailored React components, ranging from charts to buttons to lists, Material UI offers a wide variety of components that seamlessly integrate in most react projects. The purpose of Material UI is to reduce the time developers use to create reusable components. "MUI offers a comprehensive suite of UI tools to help you ship new features faster" [19].

Recharts, similarly to Material UI, is an external library of React components. Recharts has an extensive collection of reusable react components that visualize and graph data. "Recharts is a composable charting library built on React components." [20]. For visualizing complex data in scalable charts, the group found that recharts was a great tool.

### 3.2.4 Redux

Redux is a prominent JavaScript library that primarily serves as a state management tool. It provides a central repository where diverse states can be stored and accessed from any point in your project [21]. By ensuring data and state synchronicity throughout the application, Redux significantly contributes to a well-maintained system. In this particular project, Redux plays a crucial role in preventing excessive prop drilling and maintaining a tidy component structure. As it allows for global access to information within the project, data and states

referenced across various components are readily available, contributing to a more efficient and cleaner codebase.

Redux also has its own chrome developer tools extension. This is an extremely useful feature for development, as it offers the ability to access information about a redux state, like initialization, current value, method, etc. The redux developer tool is an elegant and effective tool compared to the alternatives, like manually logging variables and states to the console.

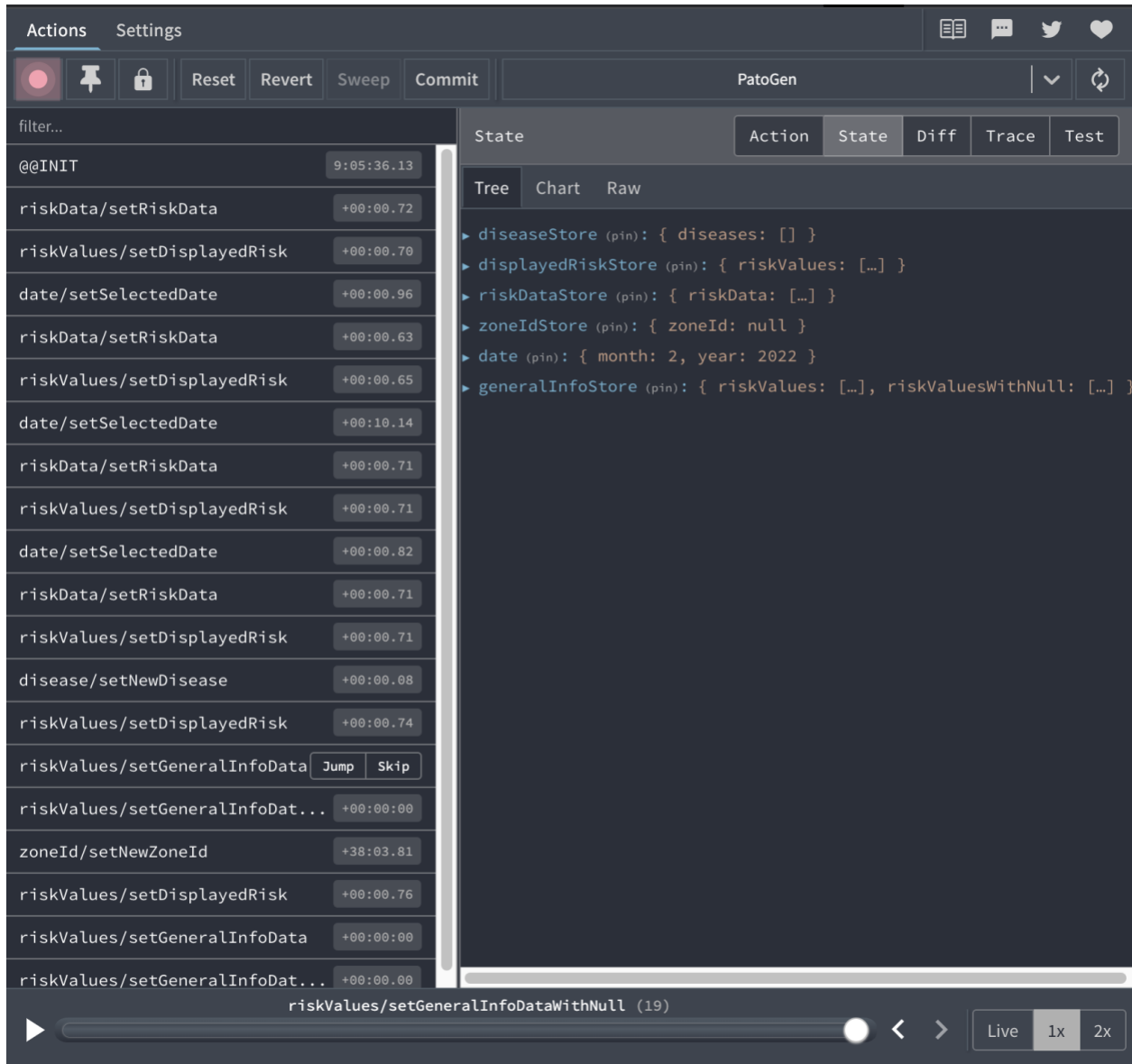


Figure 2 (Redux devtools example, showing the different states managed)

### 3.2.5 Google maps API

The Google Maps API allows developers to access the popular and widely recognized Google Maps. Google's data and functionality could therefore be incorporated in different private or corporate projects. The interactive map is highly customizable making it suitable for web, mobile or unique applications. By creating an API key, which is a specific code associated with the intended project, a developer is able to make calls to Google's API using this key. This way Google can regulate and monitor the usage of different applications.

### 3.2.6 Spring Boot

Spring Boot is a powerful Java framework designed to simplify backend development by providing pre-configured components and functionality, particularly for database interactions and web services. As a part of the larger Spring framework, Spring Boot enhances developer productivity through streamlined database connectivity, annotation-based syntax, dependency injection, and autoconfiguration. One of the most common use cases is the development of REST APIs, which serve as an abstraction layer for applications, enabling faster development cycles and more maintainable code [22] [23].

### 3.2.7 Version Control

Version control systems like Git are essential for managing changes in software development projects, allowing developers to track and collaborate on different code versions. Git, a popular distributed version control system, offers robust functionality and enables developers to work independently with local repositories. Its primary purpose is to provide safety and stability during development by saving snapshots of code, ensuring easy recovery from unexpected events, and minimizing the risk of losing critical work [24].

#### 3.2.7.1 GitHub

GitHub is the leading platform for hosting remote Git repositories, offering an intuitive graphical user interface that facilitates collaboration among multiple developers. It provides an extensive array of tools to streamline development processes, including support for agile methodologies, in-depth insights into project progress, comprehensive commit history, and more. As a versatile platform, GitHub empowers developers to work more efficiently and effectively in a collaborative environment [25].

### 3.2.7.2 GitHub actions

GitHub Actions is a powerful feature of GitHub that provides a robust platform for Continuous Integration (CI). By creating highly customizable workflow files, GitHub Actions enables the automation of various tasks that are triggered by events, such as Git pull requests. These workflows can execute user-defined scripts to perform a wide range of tasks, including building projects, running tests, and deploying applications. This seamless integration with the GitHub platform streamlines the development process and enhances collaboration among team members [26].

### 3.2.8 OpenAI (ChatGPT, GPT-4)

A brief time before the development of the thesis began, an AI (Artificial Intelligence) research and deployment company named Open AI [27] released a revolutionary large multimodal model known as GPT-4, which is a machine learning model capable of handling text and image inputs, and output text [28]. As GPT-4 is capable of handling code input, it offers the ability to aid in developing, debugging, and generally improving code quality and speed of development.

### 3.2.9 Postman

Postman is a highly popular platform for developing and testing APIs. Postman's popularity is related to the program's ease of use, simple interface, extensive functionality, debugging opportunities, and collection structure [29]. Postman can also be used for continuous integration, which is a good addition for the reliability of API endpoints.

### 3.2.10 Microsoft Authentication Library

The Microsoft Authentication Library (MSAL) is a set of client libraries and APIs that enable developers to integrate authentication and authorization capabilities in their applications. MSAL is designed to work with Azure Active Directory (Azure AD) and Microsoft accounts (MSAs), allowing users to sign in with their Microsoft identities and grant access to their resources and data. It also supports a wide range of programming languages and platforms, including .NET, Java, Python, iOS, Android, and JavaScript [30]. It also comes with predefined react hooks that make developing in react more effective.

### 3.2.11 Deployment tools

NTNU, in partnership with OpenStack, provides virtual machines equipped with substantial resources to cater to various project requirements. Each virtual machine is provisioned with two virtual CPU cores, 6GB of RAM, and 40GB of storage, along with the Ubuntu 22.04 operating system. These specifications effectively accommodate the team's objectives for employing the virtual machine, primarily as a server to host web applications and ensure their public availability on the internet.

Nginx is an open-source web server and reverse proxy server. It offers high performance, stability, is easy to configure and is resource effective. Nginx is used to serve web content to users and is often placed in front of other web servers to improve load handling and response times [31].

## 3.3 Collaboration

### 3.3.1 Jira

Jira is a project management and issue tracking tool developed by Atlassian, initially designed for software development, but now offers solutions for different industries and project types. It offers a wide range of features, including customizable issue types, custom workflows, support for Agile methodologies with Scrum and Kanban boards, reporting and dashboards, integrations with third-party tools, and robust access control and permission management.

### 3.3.2 Confluence

Confluence is a powerful collaboration and knowledge management tool developed by Atlassian, designed to help teams create, organize, and share content more efficiently. It serves as a centralized platform for storing and managing documents, meeting notes, project plans, and other essential information. Confluence supports rich-text editing, multimedia embedding, and offers a wide range of templates for various use cases. Key features include version control for tracking document changes, real-time collaboration with simultaneous editing, and integration with other Atlassian products such as Jira.

## 3.4 Development

### 3.4.1 Agile Methods

Agile methodologies were employed to enhance team collaboration and organization. This dynamic approach to project management emphasizes flexibility and continuous improvement, enabling teams to adapt rapidly to changes and challenges encountered throughout the project's life cycle.

#### 3.4.1.1 Sprints

Working in sprints is one of the key practices of agile development. A sprint is a working period for a project, commonly around 1-4 weeks, which is focused on completing a larger goal. Splitting the development process into sprints helps keep the goal in mind, and splits the tasks into smaller, more concise, easier to handle pieces.

The team agreed to work in sprints of 2 weeks, and average roughly 30-35 hours of work a week per member.

#### 3.4.1.2 Retrospectives

At the end of each sprint, the team reflects on what has been good in a sprint, and what could be improved. Reflecting on the process in the sprint is necessary to uncover flaws and make development more effective. An example of a retrospective can be found in the attached document "Project manual", section 4.

### 3.4.2 Minimal Viable Product & User tests

Working closely in relation to the wireframes, and with time to spare, the team managed to produce an MVP (Minimum Viable Product). Using mainly basic tools for web development including JavaScript (React), HTML and CSS, the web-based application now looked and felt like the envisioned wireframes previously created in Figma. Even though this is the minimum requirement for an application like this, the MVP made it easier to conduct more intuitive user testing and experimentation to find weak points within the application.

User tests were mainly done by the clients, but also by peers in the school computer lab. They were encouraged to provide general feedback and to attempt to break the application in any way they could. Bugs and errors were immediate, but the overall vision, look and feel was greeted nicely by everyone that tried it out. The team noted user stories and fixes in Jira as tasks. This made development smooth and easy as every member could grab a task and mark it as "in progress" to prevent team members from working on the same tasks simultaneously.

The most notable feedback provided by the peers at the school while testing was the suggestion to implement URL parameters to provide the users a way to copy a set of filters to another user. The other testers that were present and the group members agreed that this was an outstanding idea, so the development to implement this started the same day.

By closely collaborating with the wireframes and effectively managing their time, the team fortunately developed a Minimum Viable Product (MVP) in suitable time. The web-based application, built primarily using fundamental web development tools such as JavaScript (React), HTML, and CSS, achieved the desired look and functionality outlined in the initial Figma wireframes. While the MVP met the minimum requirements for such an application, its existence greatly facilitated intuitive user testing and experimentation, allowing for the identification of weak points within the system.

User testing primarily involved clients, but peers in the school computer lab were also encouraged to participate. Testers were invited to provide general feedback and to actively attempt to uncover any flaws or issues within the application. While some bugs and errors were identified, the overall design and user experience received positive feedback from all testers. To ensure efficient development, the team utilized Jira to document user stories and log fixes as individual tasks. This streamlined the development process, preventing team members from duplicating efforts by enabling them to assign and track tasks using the "in progress" status.

During testing by peers at the school, notable feedback emerged, suggesting the implementation of URL parameters to enable users to copy a set of filters and share them with others. This suggestion received unanimous agreement from the other testers and group members, leading the team to initiate development of this feature on the same day.

### 3.4.3 Connecting to Azure Database

To display relevant information, data is required. PatoGen provided the team with a large set of their own collected data for the development of the application. The group was given permission to access a direct copy of the main database which updates concurrently. PatoGen performs analysis on hundreds of samples every day and their databases are frequently updated. This means that an application should optimally retrieve updates in real time to the user.

The team members were given individual login credentials for connecting to the Azure database. Afterwards, the main focus was to become familiar and discuss relevant data fields with the client and the supervisor. The database had an unnecessary complex structure, which



meant that a lot of tables had to be “joined” to provide basic information about a customer order. Joining several tables for every request is computationally expensive [32], which means that the provided data structure resulted in inferior performance.

Therefore, the team suggested to PatoGen that a new custom “view” would increase the performance of the application. Considering the fact that the database was restricted to read only, the group provided the client with SQL-code for the view. This view contained all data that was required to finish developing the application and meet all the customers' and clients' needs.

### 3.4.4 Spring Boot

Spring Boot was chosen for its reliability and simplicity. Although one of the initial ideas was to send database queries directly from the frontend, this was quickly dismissed as it was agreed that this was not the best practice due to potential security risks during data transfers. Furthermore, handling database requests directly from the frontend doesn't scale well as the application grows in size, as managing numerous or complex requests can quickly become challenging. Another issue to consider is the potential for data from the database to arrive in undesired formats, necessitating additional parsing.

Using spring boot to retrieve the data from the database and manipulating it to a format that suits the use case of the frontend was deemed as a better solution. This way, spring boot offers a layer of abstraction between the database and frontend, whereas spring boot handles the technicality of the data and database handling and serves as an API for the frontend. It also provides much utility, like implementing user authentication, adding restrictions to the endpoints – who is allowed to access it, etc. Debugging opportunities are strong with plenty built in functionality and testing and verifying the data are all steps spring boot can provide.

### 3.4.5 Visualizing data

Being able to connect the correct values onto the map was a big step in successfully tying the project together, as the map represents the heart of this application and visually gives it a more finished look. When selecting the other visualization tools for this app the team depended on PatoGen and their employees/specialists to help. As the team only have a certain basic understanding in this industry, the ones that will use it know more about what is needed and appreciated in this kind of application. Multiple meetings were had, and it became more and more obvious that no one really knew what they wanted. It was up to the team to construct some charts or diagrams that best represented the data and rather receive feedback/input from PatoGen.

### 3.4.5.1 Charting libraries

The reasoning for choosing charting libraries like "Recharts" or "Material UI" was for quick and easy alteration between look and feel. For instance, data from an entire year can be represented in a single graph (as seen in the picture below). The graph displays the percentage of positive test results from a selected disease and displays a value between 0 and 100 percent for each month, for an entire year at a time. For instance, 8,7 percent of all tests for the disease "AGD-PERU" submitted in April of 2022 were positive.

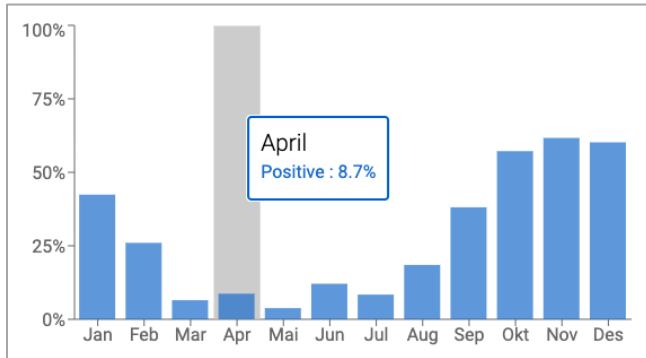


Figure 3 (Historical data graph from "Recharts")

This functionality was made specifically for the employees at PatoGen, as they would like to view trends of which diseases bloomed in different months and then compare them to previous years to identify patterns.

Given the size and complexity, showing larger graphs in the applications information window brought some challenges. For example, the width of the side window did not allow for the x-axis to display every field. It was therefore made changes to some of the graphs, which were flipped to show horizontally instead.

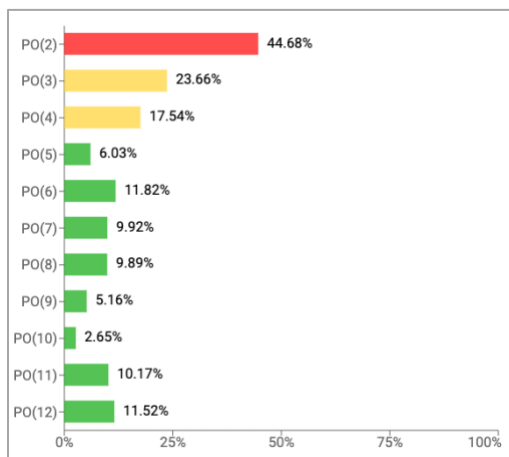


Figure 4 (General information graph)

### 3.4.6 Improving availability

During the development process the team conducted continuous user tests on peers and clients, resulting in a lot of constructive feedback. A particular functionality lacking was availability. Separate groups of users did not find it obvious what to do when handed over the application. As the team was working on this application every day and knew exactly how to navigate the app, small components like hint buttons and informative texts were forgotten. Keeping the application as clean and minimalistic as possible was always a priority, so integrating filler information and helpers was a difficult, but necessary measure to really extend the applications quality.

The group decided on implementing hint buttons that display text descriptions of the applications functionality. These buttons come with a highly recognizable question mark symbol which communicates to the user that they will contain information if some functionality is not understood. There are two of these buttons within the application. One for both dropdown menus, which is used to navigate and update most of the applications functionality.

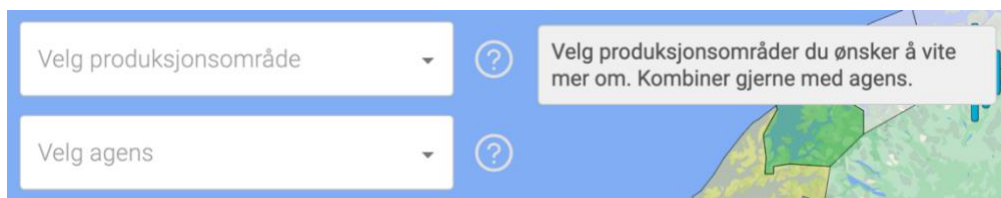


Figure 5 (dropdown hint buttons (on hover))

In Norwegian, this hint button states "Select a production area which you would like to know more about. Feel free to combine with diseases". To tell the user what possibilities there are to customize both the map and side window by choosing an area, a disease, both, or nothing at all.

#### Bakgrunn

Risikovurderingene baserer seg på forholdet mellom positive og negative tester fra de individuelle produksjonsområdene. Ved mindre enn 15% positive tester er farenivået grønt. Dersom 30% av testene er positive vil farenivået være rødt. Mellom disse er farenivået gult.

[Les om trafikklyssystemet på regjeringen.no](https://www.regjeringen.no)

Figure 6 (Descriptive text about color values)

Additionally, inside the side window, explanatory text and information is added to give the user some pinpointed information about what the different charts or icons represent. For example, what gives a production area green, yellow, or red risks, as seen in the picture above (the text translates to: "The risk assessments are based on the ratio of positive and negative tests from the individual production areas. If less than 15% of the tests are positive, the risk level is green. If 30% of the tests are positive, the risk level is red. Between these percentages, the risk level is yellow").

### 3.4.7 Authenticated access

Since this application uses a lot of confidential information, PatoGen requested that the solution should be behind a wall of authentication. As PatoGen uses Microsoft Azure Active Directory (Azure AD) for their current operations, a similar approach was desired in this application. Azure AD has premade login interfaces, which the user is redirected to when first entering the website. Only recognized login credentials will have access to the application after logging in. The list of authorized users is stored inside Azure AD.

### 3.4.8 Postman

Whenever a new endpoint has been established in spring boot and is ready for testing, Postman is used to send a request to the endpoint to verify that the data is correct. It is also great for debugging endpoints, as if they do not work correctly, tweaks can be made in spring boot and the endpoint can be tested again.

### 3.4.9 Version control

When working in a team it is crucial to implement version control for code safety and consistency. The team has used Git and GitHub for version control. GitHub is used to create a remote repository, where multiple collaborators can work together on a single code base.

Every time a member makes changes to the code, git tracks the changes locally. When a member then has developed code and is satisfied with the result, the member can commit these changes, saving them, then pushing these changes so they can be reached by the other team members. Given that the team has almost always worked together in person, only the main branch has been used during the project, since the team had a high degree of communication while developing.

## 3.4.10 Collaboration

Collaboration tools are crucial for creating an excellent product, therefore multiple tools with different purposes were used from the start of the project.

### 3.4.10.1 Jira

Jira was used to track ongoing issues for each sprint during development. Whenever an issue was discovered, it would be logged in jira, and added to the current sprint. From there on, the issue could be assigned a working member of the team. This way all members of the team know what each member is working on, making collaboration easier, preventing multiple members working on the same task.

Another function in Jira that was highly relevant was "Tempo". This is a highly user-friendly time logging functionality, where a working member can log time worked on an issue in the timesheet. This provides a comprehensive retrospective on what issues have been the most demanding, as well as seeing the time and effort each member has put down.

### 3.4.10.2 Confluence

Confluence has been used for administrative features, mostly writing meeting notes, sprint retrospectives, and taking notes during development on crucial information. This makes it easy to recall previously discussed themes and lowers the risk of misunderstandings.

### 3.4.10.3 Discord & Messenger

Discord is a chat application that allows voice, video, and text. The group has primarily used this application for loose continuous communication with the client which is also using Discord. This application makes it easy to update and share new functionality or requirements as well as receive help and feedback. Whereas Discord makes working in groups simpler, messenger allows for a push notification to update the team on upcoming meetings or other difficulties etc. Discord also allows for smaller file & code sharing, which has been handy on multiple occasions.

## 3.4.11 Processing and optimization

During the development of the application, some interesting questions arose regarding processing and optimization. The initial database-structure resulted in a lot of unnecessary joins, which in turn resulted in large tables with many fields as previously discussed in chapter

3.4.3 "Connecting to Azure Database". Joining tables is computationally demanding [32], and it resulted in severe performance issues. As a result, a new database view was created containing all relevant data.

However, even though the database view is simplified with as few columns as possible, there are still more than 800,000 rows of data. None of the queries in the backend request the entire view at once, but some of the queries need data for an entire year at a time, which can contain around 300,000 rows in the table (as shown in the picture below). These queries can contain up to around 24MB of data, which can result in some problems for the user. Considering this fact, the group had some interesting discussions about processing. Firstly, about whether the processing should be client-side or server-side. Secondly, what the data structure should be like when retrieving it from the database. Both client-side and server-side processing has some drawbacks, which are discussed in the following paragraphs.

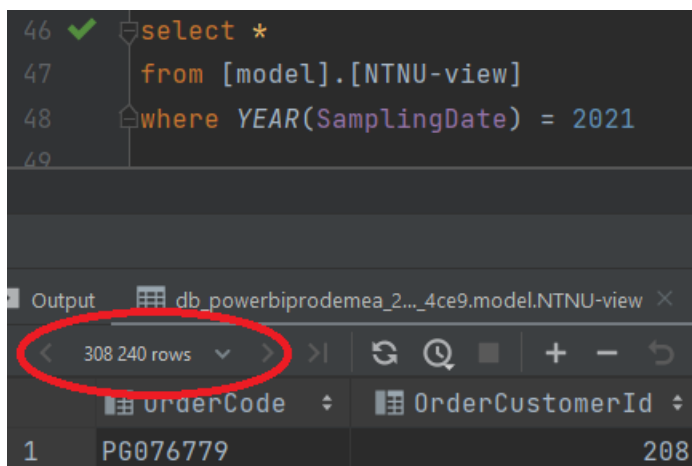


Figure 7 (number of rows in a query)

Even though most user devices can handle 24MB of data, there may still be a considerable number of users with older or less powerful devices. Processing a dataset of this size on the client-side may cause performance issues, consume a substantial portion of the device's memory, or even crash the application on less capable devices. Client-side processing relies on the user's device resources, which can vary greatly. While some devices might process the data quickly, others might take a considerable amount of time, leading to an inconsistent user experience. There are also some security risks associated with processing the data on the client-side. The data is more accessible to users and can be more easily tampered with, potentially compromising data integrity [33].

Lastly, and possibly the most crucial point in this discussion, is the fact that transferring 24MB of data from the server to the client may consume significant bandwidth and take a noticeable amount of time, especially for users on slower or less reliable internet connections. If the processing is done on the server-side, the server can do some calculations to reduce and simplify the data to fit in a smaller table. This table could potentially be reduced to a few bytes, which could drastically help the performance of the application.

A drawback to giving the server responsibility of processing tasks is that intensive or resource-demanding tasks can put a strain on the server, potentially leading to performance bottlenecks or decreased overall system performance. This issue is even more prominent

when the number of clients increases, as the server can become overwhelmed with processing requests. Scaling server-side processing can be complex and may require additional hardware, infrastructure, or distributed computing techniques to handle the increased load effectively [34].

Considering all these factors, the team eventually decided to do most of the data processing on the server side. The application is intended for a relatively small handful of customers, so the hardware needed to maintain server stability is fairly low. If the client eventually wants to use a cloud service for the deployment of the website, the cost of server scalability will not be too high, unless the target group of the website suddenly grows rapidly.

Below is a provided example of how this decision drastically improved the performance of the application. Instead of returning 12 451 rows of data directly to the client, the data is processed on the server and then only 12 rows of data are transferred (one row for each month, with a total size of 934 Bytes).

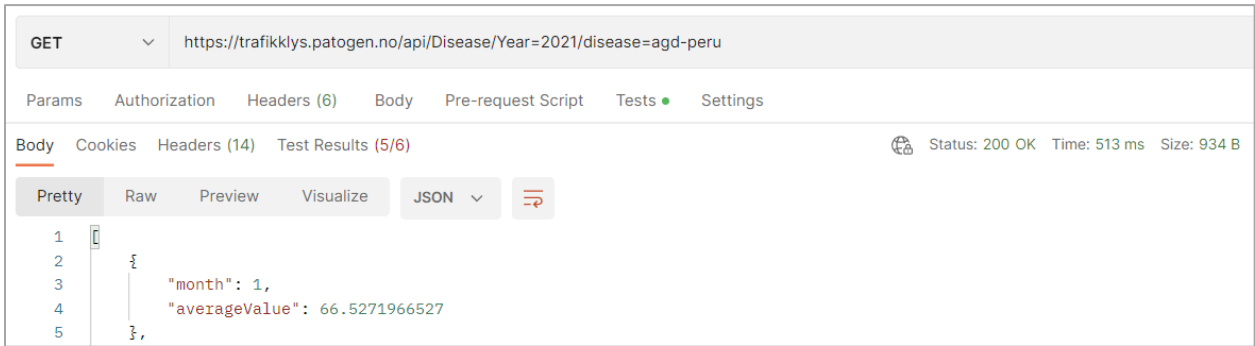


Figure 8 (performance of final endpoint)

As seen in the picture below, retrieving the complete list of 12,451 rows from the database takes around 1,4 seconds with a fast and stable network, and this data is unprocessed, which means that the *total* time for retrieving and processing data would be even higher than 1,4 seconds. In the currently implemented solution the total time is 513 milliseconds, which is more than three times as fast.

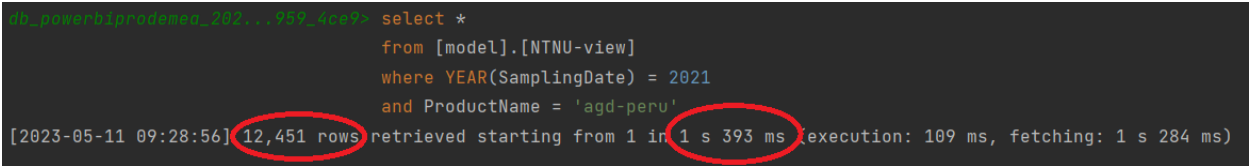


Figure 9 (size and performance of query)

## 4 Results

This section of the report focuses on the finished process. Here the different solutions within the application will be explained and discussed as the methodology and technologies are considered. Engineering results specify the actual application and the parts surrounding its development, whereas administrative results center the parallel processes included such as documentation and teamwork.

### 4.1 General

During this process, the group has been able to accomplish deployment of a fully functional application that meets all requirements from the client. The application is built using the JavaScript framework Reactjs, it contains a spring boot backend integrated with PatoGen's Azure database, it is deployed on a provided OpenStack server, and it is hosted to the web using nginx. Authorization is provided by Microsoft's Active Directory to keep control over future sensitive data. An in-depth description of the reasonings behind the selected technologies are found in "3.2 Tools". Due to familiarity and confidence with the tools, the team has been able to perform and deliver a product that meets all expectations.

### 4.2 Engineering results

The web-based application was intended to serve to simplify complex data and visualize PatoGen's most recent findings. As interpreted by the client PatoGen, showing customers the results of their analysis can be reduced and simplified, since the only people who could possibly understand the raw data are the ones generating it. A quote by a PatoGen employee and fish health expert has stuck with the team ever since meeting him; "KISS – keep it simple stupid". This quote has been a source of inspiration for the team through the entire process. The goal was that if you were to hand this application over to anyone, they would figure it out.





Figure 10 (Explanatory picture of the result)

The picture above highlights the main interactive components that this application offers. By altering the dropdown menus or date picker, the map and collapsible side window will automatically change appearance and adapt to the recent changes. This gives the user the ability to tailor the information displayed to match their purpose. The application is therefore considered to be highly dependent on computer and human interaction to reach full potential.

Additional interactive components include disease specific date selection inside the collapsible side window. Functionality explained in chapter 4.2.4 "Filtered search and date picker".

#### 4.2.1 Map integration with polygons

Using Google Maps as the applications integrated map proved to be effective as the amount of addons, documentation and customization available are unmatched. Drawing polygons, changing coloring, removing borders and cities made the application look simple and elegant.

```

<GoogleMap mapContainerClassName={"map"} center={center} zoom={locationFound ? 7 : 4.5} options={options}>
  <div>
    {polygons}
  </div>
</GoogleMap>

```

Code 1 (google maps container component)

Inside this component lies “polygons”, these are lines drawn onto the map using coordinates that represent each production area. The color of these polygons is determined by the percentage of positive tests for the specific production area. The team figured that in addition to the green, yellow, and red colored risks, another color should be provided if data is lacking. As a result, the team decided that polygons should display a grey color upon not having enough data available. For instance, this usually happens at the start of a month, when there is no new data submitted. The production zones will therefore be grey, indicating that there is not enough data available for the zone(s).

The polygons themselves were created by extracting a list of latitude and longitude coordinates from a JSON-file that the client provided. A line is then drawn between every coordinate-point from this list, which eventually becomes a polygon-shape. The implementation of this logic is shown in the code below. The returned list from this function is added as a prop (“paths”) to a Polygon-component that already exists in the Google Maps API (as shown in the next picture).

```
/**
 * Generates the polygon paths for a specific production zone
 * @returns a matrix of LatLng-objects
 * @param zoneId the id of a zone, starting from 0 in the south of Norway to 13
 */
function generatePaths(zoneId) {
  //finds data associated with the given zoneId
  const object = coordinates.find(obj => obj.properties.id === zoneId);
  const list = []
  //get the coordinates for the zone and return them in a list of lng-lat-objects
  if (object) {
    const coordinatesMatrix = object.coordinates;
    coordinatesMatrix.forEach((array) => {
      list.push({lng: array[0], lat: array[1]})
    })
  } else {
    console.error("The provided zoneId doesnt exist")
  }
  return list;
}
```

Code 2 (Code for extracting LatLng-coordinates)

```
<Polygon
  onMouseOver={onPolygonMouseOver}
  onMouseOut={onPolygonMouseOut}
  onClick={onClick}
  options={generateOptions()}
  paths={[[generatePaths(id)]]}
/>
```

Code 3 (implementation of the generatePaths-function)

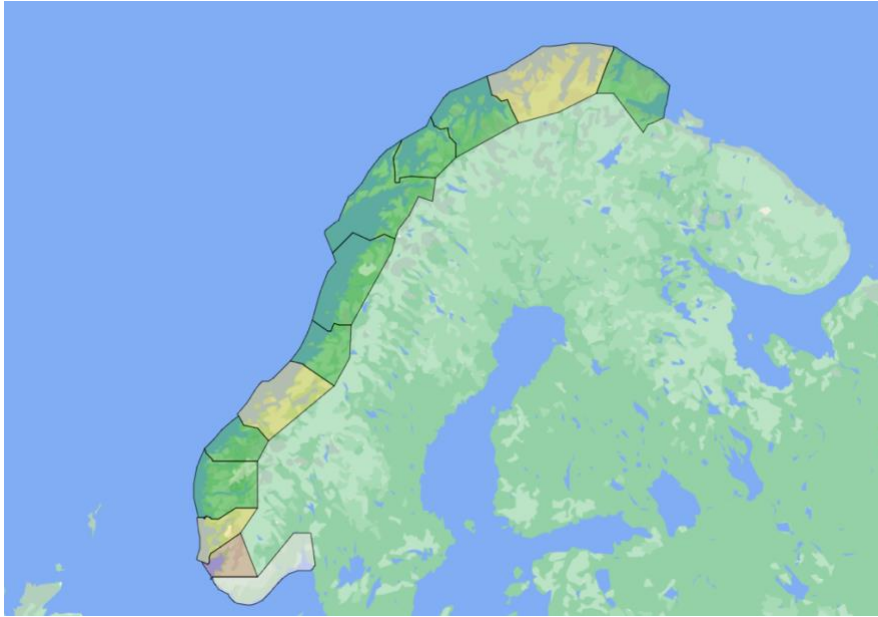


Figure 11 (Resulting map with polygons)

As shown in the picture above, for the first production area (PO1 – the Swedish border to Jæren, bottom of the picture) does not display any colors. The reasoning for this is that there is not enough data or not enough submitters. In this specific area there are only a few farms making it tricky to display data without exposing singular companies and such.

Additionally, the map provides another interactive method which allows the user to click polygons to select new production areas to review. A hover pop up message indicates the production area's name and risk assessment with a slight outline differential to show the user that this production area is selected.



Figure 12 (on click/on hover polygon popup)

## 4.2.2 Design principles

As previously mentioned, the group strived to follow the quote "KISS" or "keep it simple stupid". Considering the fact that technology isn't always well-understood by the general public, a simple UI design is desirable as it makes the process of using the application much easier.

As it stands, the application has strived to keep “unnecessary” functionality to a minimum, if any at all. The only parts users can alter the state of in this application are selected production area, disease(s), and date. The focused sections of this single-page application are that of visualizing the different customizations. The map covers the entire screen and side window displaying information covers about 40% horizontally when opened (50% horizontally on tablet / 60% vertically on mobile) for optimal viewing possibilities. Dropdown menus and timecard are free floating components that do not cover a large portion of the map.

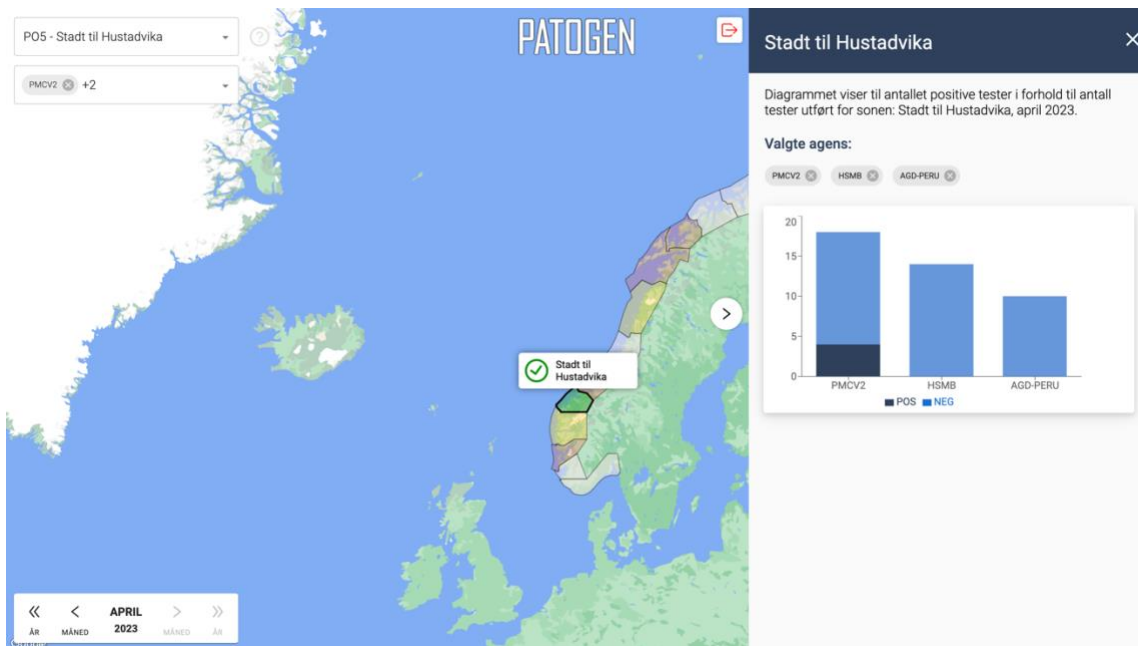


Figure 13 (sideWindow PC-screen)

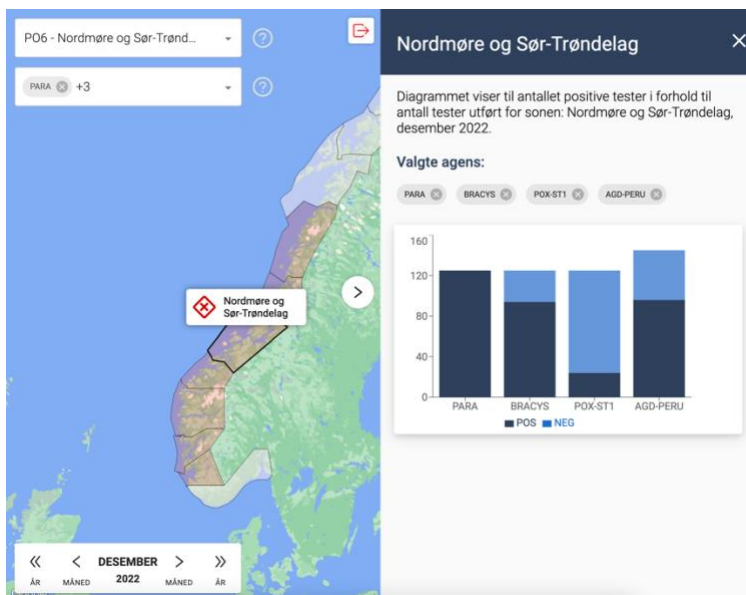


Figure 15 (sideWindow Tablet-screen)

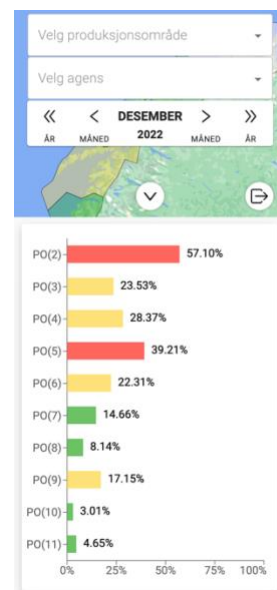


Figure 14 (sidewindow Mobile-screen)

### 4.2.3 Authorizing accounts

Authorizing users was not a priority during the development of the application. After some initial discussions with PatoGen, it was decided that authorization should be a stretch goal. They were more interested in seeing rapid development in the core functionality. One of the considerations in the discussion was that PatoGen had some very specific demands about confidentiality, as previously discussed in chapter 1.3.1 "Confidentiality", which undermined the importance of authorizing users. As previously described, users are not able to see any data from individual fish farms, and there must be at least three customers in a production area before the application displays anything to the user.

The group did eventually finish the application with enough time left to consider finishing the stretch goal, which PatoGen expressed that they would like the group to do. The client is also considering extending the scope of the application in the future to display data about individual fish farms to the individual customers, which renders authorization an absolute necessity.

For authentication, PatoGen suggested using Microsoft Azure Active Directory. Mainly because this technology was already familiar to them, and because their customers/users already have registered Microsoft accounts with PatoGen.

To implement the authorization, the group used the Microsoft Authentication Library (MSAL). The group configured a tenant in the Azure Portal and created a configuration file in the source code. The configuration file contains several variables and application properties, which are exported to other relevant functions like the predefined hooks from the MSAL-library.

The code provided below shows the custom react component which is implemented to redirect the user to a login-screen. This implementation is a high-level abstraction of a login screen. The abstraction ensures low coupling, because the code which is rendering the component does not know about the internal logic.

```
import {useMsal} from "@azure/msal-react";
import {loginRequest} from "./authConfig";

/**
 * Redirects the user to the login screen
 */
export function LoginScreen() {
  useMsal().instance.loginRedirect(loginRequest).catch(console.log);
}
```

Code 4 (Custom login-screen component)

In the code below, the previously mentioned abstraction is visible. The App-component does not know the internal logic of the LoginScreen-component, it only renders the component if

some parameters are true. The logic is as follows: the code inside the Unauthenticated Template-component is called if a user is not logged in, which is where the custom "LoginScreen"-component is rendered. Therefore, a user is redirected to a login screen if they are not logged in. However, if a user is logged in, the MainContent-component is rendered, which contains the core content of the entire application.

```
function App() {  
  return (  
    <BrowserRouter>  
      <AuthenticatedTemplate>  
        <MainContent/>  
        <UrlHandler />  
      </AuthenticatedTemplate>  
      <UnauthenticatedTemplate>  
        <LoginScreen/>  
      </UnauthenticatedTemplate>  
    </BrowserRouter>  
  );  
}  
export default App;
```

Code 5 (Custom login-screen component)

#### 4.2.4 Filtered search and date picker

Searching or selecting different options gives the user a whole new visualization. For example, someone working for a fish farming company selecting the production area they are currently located in and a disease that has been/could potentially be causing problems. This specific customization will display a graph showing the number of tests provided with the number of positives for the month selected. Graphs and customizations stay the same when going back and forth in time. It is therefore possible to find patterns of contagiousness, an example of this is selecting a disease without specific areas and inspecting yearly progressions.

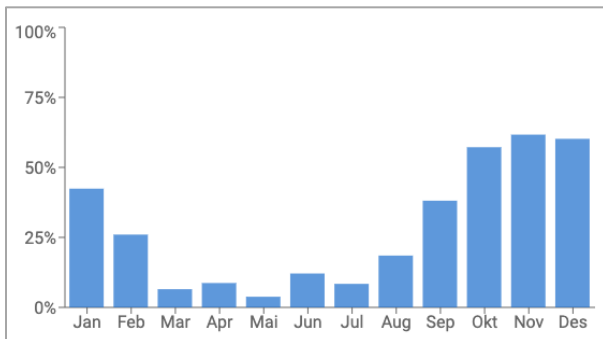


Figure 17 (AGD-PERU 2021)

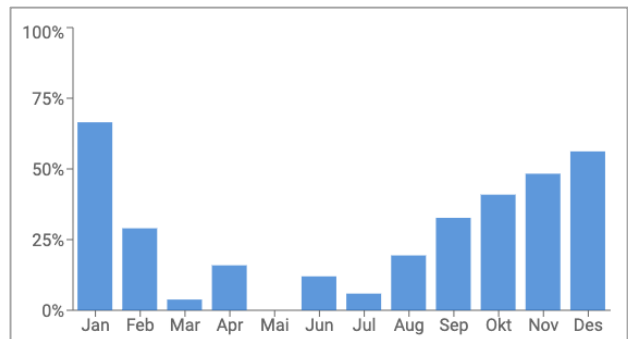


Figure 16 (AGD-PERU 2022)

As this was one of the most specific requests from employees at PatoGen, the team understood the importance of this functionality. Employees discuss and show each other latest



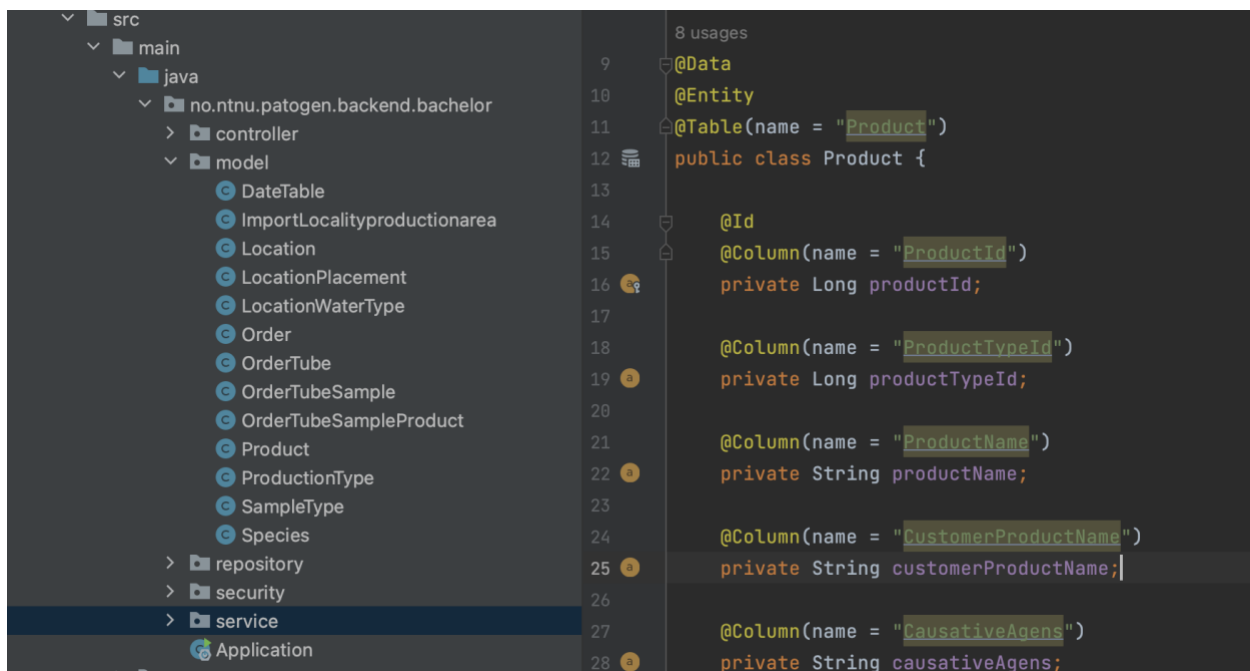
trends within the field. As shown in the previous diagrams, there are patterns for when a certain bacterium is active or unactive. AGD-PERU is a gill disease that targets fish and is caused by the amoeba "*Neoparamoeba per Urans*". This specific disease usually blooms when fall arrives and water temperature drops, and calms down again when the summer approaches and water temperature rise.

## 4.2.5 API implementation

One of the priorities the team had during the development was handling the data provided by PatoGen in a good manner. The Spring Boot serves as a backend, between the database provided by PatoGen, and the React frontend. This serves numerous advantages, as it scales well, makes changes easier, and in general is aligned with the principles of coupling and cohesion.

### 4.2.5.1 Initial approach

During early development, it was attempted to use standard spring boot syntax consisting of Entity, repository, service, and controller classes. Whereas entity represents an entity – or a row, in the table of a database, repository is used for retrieving data, service is used for manipulating the retrieved data, and controller for handling requests. This approach works well for retrieving data from a single table.



```
8 usages
9  @Data
10  @Entity
11  @Table(name = "Product")
12  public class Product {
13
14      @Id
15      @Column(name = "ProductId")
16      private Long productId;
17
18      @Column(name = "ProductTypeId")
19      private Long productTypeId;
20
21      @Column(name = "ProductName")
22      private String productName;
23
24      @Column(name = "CustomerProductName")
25      private String customerProductName;
26
27      @Column(name = "CausativeAgens")
28      private String causativeAgens;
```

The screenshot shows an IDE with a package/class hierarchy on the left and the code for the Product entity class on the right. The hierarchy includes a package 'no.ntnu.patogen.backend.bachelor' with sub-packages 'controller' and 'model'. The 'model' package contains several classes, including 'Product'. The 'Product' class is highlighted in the hierarchy. The code on the right shows the Product class with annotations for @Data, @Entity, and @Table, and fields for productId, productTypeId, productName, customerProductName, and causativeAgens.

Figure 18(package/class hierarchy, including example of Entity class)

```

2 usages
public interface ProductRepository extends JpaRepository<Product, Void>, JpaSpecificationExecutor<Product> {
    List<Product> findAllByProductId(Long id);
}

```

Code 6 (Example of repository interface)

```

import ...
2 usages
@Service
public class ProductService {

    1 usage
    @Autowired
    private ProductRepository productRepository;

    /**
     * Gets all the products from the database
     *
     * @return all the products in the database
     */
    1 usage
    public List<Product> getAllProducts() {
        return productRepository.findAll();
    }
}

```

Code 8 (Example of Service class)

```

import ...
/**
 * REST API controller serving endpoints for products
 */
@RestController
@CrossOrigin(origins = "*")
public class ProductController {

    1 usage
    @Autowired
    private ProductService productService;

    /**
     * Gets all products in the database
     *
     * @return List of all products
     */

    @GetMapping("/api/product/getAll")
    public List<Product> getAll() { return productService.getAllProducts(); }
}

```

Code 7 (Example of Controller class)

However, as mentioned in chapter 3.4.3 "Connection to database" and 3.4.12 "Processing and optimization", there was a necessity to make several joins to tables in order to get the data desired for usage in the frontend. In Spring Boot, this is a slightly complex process as it has to be done between entity classes. This also ended up causing unjustifiably long request times.

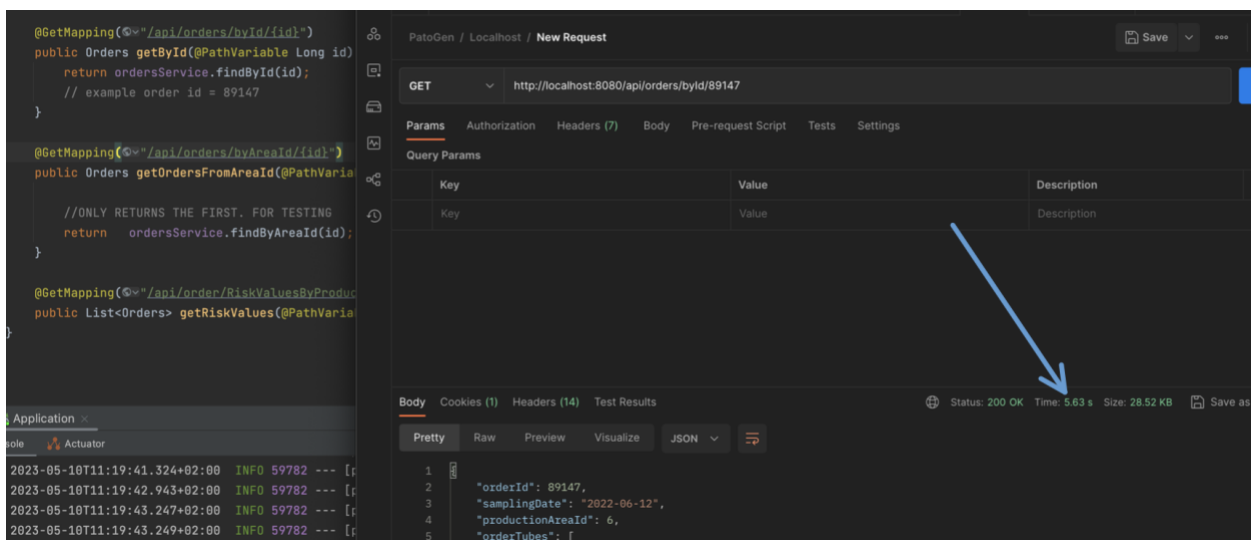


Figure 19 (A single request taking over 5.6 seconds)



## 4.2.5.2 JPA Repository

Assisting the client to create a single view with all the data, the team required drastically simplifying the process of requesting data, as the data was ready to be extracted without the usage of computationally expensive join operations.

Using JpaRepository, queries could be written in the repository classes for CRUD operations. Spring boot comes with a fleet of predefined queries based on a name syntax. Unfortunately, the predefined queries do not support more complex queries, JpaRepository also supports native(custom) queries. With the new view, native queries were used to retrieve data from the database.

```
/**
 * Selects all product area which are between 1 and 13, their associated amount of positive tests in a percentage, for a month in a year.
 * @param year the year to be searched
 * @param month the month of values desired
 * @return a list of product areas from 1 to 13 with their associated amount of positive tests.
 */
3 usages
@Query(
    value = "SELECT\n" +
        "    ProductionAreaBwId,\n" +
        "    (COUNT(CASE WHEN ResultText = 'POS' THEN 1 END) * 1.0) / NULLIF(COUNT(ResultText),0) AS pos_percentage\n" +
        "FROM\n" +
        "    [model].[NTNU-view]\n" +
        "WHERE\n" +
        "    YEAR(SamplingDate) = ?1 AND MONTH(SamplingDate) = ?2 " +
        "AND ProductionAreaBwId BETWEEN 1 AND 13 AND LocationPlacementId = 1 \n" +
        "GROUP BY\n" +
        "    ProductionAreaBwId\n" +
        "ORDER BY ProductionAreaBwId",
    nativeQuery = true)
Double[][] findAllPercentagePositiveByDate(int year, int month);
```

Code 9 (Example of a native query using JpaRepository)

Unfortunately, using native queries in JpaRepository had its own catch. The main issue with JpaRepository is that a repository can only be connected to a single entity class, and it only works with the fields of the associated entity class.

```
4 usages
public interface SampleDataRepository extends JpaRepository<SampleData, Void>, JpaSpecificationExecutor<SampleData>
```

Code 10 (example of the SampleData entity class being associated to repository interface)

As a result, the return type must contain all the corresponding fields defined in the entity class. It was attempted to create a DTO which only used the fields that are actually needed for usage in the frontend, but this was not allowed, as it did not fulfill all the fields of the entity class associated to the repository.

One workaround was to return a table with the datatype Double[][], a two-dimensional array, to extract data. This approach was used for a certain period, but it was sub optimal to work with, as the data must be manipulated from a matrix standpoint, rather than a list of arrays which are easier to index and use in general. This approach also has a lot of limitations, as if not all the fields of the data extracted from the query are of the intended type, it will not work.

### 4.2.5.3 JDBC Template

Having encountered the problem with JpaRepository, it was concluded that a different approach had to be used for querying data. After doing some research, it was discovered that JdbcTemplate could be used for extracting data, as this quiring feature has the flexibility to extract exactly the data that is desired. It has also proved to be concise, as both entity and repository classes serve no purpose using JdbcTemplate, as it does not use that format. This way, each table in the database only needs a service and controller class, and results in a cleaner structure altogether.

```
/**
 * This query selects all tests for a specific disease and year,
 * and then returns the average risk values for each month
 *
 * @param year The year of the data selected.
 * @param disease The name of the disease.
 * @return a list of RiskForEveryMonthDTO
 */
1 usage
public List<RiskForEveryMonthDTO> getRiskForSpecificDiseaseEveryMonth(int year, String disease) {
    String sql = "SELECT MONTH(SamplingDate) AS Month, " +
        "(COUNT(CASE WHEN ResultText = 'POS' THEN 1 END) * 1.0) / NULLIF(COUNT(ResultText),0) AS AverageValue\n" +
        "FROM [model].[NTNU-view] WHERE LocationPlacementId = 2\n" +
        "AND ProductName = :disease AND YEAR(SamplingDate) = :year\n" +
        "GROUP BY MONTH(SamplingDate) ORDER BY Month";

    Map<String, Object> params = new HashMap<>();
    params.put("year", year);
    params.put("disease", disease);

    return namedParameterJdbcTemplate.query(sql, params, (resultSet, rowNum) -> {
        int month = resultSet.getInt( columnLabel: "Month");
        double averageValue = resultSet.getDouble( columnLabel: "AverageValue") * 100;

        return new RiskForEveryMonthDTO(month, averageValue);
    });
}
```

Code 11 (Example of a query using jdbcTemplate)

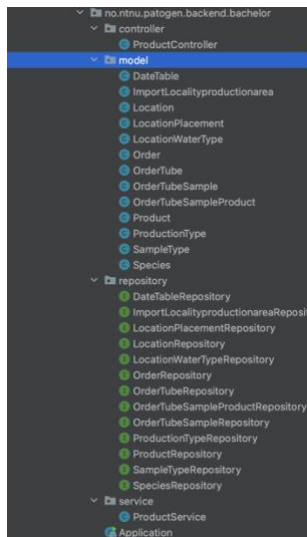


Figure 22 (Initial file-structure, service and controller classes not fully implemented)

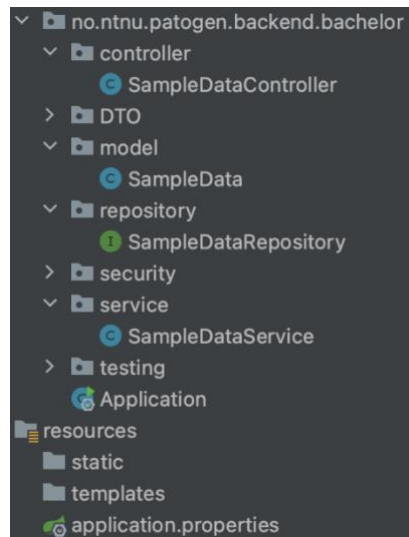


Figure 21 (Jpa, NTNU-view file-structure)

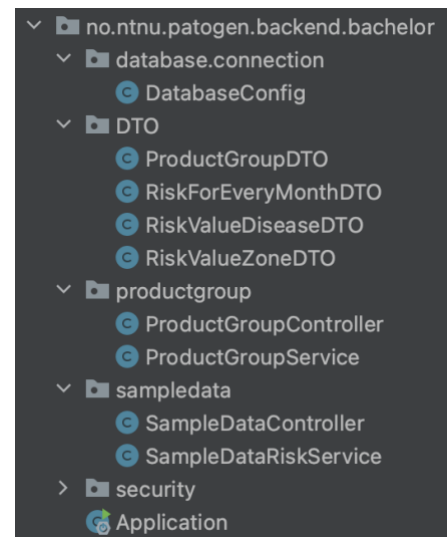


Figure 20 (Jdbc, updated NTNU-view file-structure)

## 4.2.6 Assessing disease impact

Diseases exhibit significant variation, encompassing distinct properties and consequences for fish farms. Their ability to thrive or struggle is influenced by diverse climates and conditions. For instance, certain diseases, such as NSP1PD, are considered critical when detected above a specific geographical latitude. According to the client, even the detection of a single positive sample warrants giving the entire production area a critical status (symbolized as a red color).

Different diseases also have certain thresholds or “tipping points” for when the disease should be treated as a threat. These thresholds indicate the critical level at which a disease can rapidly spread throughout a fish population, potentially causing an outbreak. The *outbreak thresholds* are unfortunately not researched enough, so the client was not able to provide set of values for thresholds, or any systematic way of handling different diseases in the dataset. Therefore, all diseases are handled equally in all the data calculations in the project. The client specified that a more concise way of treating different diseases in the dataset could be implemented in future versions of the application if more research is done regarding these tipping points.

The calculations made to assess what traffic light-color to assign to a production area was as follows:

- 1) The selected date, disease and production area are parameters selected by the user. These parameters change the displayed data on the webpage.
- 2) When these parameters are selected, associated data is retrieved from the database.
- 3) For every selected disease, the ratio of negative and positive test results is calculated by dividing positive tests by the total number of tests.
- 4) If the ratio is above certain thresholds (15% for yellow, 30% for red), a color is displayed in the zone.

## 4.2.7 Deployment

The project files are copied to the server using Git clone, which are continuously active on the server. The frontend is compiled to a build file, which is used to host the application.

To comply with the authentication requirements of PatoGen, the hosted application needed to have https implemented, since it is a requirement for using Azure Active Directory on a publicly hosted site. PatoGen provided a wildcard certificate for this, so that the name of the hosted website is configurable. By request, the name of the site is trafikkllys.patogen.no.



Figure 23 (The URL of the website)

## 4.3 Administrative results

Due to the team working beside each other every day, fixing bugs, and handling other difficult problems, all the progressions were thoroughly discussed between each other and ultimately figured out. Every member usually came up with their own opinions but also listened to other solutions. This process proved to be crucial in the development of the application. It often resulted in less need for research, as the need for reading long stack traces or searching the web for potential fixes was minimal if someone within the team had knowledge about the problem.

As the team worked together in person, sharing information and notes became slightly inorganic. However, as time went on the team became better at noting both small and important matters down in Jira or Confluence. These tools were used throughout the entire process to mark deadlines and progress. Notes from our meetings with supervisor or client are stored in Confluence and tasks in Jira making it possible for attendees to look through or reference them later. Working in the same codebase, utilizing git for version control helped us tremendously.

### 4.3.1 Using Jira & Confluence

For assigning tasks and keeping every team member up to date, Jira was used. Every user can create their own task and assign it to others or themselves. After the person assigned an issue/task is finished, the task could be marked as done, letting the team know.

Workflows, Roadmaps, Issue logs are provided as attachments to this document (in "Project Manual" chapter 2, 5, 6).

Confluence allows teams to collaborate in a space where only members can edit, supervisors or clients are invited to review meeting notes or updates if necessary. The team has used confluence to create roadmaps, take meeting notes and give sprint retrospectives. ("Project Manual" chapter 3).

### 4.3.2 Git

Having previous experience utilizing GitHub and git in general really makes a difference when working together in groups. The ability to continuously extend and work together on the same code allows for new functionalities to be seamlessly integrated. The group used git for version control throughout the whole development process of this application. Having continuous commits with over 300 in total spread evenly across the spring-semester ("Project Manual" - git-contribution summary).

GitHub Actions, a feature available in GitHub was used by the team mainly for continuous integration, checking that newer commits passed certain tests. If passed, the team knew that latest changes could be deployed without failure.

### 4.3.3 Milestones

The team created a subfield in the pre project plan called milestones. Here important dates were set to ensure that the team would be working towards finishing certain parts of the project within set dates. However, this has proved to be somewhat inaccurate as it does not really take development into consideration. For administrative work it represents the multiple subtasks the team have had to complete along the way and development has been done in between these dates when schedule allowed it (Attachments "Pre project plan 3.2").

After looking back at the milestones that were set, it is clear to see that the team was looking for a steady workflow, starting and finishing tasks early to have sufficient time in the end to perfect both product and report. The team has been successful in sticking to this philosophy as the team was able to complete tasks along the way with enough time to complete the project over the past months.

### 4.3.4 Quality insurance

To ensure that the project met all expectations, the team wrote down certain measures. These include thorough planning, documentation, and continuous feedback from both supervisor and client. After finishing some new functionality, the team would always discuss these with the client to ensure that it was understandable and necessary. As documented in confluence, every meeting the team had with the client was concluded with a meeting note including changes and discussions for a better product. The feedback that was received from these meetings really set the standard for the quality of this project, as it helped the team a lot with understanding the business and the client's needs.

Another key step in ensuring quality was assigning roles within the team, having a team lead helped substantially with keeping focus and initiating meetings. Also, members being responsible for keeping the overall quality of design and code up to par was necessary to understand each other's changes etc. Lastly someone that oversees documents and structure proved handy as the team were required to provide substantial documentation for the end delivery.

## 4.4 Performance and Optimization Results

### 4.4.1 Application performance

During the development process, the team monitored the application's performance by using browser-based tools such as Google Chrome's Developer Tools. The team ensured that the application responded quickly to user inputs, minimizing the load time for content, and maintaining a smooth user experience. Regular performance testing and optimizations were carried out to ensure that the application continued to perform well as new features were added.

### 4.4.2 Code optimization

The team continuously reviewed the codebase for opportunities to optimize and improve the overall code quality. By adhering to best practices, the team minimized technical debt and ensured that the code was maintainable and scalable for future developments. Code reviews were conducted regularly, which provided opportunities for the team to identify and address any potential bottlenecks or areas for improvement.

### 4.4.3 Accessibility and usability

Throughout the development process, the team focused on creating an accessible and user-friendly interface for the application. By following accessibility guidelines and best practices, the team ensured that the application could be used by a diverse range of users, including those with disabilities. The team performed usability testing with different user groups to gather feedback and adjust if needed, ensuring that the application met the needs of its intended audience.

### 4.4.4 Code reliability

To ensure reliable, bug free code, multiple proactive steps were implemented. Measures such as CI (continuous integration) and testing stands for the majority of reliability insurance.

#### 4.4.4.1 Continuous integration

Using GitHub Actions, two workflow files were created to ensure consistency between commits and maintain reliability of code. One workflow file is to build and test the react application, as well as the spring boot application. This sets up a virtual machine for both the frontend and backend, and checks if it can compile. If it succeeds, it means the code can compile and run on the virtual machine, which means it works for any system with correct versions of npm and java, respectively. Thereafter, locally created tests in both the frontend and backend are run to ensure the application works as intended.

```
jobs:
  Build-test-frontend:
    runs-on: ubuntu-latest

    env:
      CI: false

    steps:
      - name: Checkout repository
        uses: actions/checkout@v2

      - name: Build React app
        uses: actions/setup-node@v2
        with:
          node-version: "19.x"
      - run: |
        cd frontend
        npm ci
        npm run build
        npm test
```

Code 13 (Build frontend job)

```
Build-test-backend:
  runs-on: ubuntu-latest

  steps:
    - name: Checkout repository
      uses: actions/checkout@v2

    - name: Build Spring Boot app
      uses: actions/setup-java@v2
      with:
        java-version: "17"
        distribution: "temurin"

    - run: |
      cd Backend/patogen.backend.bachelor
      mvn clean verify -Dspring.datasource.username=${{ secrets.DB_USERNAME }}
```

Code 12 (ci build backed job - all parameters in mvn clean verify not depicted)

The second workflow file was later implemented later in development as it consists of automating Postman endpoints tests for the backend running on the server. This is to ensure the server's endpoints are working as intended. It runs a pre-created folder in Postman with the endpoint tests as well as a fleet of internal JavaScript tests to verify the data received is as desired in terms of having the correct fields, being correct datatype, etc.

```

name: Test server endpoints

on:
  push:
    branches:
      - master

jobs:
  automated-api-tests:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up Node.js
        uses: actions/setup-node@v2
        with:
          node-version: 14
      - name: Install Newman
        run: npm install -g newman
      - name: Download and run Postman collection
        env:
          POSTMAN_API_KEY: ${ secrets.POSTMAN_API_KEY }
        run: |
          newman run https://raw.githubusercontent.com/Postmanlabs/newman-test-suite/master/collections/collection.json --apikey=${ secrets.POSTMAN_API_KEY }

```

Code 14 (CI Automated Postman test)

These workflow files are set to trigger on each git push, meaning that every time new code is submitted to the codebase, these files will be run.

#### 4.4.4.2 Unit testing

A part of the continuous integration workflow are unit tests for the backend and frontend. The frontend tests commonly used methods as well as all existing redux states. The backend has simple tests for all DTOs.

The tests check if constructors and functions work as initially intended, and that objects and redux states can be initialized, and be mutated – if it corresponds with business logic.

## 4.5 Integration and Compatibility Results

### 4.5.1 Integration with external systems

The team successfully integrated the application with external systems such as Microsoft Azure Active Directory for authentication and Google Maps API for map integration. The integration process made the team able to leverage the features and functionality provided by these external systems effectively, enhancing the application's capabilities. An extensive look at the communication between application, server and database can be found in attachments "Sequence diagram".



## 4.5.2 Cross-platform and browser compatibility

The team tested the application on various platforms (Windows, macOS, Linux) and in different web browsers (Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari) to ensure that it was fully compatible and functional across different environments. The team addressed any compatibility issues that were identified during the testing phase, ensuring that the application provided a consistent user experience regardless of the platform or browser used.

The application is also fully scalable, slightly manipulating the UI to fit smaller and larger devices (4.3.2). This makes it possible to use the application on both computer, tablet and mobile. The application is intended for customers, employees, and experts to show each other discoveries and trends, which means that the application should be functional on mobile devices, as they are generally available out of the office.

## 4.6 Learning Outcomes and Skill Development

Throughout the project, the team members were able to develop and enhance their skills in various areas, including:

- Project management and teamwork: Working together in a structured manner, managing tasks, and maintaining open lines of communication.
- Software development: Enhancing the team's understanding of React.js, Spring Boot, and other relevant technologies.
- Problem-solving and debugging: Tackling complex issues and finding solutions by leveraging the collective knowledge and experience of the team.
- Research and learning: Exploring new tools, technologies, and techniques to improve the application's functionality and overall quality.
- Presentation and documentation: Effectively communicating the project's progress and results to the client, supervisor, and other stakeholders.

The skill development and learning outcomes not only contributed to the successful completion of the project, but also provided valuable experience for the team members, preparing them for future challenges in their careers.

## 5 Discussion

In the discussion chapter, the focus will be on interpreting and analyzing the results obtained from the entire process. Discussing the choices that were made and the potential methods that could have been used will give a deeper understanding and insight for both the reader, as well as the team.

### 5.1 Choice of technologies

React.js was chosen for the frontend due to the team's familiarity with the technology, which allowed for rapid development of the application. The use of React.js also provided an advantage in terms of scalability and maintainability, as it is a widely used library with extensive documentation and community support. Spring Boot was chosen for the backend as it enabled easy integration with PatoGen's Azure database and provided a robust platform for building the necessary APIs. Using Microsoft Azure Active Directory for user authorization was a logical choice, as PatoGen and its users were already familiar with the system.

The use of Google Maps API allowed for seamless integration of map features and polygons, providing a visually appealing and interactive interface for users. Additionally, the use of Git and GitHub for version control and collaboration proved invaluable, allowing the team to work simultaneously on the codebase while maintaining a prominent level of quality. Lastly, the use of an Openstack server with nginx to host the application proved to be simple and effective.

### 5.2 Team collaboration and communication

The team's approach to collaboration and communication played a crucial role in the effective completion of the project. By working closely together and frequently discussing issues, the team was able to effectively problem-solve and avoid significant roadblocks in the development process. The use of Jira and Confluence for project management and documentation also contributed to the team's success, ensuring that tasks were properly assigned, and progress was documented.

#### 5.2.1 Version control

As mentioned in chapter 4.3.2 "Git", the team used GitHub for version control during the development of this application. Initially the team was torn between options, creating separate branches for development, or keeping the development all in the same branch. Since the team valued teamwork and planned to work together in person every day, it seemed

easiest to all be working within the same “master” branch. Starting out, this strategy worked out well, but eventually some bad changes were committed, and some development time was lost to joint debugging. However, the team learned from their mistakes and became careful when committing changes and closely analyzed their changes before pushing them to the main branch.

Another, more common approach, which arguably should have been used in combination with branching, is pull requests. This would integrate code reviewing on each code push, which has been proved to be effective in reducing bugs and improving code quality, since it is reviewed by more collaborators [35].

By likelihood, creating separate branches for different development processes would be a better approach. This would allow the team to work on separate tasks without them interrupting one another.

## 5.2.2 Testing

The usage of testing in the project is deemed sufficient, as it tests a large amount of functionality for both the backend and frontend. A case could be made for testing graphical components in the frontend could prove beneficial, but these tests are difficult and timely to create, and with changes in design, might even be replaced in later development.

Instead of having tested the backend in spring boot, Postman is used to test the endpoints and data received. This is again because more complex testing in spring boot like mock testing is difficult and time consuming and testing in Postman proved simple and effective.

## 5.3 Time constraints and milestones

While the team fulfilled all expectations listed in “milestones” (Attachments “pre project plan”) and completed the project, it was acknowledged that more work could have been done earlier in the semester. The mandatory subject INGA2300 limited the team's available time and energy for the bachelor thesis, which may have impacted the overall quality of the project. Despite these constraints, the team was able to deliver a functional and valuable application for PatoGen.

## 5.4 Quality assurance

The team's commitment to quality assurance through planning, documentation, and continuous feedback from the supervisor and client ensured that the project met expectations. Assigning specific roles within the team for overseeing design, code quality, and documentation contributed to the overall success of the project.

## 5.5 Alternative approaches

While the chosen technologies and methodologies proved successful for the project, alternative approaches could have been explored. For example, using a different frontend framework or library, such as Angular or Vue.js, could have yielded different results in terms of development speed and application performance. Additionally, alternative backend frameworks or database systems might have been considered, depending on the team's experience and the project's specific requirements.

In conclusion, the team's choice of technologies, collaborative approach, focus on simplicity, and commitment to quality assurance resulted in the beneficial development of a functional and valuable application for PatoGen. Despite facing time constraints and competing priorities, the team was able to effectively meet the project's objectives and deliver a solution that met the client's expectations.

## 5.6 Agile Methods

### 5.6.1 Sprints

Although agile methods were used during development, the implementation of it could have been better. For instance, sprints were intended to be 2 weeks long, but sometimes the sprints were finished at inconsistent times, resulting in what should have been a total of 8 sprints, turned into 7. There did not arise any negative consequences of this, but in general, it is good practice to follow the initial plan, as too many deviations can lead to the team losing track of the original goal.

### 5.6.2 Daily standups

A common practice of agile methods are daily standups. Since the group met daily in person, there was never a question about what each member was working on, as the team communicated closely and contributed to solving each other's problems and providing different views and ideas for each other during development. Therefore, there was never a strict point of the day for a daily standup, but rather a natural stream of communication.

### 5.6.3 Jira

In terms of Jira, there could have been more logging of issues. The main cause for the lack of logging is due to the team consistently working closely in person, making logging more of a hinderance than a useful feature, since each member knew what the other members were doing. In hindsight, the number of issues logged was sufficient, but in a different scenario where documenting work and being able to refer to completed tasks is of high importance, logging issues should receive more focus.

### 5.6.4 Confluence

In terms of using confluence, it served as a handy tool to document notes, especially for meetings. Despite this, it should possibly have been used more frequently in taking notes about more important decisions and complex work completed, since these factors can be easily forgotten, and can be useful in the context of the report.

### 5.6.5 Client and supervisor

There were also inconsistencies with meetings with our client and supervisor, which were originally planned to be at the end of each sprint (every two weeks). Both our supervisor and client had busy schedules, and meeting rooms on campus were often unavailable. It quickly became a very difficult task to arrange meetings where all these 3 factors were aligned. This resulted in more casual forms of communication, like a discord channel where our client was included. Because of this, the group had great, continuous communications with the client, and were able to receive constant feedback and information.

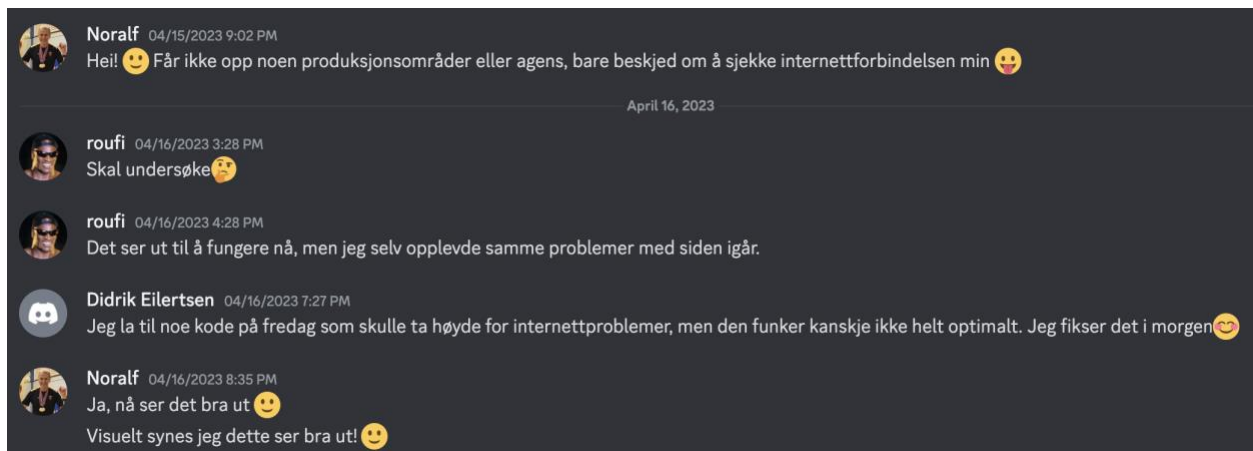


Figure 24 (Example of communications with client using discord)

Nevertheless, the group still had several meetings with the client, often at their offices or in online conference calls. The group maintained the goal of having meetings with the client

every two weeks, but they were not exclusively held at the last date of the sprints. Receiving continuous feedback from the client provided necessary information for the group to know what steps were next and were always on the right course.

## 5.7 Fulfillment of Client Requirements

The requirements from the client have been listed in Chapter 1.3. This chapter will evaluate whether these requirements have been met.

### 5.7.1 Confidentiality

PatoGen's dataset comprises sensitive and confidential information concerning their customers, warranting strict limitations on access by other users. The team has effectively addressed this requirement in the current version of the application, ensuring that no competitor can extract any specific customer data through any means.

Furthermore, the team delivered another specific requirement: data will only be displayed when information from at least three customers is available simultaneously. This logic is implemented in a manner that regardless of the user's search parameters, no data will be shown unless the requested dataset contains a minimum of three customers.

### 5.7.2 User Authentication

The client wanted to authenticate users as an extra security measure to restrict the possibility of leaking confidential information about their customers to the public. The team implemented this by using the Microsoft Authentication Library, which restricts access to the application to a list of users defined in Azure Active Directory. A more in-depth description of this process is found in chapter 4.2.3 "Authorizing accounts".

### 5.7.3 Data visualization

The client presented specific requirements for data visualization in their system. They desired a user-friendly interface that would enable users to easily compare data from different production areas and apply specific search parameters to filter the data. Although PatoGen possessed a thorough understanding of the provided data, they lacked a clear vision for its visualization. Consequently, the development team took the initiative to brainstorm and propose visual solutions, which were later confirmed with the client.

The implementation of filtered searches and a date picker feature enhanced the system's usability, enabling users to customize their experience by viewing and analyzing data according to their specific requirements.

## 5.7.4 Graphical interface

In response to the client's request, the team embraced the "KISS" (Keep It Simple Stupid) principle as a guiding philosophy when designing and developing the application's graphical user interface. This approach was chosen to ensure the creation of an intuitive and easily comprehensible interface for users. By prioritizing simplicity, the team focused their efforts on delivering essential functionality while effectively visualizing PatoGen's data in a meaningful manner.

# 6 Conclusion

Building upon the previous chapters, the conclusion will not only provide a comprehensive overview of the entire project but also emphasize the key contributions made throughout the research. By reflecting on the methodology, analysis, and findings, this final chapter will offer valuable insights into the practical applications and potential benefits of the work.

Moreover, the conclusion will serve as an opportunity to critically assess the strengths and weaknesses of the research, acknowledging any areas that could be improved or further explored. Through this evaluation, the conclusion will present a balanced perspective on the overall quality and relevance of the thesis, ultimately contributing to a deeper understanding of the topic and its broader implications in the field.

## 6.1 Summary of Achievements

### 6.1.1 Technology Selection

One of the key factors contributing to the successful development of the web-based application for PatoGen was the selection of appropriate technologies. The team's choice of React.js for the frontend allowed for rapid development while maintaining scalability and maintainability. Spring Boot was chosen for the backend due to its seamless integration with PatoGen's Azure database and its robust platform for building APIs. Furthermore, Microsoft Azure Active Directory provided a familiar and efficient authentication system, while the Google Maps API facilitated the creation of an interactive and visually appealing map interface. The combination of these technologies enabled the development of a feature-rich and user-friendly application.

## 6.1.2 Team Collaboration and Project Management

Effective team collaboration, communication, and project management played a crucial role in the completion of this project. Utilizing Jira and Confluence for task assignment, progress tracking, and documentation ensured that the team remained organized and efficient throughout the development process. Regular meetings, open lines of communication, and an emphasis on teamwork facilitated problem-solving and enabled the team to overcome challenges that arose during the project.

## 6.1.3 Application Design and Functionality

The team's adherence to the "KISS" (Keep It Simple Stupid) principle in the design and development process resulted in an intuitive and user-friendly application that effectively visualized PatoGen's data in a meaningful and accessible way. By prioritizing core functionality and a clean, straightforward user interface, the team was able to create an application that met the needs of its intended audience and provided value to PatoGen.

## 6.1.4 Overcoming Time Constraints

Despite facing considerable time constraints due to parallel subjects, the team managed to meet the set milestones and deliver a functional and valuable application for PatoGen. This achievement underscores the team's dedication, resilience, and ability to prioritize tasks effectively under challenging circumstances.

## 6.1.5 Quality Assurance

The team's commitment to quality assurance was evident through their thorough planning, comprehensive documentation, and continuous feedback from the supervisor and client. By assigning specific roles within the team to oversee design, code quality, and documentation, the team ensured that the project maintained high standards and met the expectations of all stakeholders.



## 6.2 Limitations and Future Research

### 6.2.1 Alternative Approaches

While the chosen technologies and methodologies proved to fit seamlessly for this project and should count as modern for years to come, alternative approaches could be explored in future research and development. For example, the use of different frontend frameworks or libraries, such as Angular or Vue.js, might yield different results in terms of development speed and application performance. Similarly, alternative backend frameworks or database systems could be considered, depending on the team's experience and the specific requirements of the project.

### 6.2.2 Additional Features and Improvements

Future research could also investigate the implementation of additional features and improvements to enhance the application's functionality and user experience. These enhancements might include advanced data visualization techniques, more granular filtering options, or the integration of machine learning algorithms for predicting fish health. By incorporating these new features, the application could provide an even greater value to PatoGen and its stakeholders.

As discussed in chapter 4.2.6 "Assessing disease impact", different diseases should optimally be handled in a unique way in all calculations, as the different diseases encompass distinct properties that operate under specific conditions. If more research is done regarding this subject, a future version of the application could implement a more concise way of handling specific diseases when calculating risks. Custom messages and warnings could be implemented if certain diseases were close to "outbreak thresholds". Local geographical conditions and parameters (water temperature, seasons, latitude, salt-levels) could also be regarded when handling the datasets.

## 6.3 Identified problems

### 6.3.1 Spring Boot in deployment

One issue arose after development had finished. Sometimes the Spring Boot backend on the server stops working, currently the exact reason behind it is unknown, but after doing some debugging, it was discovered the spring boot process running the backend is most likely stuck in an "Interruptible sleep state" which suggests that the process is waiting for an

event to complete [36]. As a result, none of the endpoints are reachable, and no data can be retrieved.

This happens at seemingly random moments, usually days or weeks after deployment. After the backend stops working, the only known fix is to restart the backend on the server. To combat this issue, logging was implemented to log error messages output by the system, which hopefully will provide information associated with the problem so it can be resolved later.

## 6.4 Final Remarks

In conclusion, this bachelor thesis project demonstrates the promising development of a beneficial web-based application for PatoGen that effectively visualizes fish health data, providing valuable insights for stakeholders in the aquaculture industry. The team's choice of technologies, collaborative approach, focus on simplicity, and commitment to quality assurance have resulted in a solution that meets the client's expectations and offers a solid foundation for future enhancements and research opportunities. Through this project, the team has gained valuable experience and skills in project management, software development, problem-solving, and teamwork, preparing the team for future challenges in their careers.

## 7 Societal Impact

The aim of this chapter is to discuss the societal impact of this bachelor thesis project. The project is carried out by students at The Norwegian University of Science and Technology (NTNU) and employees at PatoGen, a company specializing in fish health and disease analysis in Norwegian fish farms. Both are regarded as institutions with a common value, sustainability, new technology and groundbreaking development. New technology and developments should be sustainable and have a positive economic and environmental impact.

### 7.1 Benefits for Aquaculture Industry

The application offers numerous benefits for the aquaculture industry, including improved monitoring and management of fish health. By providing a visual representation of the health data, the application enables fish farm owners and managers to make better-informed decisions regarding the welfare of their fish stocks. This can lead to reduced fish mortality, improved productivity, and overall increased efficiency within the industry. Especially for countries like Norway, where seafood represents a large amount of the country's export; 2.9 billion tons in 2022 [37]. Using applications like this one could potentially have huge economic and sustainable impacts.

#### 7.1.1 Economic implications

The application's ability to optimize fish health management can have positive economic implications for the stakeholders involved. Healthier fish stocks can lead to higher yields and better-quality products, which in turn can increase profitability for fish farm owners. Moreover, by enhancing the sustainability and efficiency of the industry, the application can help ensure the long-term growth and success of the Norwegian aquaculture sector. Implementing the traffic light system for individual production areas could result in a growth of 24000 additional tons of exported fish [2], the Norwegian government believes.

### 7.2 Environmental Impact

By promoting optimal fish health and fish farm management practices, the application can potentially reduce the environmental impact of fish farming. Healthier fish stocks may result in fewer disease outbreaks and therefore a decreased reliance on antibiotics and other medications. Additionally, a well-managed fish farm has a lower risk of negatively affecting local ecosystems. These are examples of situations where this application can reduce the environmental footprint of this large industry.

## 7.3 Public health and food industry

The application's visualization tool can also have indirect benefits and impact on the public health and food industry. By reaching better and healthier fish stocks, the application can contribute to the production of safer and higher-quality seafood products for consumers. Furthermore, by supporting a more sustainable and efficient aquaculture industry, the application can play a role in ensuring a reliable and secure supply of fish products to meet the growing global demand for seafood.

## 7.4 Future Research and Development

Providing the opportunity for users to further filter their searches could also be implemented to offer users the ability to parse and analyze highly specific data, even though the main design principle was to keep the interface as simple and understandable as possible. Introducing location-specific data into the application is also a possibility. Due to the public's restricted access to PatoGen's data, the team was only able to use information about production areas to visualize a more general status. If PatoGen chooses to further work on this application, it could be made possible to let the customers see more personalized data, for instance their own confidential data about specific fish farms.

The societal impact of the application is expected to grow as more research is conducted and new features are developed. Future research could potentially focus on updating the visualization techniques, integrating additional data sources like open-source data about fish-lice, and maybe explore advanced analytics including artificial intelligence for predicting outcomes and development within the fish farms. These improvements can further enhance the application's ability to support sustainable development and promote healthy fish farming practices.

## Bibliography

- [1] C. Knudsen, "58 millioner laks døde i norske oppdrettsanlegg i 2022," 7 February 2023. [Online]. Available: <https://e24.no/hav-og-sjoemat/i/4omy9G/58-millioner-laks-doede-i-norske-oppdrettsanlegg-i-2022>.
- [2] Regjeringen, "Regjeringen skrur på trafikklyset i havbruksnæringen," 4 February 2020. [Online]. Available: <https://www.regjeringen.no/no/dokumentarkiv/regjeringen-solberg/aktuelt-regjeringen-solberg/nfd/nyheter/nyheter-2020/regjeringen-skrur-pa-trafikklyset-i-havbruksnaringen/id2688939/>.
- [3] IBM, "ibm.com," 15 March 2023. [Online]. Available: <https://www.ibm.com/docs/en/informix-servers/14.10?topic=server-clientserver-communication>.
- [4] Oracle, "What is a relational database?," [Online]. Available: <https://www.oracle.com/database/what-is-a-relational-database/#link2>.
- [5] Atlassian, "What is Agile?," [Online]. Available: <https://www.atlassian.com/agile>.
- [6] Josikakar, "Software Engineering | Coupling and Cohesion," 18 April 2023. [Online]. Available: <https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/>.
- [7] IBM, "What is software testing?," [Online]. Available: <https://www.ibm.com/topics/software-testing>.
- [8] T. Hamilton, "Unit Testing Tutorial – What is, Types & Test Example," 4 3 2023. [Online]. Available: <https://www.guru99.com/unit-testing-guide.html>.
- [9] Amazon Web Services, "What is Continuous Integration?," [Online]. Available: <https://aws.amazon.com/devops/continuous-integration/>.
- [10] Educative, [Online]. Available: <https://www.educative.io/answers/client-side-vs-server-side>.
- [11] Universitetet i Oslo, 19 May 2021. [Online]. Available: <https://www.mn.uio.no/ibv/tjenester/kunnskap/plantefys/leksikon/p/pcr.html>.
- [12] A. J. B. K. S. D. Jessica Penney, "National Library of Medicine," 8 April 2022. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9088631/>.
- [13] V. Kanade, "Spiceworks.com," 22 July 2022. [Online]. Available: <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-hci/>.

- [14] B. Kopf, "Toptal," [Online]. Available: <https://www.toptal.com/designers/ui/figma-design-tool>.
- [15] Angular, "What is Angular," 28 February 2022. [Online]. Available: <https://angular.io/guide/what-is-angular>.
- [16] R. Thankaraj, "Micronaut vs Quarkus vs Spring Boot Reactive Framework Deep Comparison," 3 July 2021. [Online]. Available: <https://regupathit.medium.com/quarkus-vs-micronaut-a-deep-comparison-84cae4fea966>.
- [17] P. Loshin, "techtarget.com," February 2022. [Online]. Available: <https://www.techtarget.com/searchdatamanagement/definition/SQL>.
- [18] Reactjs, "A JavaScript library for building user interfaces," [Online]. Available: <https://legacy.reactjs.org/>.
- [19] Material UI, "Material UI," [Online]. Available: <https://mui.com/>.
- [20] Recharts, "Recharts," [Online]. Available: <https://recharts.org/en-US/>.
- [21] N. Ighodaro, "Understanding Redux: A tutorial with examples," 3 October 2022. [Online]. Available: <https://blog.logrocket.com/understanding-redux-tutorial-examples/>.
- [22] IBM, "What is Java Spring Boot," [Online]. Available: <https://www.ibm.com/topics/java-spring-boot>.
- [23] M. Mulders, "What Is Spring Boot," 16 September 2019. [Online]. Available: <https://stackify.com/what-is-spring-boot/>.
- [24] git, "git --fast version control," [Online]. Available: <https://git-scm.com/>.
- [25] J. Juviler, "blog.hubspot," 14 September 2022. [Online]. Available: <https://blog.hubspot.com/website/google-maps-api>.
- [26] GitHub, "Understanding GitHub Actions," [Online]. Available: <https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions>.
- [27] OpenAI, "About," [Online]. Available: <https://openai.com/about>.
- [28] OpenAI, "GPT-4," [Online]. Available: <https://openai.com/research/gpt-4>.
- [29] Javatpoint, "Postman Tutorial," [Online]. Available: <https://www.javatpoint.com/postman>.
- [30] Microsoft, "Overview of the Microsoft Authentication Library (MSAL)," 10 December 2022. [Online]. Available: <https://learn.microsoft.com/en-us/azure/active-directory/develop/msal-overview>.

- [31] Kinsta, "What Is Nginx? A Basic Look at What It Is and How It Works," 26 01 2022. [Online]. Available: <https://kinsta.com/knowledgebase/what-is-nginx/>.
- [32] R. Grischenko, "Mighty," 15 October 2021. [Online]. Available: <https://www.mighty.digital/blog/data-modeling-techniques-explained>.
- [33] Indeed Editorial Team, 10 March 2023. [Online]. Available: <https://www.indeed.com/career-advice/career-development/client-side-vs-server-side>.
- [34] Eukhost, "What is Server Side Scripting (Guide of Pros and Cons)," 20 December 2015. [Online]. Available: <https://www.eukhost.com/blog/webhosting/server-side-scripting-pros-and-cons/>.
- [35] F. Morina, "Benefits of Using Pull Requests for Collaboration and Code Review," 01 December 2022. [Online]. Available: <https://developer.nvidia.com/blog/benefits-of-using-pull-requests-for-collaboration-and-code-review/>.
- [36] J. L. Perez, "baeldung," 30 November 2022. [Online]. Available: <https://www.baeldung.com/linux/uninterruptible-process>.
- [37] Norwegian Seafood Council, "seafood.no," 4 January 2023. [Online]. Available: <https://seafood.no/aktuelt/nyheter/norge-eksporterte-sjomat-for-1514-milliarder-kroner-i-2022/>.
- [38] Postman, "What is Postman?," [Online]. Available: <https://www.postman.com/>.
- [39] A. Naor, "freecodecamp.org," 25 August 2020. [Online]. Available: <https://www.freecodecamp.org/news/keep-it-simple-stupid-how-to-use-the-kiss-principle-in-design/>.





## Attachments

Attachments include mandatory documents that is not directly relevant to subjects in the main report or is too large to follow the flow of the document. Additional attachments like pre project plan were initially written in Norwegian, however a copy in English is provided to ensure that all readers understand the whole picture. Some parts within the pre project plan are deleted for this submission as it included private information that should not be published without consent.

## Contents

<b>Forprosjektsplan (Norsk)</b> .....	2
<b>Pre Project plan (English)</b> .....	9
<b>Sequence Diagram</b> .....	15
<b>Wireframes</b> .....	16
<b>Application Result Overview</b> .....	18

# Forprosjektsplan (Norsk)

## Mål og rammer

### 1.1 Orientering

Oppgaven "Risikomodel og webapp" fra PatoGen var den åpenbare kandidaten for valget vårt av bacheloroppgave. Dette er først og fremst fordi vi har lyst til å lage en enkel, visuell og brukervennlig nettside, men også fordi fiskeoppdrett er en bransje med interessante utfordringer og problemstillinger. Samtlige i gruppen vil også gjerne opparbeide seg erfaring fra hav og fiske-industrien. Dette er fordi vi er bosatt i Ålesund og industrien har en stor markedsandel i byen.

Denne oppgaven var øverst på vår liste over mulige bachelorprosjekter, noe vi tydelig klargjorde i prosjektsøknaden hvor vi ytret våre egenskaper og motivasjon for utførelsen av denne oppgaven.

### 1.2 Problemstilling / prosjektbeskrivelse og resultatmål

Oppdragsgiveren PatoGen jobber med ressursutnyttelse av oppdrettsfisk. Dette gjennomfører de hovedsakelig ved å analysere sykdomsforløp til fisken. Ansatte og kunder av PatoGen ser et økende behov for en brukervennlig løsning som gir en totaloversikt over nåværende sykdom- og smittestatus til oppdrettsanleggene.

Oppdrett i Norge deles inn i flere geografiske soner. Disse produksjons-områdene følger det såkalte "trafikklyssystemet", hvor hver sone blir tildelt rødt, grønt eller gult lys - som representerer hvorvidt produksjonsområdet kan øke eller trappe ned produksjonen for å opprettholde en god fiskehelse.

Vår jobb er å visuelt framstille nåværende status og annen relevant data for samtlige geografiske soner, slik at kunder og ansatte får en bedre oversikt over tilstanden til fiskeproduksjonen i Norge, slik at de kan ta interne beslutninger for å optimalisere produksjonen sin.

Når prosjektet er ferdig er det forventet å ha utviklet et brukergrensesnitt som enkelt kan brukes av kunder og ansatte, hvor de kan få en oversikt over en rekke ulike risikofaktorer for fiskeoppdrett. Risikoene varierer, men hovedsakelig bakterier, sykdommer og lus. Brukere skal kunne navigere på et kart over Norge med alle oppdrettsanleggene, velge ut et oppdrettsanlegg og få oversikt over risikoer basert på trafikklyssystemet. Brukere må autentiseres, og autentiserte brukere skal kun ha tilgang til informasjon av fiskeoppdrett som tilhører deres firma.

### 1.3 Effektmål

Gruppen ønsker som minstekrav å tilfredsstille oppdragsgivers behov, og har ambisjoner om å imponere oppdragsgiver. Generelt ønsker gruppen å prestere på et høyt nivå.

Den ønskelige langsiktige effekten fra oppdragsgiver sin side er at kundene deres enkelt kan analysere tilstanden til fisken i oppdrettsanleggene sine. Dette vil hjelpe kundene med å ta interne beslutninger og å øke produksjonseffektiviteten. Regjeringen har anslått at et trafikklyssystem for lakselus kan gi en årlig produksjonsvekst på [23 000 tonn](#) i året. I oppgaven vår skal vi både ta hensyn til lakselus, men også andre agenser. Dette kan føre til at produksjonsveksten kan øke *enda* mer enn hva regjeringen har anslått.

### 1.4 Rammer

- Faste grupperom på campus med tavle og TV-skjerm.
- Dekning av parkeringsutgifter når møter skal holdes hos PatoGen sine kontorer.

## 2 Organisering

Involverte aktører i prosjektet er NTNU og PatoGen.

## 3 Gjennomføring

### 3.1 Hovedaktiviteter

Opplisting av hovedaktiviteter.

- Kommunikasjon med oppdragsgiver og veileder
  - Hele gruppen kommuniserer med oppdragsgiver og veileder ved jevnlig møter for å diskutere hvordan ting har gått, og hva som er veien videre.
- Design
  - Eirik Dahle er i ledelse for designet av nettsiden.
- Versjonskontroll og CI/CD
  - Github, Branching, Pull Requests, GitHub Actions til Continuous Integration.
- Utvikling og rapportskrivning
  - Alle gruppemedlemmer skal bidra med kode- og rapportskrivning underveis i prosjektet. Vi vil bruke React som bibliotek.

- Ledelse og intern kommunikasjon
  - Jones er teamleder og er ansvarlig for:
    - Kommunikasjon mellom veileder, oppdragsgiver, og gruppe-medlemmer.
    - Sørge for at frister blir opprettholdt.
    - Sørge for at kvalitet til innlevert arbeid opprettholder høy standard i samarbeid med Didrik.
    - Sørge for at samhold til gruppen er bra, at arbeidskontrakt og oppgaver blir fulgt.
- Statistikk, databehandling, representering/sammenligning av data
  - Det er visse bransjestandarder som må bli fulgt, og utfordringer som trenger spiss faglig kompetanse - så oppdragsgiver vil hjelpe mye til med databehandling.
  - Hvis ekstra statistikk-kompetanse blir nødvendig, så kan statistikk-studenter fra NTNU bidra.
- Scrum-metodikk
  - 2 ukers sprint, møte med veileder hver uke i startfasen, annenhver uke etter man kommer godt i utviklingsprosessen.
  - Sprint planning
  - Daily standups
  - Sprint review
  - Sprint retrospective
  - Jira til arbeidsfordeling og planlegging
  - Confluence til administrativt (dokumenter, referat, etc)

## 3.2 Milepæler

Opplisting av kritiske datoer.

12/01/23 - Første møte med veileder og oppdragsgiver

27/01/23 - Innlevering forprosjektplan

03/02/23 - Tilbakemelding forprosjektplan, godkjenning

21/04/23 - Muntlig presentasjon av prosjekt på engelsk

21/04/23 - Innlevering rapport til veileder for tilbakemelding

18/05/23 - Postere skal være ferdig og klar til printing

22/05/23 - Innlevering av rapport og vedlegg i inspera, trolig presentasjon av prosjekt.

## 4 Oppfølging og kvalitetssikring

### 4.1 Kvalitetssikring

For å forsikre at kvaliteten i prosjektet holder en viss standard er det viktig at vi som gruppe leser og analyserer alt arbeid som blir utført og kvalitetssikrer dette før det eventuelt vises frem til oppdragsgiver og veileder. Regelmessig kommunikasjon med veileder vil gi oss konstruktive tilbakemeldinger slik at vi kan forbedre produktet ytterligere.

Tiltak for kvalitetssikring:

- Skissere og planlegge arbeid grundig
- Dokumentere møter, kode og annet generelt arbeid
- Regelmessig testing av applikasjon og kode-eksemplarer
- Tilbakemelding fra oppdragsgiver / veileder
- Github actions for bruk av CI

### 4.2 Rapportering

Rapportering skal tilstrebes annenhver uke til veileder og oppdragsgiver, som regel på slutten av hver sprint hvor studentene møter med veileder og oppdragsgiver.

## 5 Risikovurdering

Risikoanalyse som vurderer sårbarheter i prosjektet (hendelse, sannsynlighet, konsekvens og tiltak).

	<b>Hendelse</b>	<b>Årsak</b>	<b>Sannsynlighet</b>	<b>Konsekvens</b>	<b>Forebyggende tiltak</b>
1	Bugs	Feil i kode	Svært høy  Å skrive feilfri kode kan betegnes som omtrent umulig, ettersom det er vanskelig å forutse hvordan koden kompilerer.	Lav/Middels/høy  Applikasjonen har uønsket oppførsel og kan deretter være frustrerende å bruke. I verste fall kan deler, eller hele applikasjonen være ubrukelig.	Testing, Continuous Integration, dokumentere kode, vurdere nøye hvordan kode og arkitektur fungerer, versjonskontroll.

2	Korrupt/mistet kode	Manglende versjonskontroll	<p>Lav</p> <p>Uten bruk av git kan ikke koden anses som trygg, ettersom det kan hende den ikke er lagret, og dermed kan lett tapes. Koden kan også bli korrupt hvis noe uønsket oppstår med filene til prosjektet.</p>	<p>Svært høy</p> <p>Manglende/korrupt kode kan i verste fall føre til at deler av / hele prosjektet mistes/ikke kan brukes.</p>	<p>Versjonskontroll som git, gjerne knyttet til et remote repository.</p>
3	Uforventet permanent frafall, endringer i gruppesammensetning	Sykdom, interne konflikter, private årsaker.	<p>Lav</p> <p>Kortvarig frafall kan forekomme, som ved omgangssykdom, men sannsynligheten for et permanent frafall kan anses som lav, ettersom alle er høyt motiverte til å fullføre oppgaven.</p>	<p>Høy</p> <p>Ved et permanent frafall vil arbeidsmengden måtte omdistribueres, samt arbeidsmengden øker betraktelig på resterende medlemmer.</p>	<p>Ta vare på samholdet i gruppen, vis hensyn til andres behov og perspektiv.</p>
4	Kortvarig frafall	Sykdom, uforutsigbare hendelser.	<p>Høy</p> <p>Det er å regne med at sykdom kan oppstå, samt det er mulig at uforutsigbare</p>	<p>Lav/Middels</p> <p>I et tilfelle hvor et gruppemedlem ikke kan arbeide kan det by på</p>	<p>Varsle tidligst mulig ved noen form for frafall, ha tydelig fordelte arbeidsoppgaver, slik at det</p>

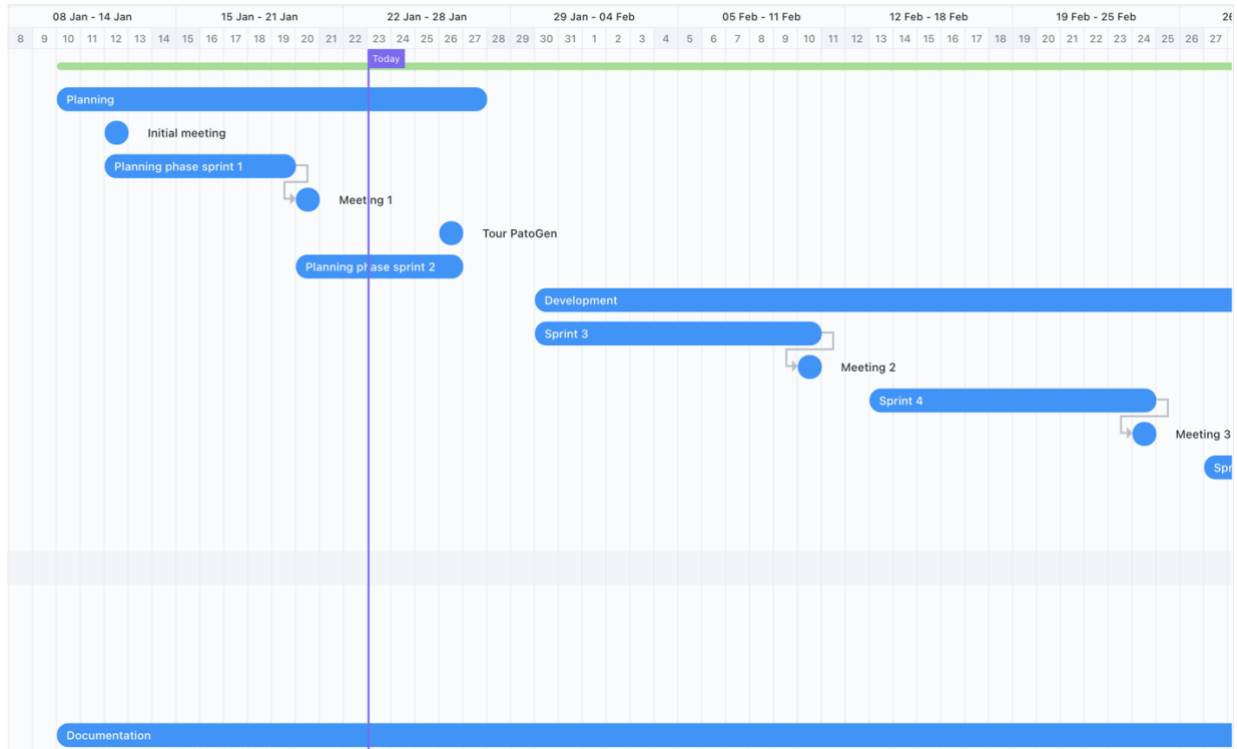
			hendelser som hindrer en persons oppmøte, kan oppstå.	skjevfordelt arbeidsmengde, og i verste fall føre til at en tidsfrist ikke blir holdt.	enkelt kan fordeles.
--	--	--	---	--	----------------------

## 6 Vedlegg

Følgende dokumenter leveres som separate filer ved innlevering i Blackboard i januar (obligatorisk arbeidskrav), men ikke i endelige leveransen av hovedrapporten den 20. mai!

### 6.1 Tidsplan

Confluence benyttes til å strukturere de forskjellige fasene i prosjektet. Ved hjelp av ulike verktøy som confluence og jira kan vi lettere få oversikt over møter og arbeid som foregår og/eller er planlagt. Disse planene strukturerer vi i tabeller og diagrammer som visualisere arbeidsprosessen ytterligere.



(Eksempelvis vil et gantt-diagram, her laget i «click-up» visualiserer arbeidsprosessen vår de neste ukene. Her er planleggingsfasen, sprinter og møter godt organisert og lagt ut i forhold til hverandre.)



# Pre Project plan (English)

## 1 Goals and scope

### 1.1 Orientation

The task "Risk Model and Web App" from PatoGen was the obvious choice for our bachelor's thesis. This is primarily because we want to create a simple, visual, and user-friendly website, but also because fish farming is an industry with interesting challenges and issues. Everyone in the group also wants to gain experience from the sea and fishing industry. This is because we are located in Ålesund and the industry has a large market share in the city.

This task was at the top of our list of possible bachelor projects, something we clearly stated in the project application where we expressed our skills and motivation for carrying out this task.

### 1.2 Issue / Project Description and Result Goals

The client PatoGen works with the utilization of farmed fish. They do this mainly by analyzing the disease course of the fish. Employees and customers of PatoGen see an increasing need for a user-friendly solution that provides a complete overview of the current disease and infection status of the fish farms.

Fish farming in Norway is divided into several geographical zones. These production areas follow the so-called "traffic light system", where each zone is assigned a red, green, or yellow light - representing whether the production area can increase or scale down production to maintain good fish health.

Our job is to visually present the current status and other relevant data for all geographical zones, so that customers and employees get a better overview of the condition of fish production in Norway, so they can make internal decisions to optimize their production. When the project is finished, it is expected to have developed a user interface that can easily be used by customers and employees, where they can get an overview of various risk factors for fish farming. The risks vary, but mainly bacteria, diseases, and lice. Users should be able to navigate on a map of Norway with all the fish farms, select a fish farm, and get an overview of risks based on the traffic light system. Users must be authenticated, and authenticated users should only have access to information of fish farming that belongs to their company.

## 1.3 Effect Goals

The group wants to at least satisfy the needs of the client and has ambitions to impress the client. In general, the group aims to perform at a high level.

The desirable long-term effect from the client's perspective is that their customers can easily analyze the condition of the fish in their fish farms. This will help customers make internal decisions and increase production efficiency. The government has estimated that a traffic light system for sea lice can provide an annual production growth of 23,000 tons per year. In our task, we will take into account not only sea lice but also other agents. This could lead to production growth increasing even more than what the government has estimated.

## 1.4 Frameworks

- Fixed group rooms on campus with board and TV screen.
- Coverage of parking expenses when meetings are to be held at PatoGen's offices.

## 2 Organization

The actors involved in the project are NTNU and PatoGen.

## 3 Implementation

### 3.1 Main Activities

Listing of main activities.

- Communication with the client and supervisor
  - The entire group communicates with the client and supervisor at regular meetings to discuss how things have gone, and what the way forward is.
- Design
  - Eirik Dahle is in charge of the design of the website.
- Version control and CI/CD
  - Github, Branching, Pull Requests, GitHub Actions for Continuous Integration.
- Development and report writing
  - All group members will contribute with code and report writing throughout the project. We will use React as a library.
  - Management and internal communication
- Jones is the team leader and is responsible for:
  - Communication between the supervisor, client, and group members.
  - Ensuring deadlines are met.
  - Ensuring that the quality of submitted work maintains a high standard in collaboration with Didrik.
  - Ensuring that the group's cohesion is good, that the work contract and tasks are followed.

- Statistics, data processing, representation/comparison of data
  - There are certain industry standards that must be followed, and challenges that require sharp professional competence - so the client will provide much assistance with data processing.
  - If additional statistical competence becomes necessary, then statistics students from NTNU can contribute.
  
- Scrum methodology
  - 2-week sprint, meeting with the supervisor every week in the initial phase, every other week after getting well into the development process.
  - Sprint planning
  - Daily standups
  - Sprint review
  - Sprint retrospective
  - Jira for work distribution and planning
  - Confluence for administrative tasks (documents, minutes, etc.)

## 3.2 Milestones

Listing of critical dates.

12/01/23 - First meeting with supervisor and client

27/01/23 - Submission of preliminary project plan

03/02/23 - Feedback on preliminary project plan, approval

21/04/23 - Oral presentation of the project in English

21/04/23 - Submission of report to supervisor for feedback

18/05/23 - Posters should be finished and ready for printing

22/05/23 - Submission of report and appendices in Inspira, likely project presentation.

## 4 Monitoring and Quality Assurance

### 4.1 Quality Assurance

To ensure that the quality of the project maintains a certain standard, it is important that we as a group read and analyze all work that is carried out and quality assure this before it is potentially presented to the client and supervisor. Regular communication with the supervisor will provide us with constructive feedback so that we can further improve the product.

**Quality assurance measures:**

- Sketch and plan work thoroughly
- Document meetings, code, and other general work
- Regular testing of the application and code samples
- Feedback from the client / supervisor
- Github actions for use of CI

## 4.2 Reporting

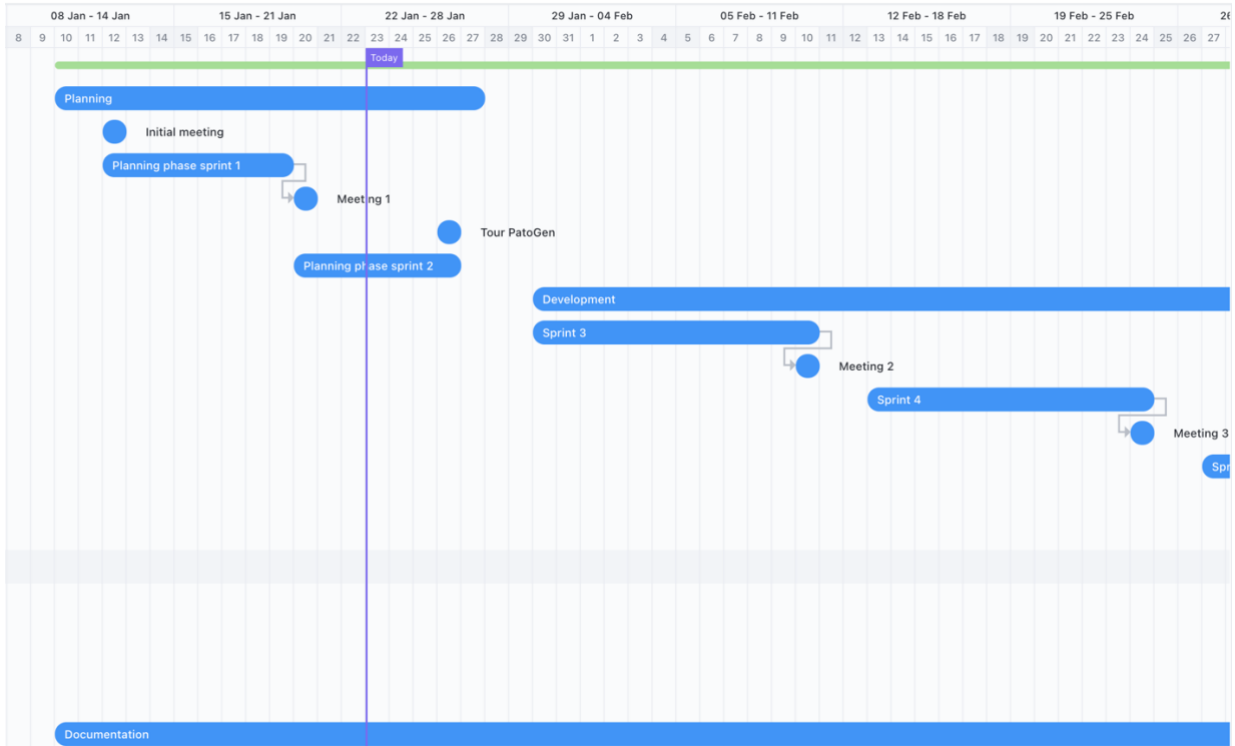
Reporting should be aimed for every other week to the supervisor and client, usually at the end of each sprint where the students meet with the supervisor and client.

## 5 Risk Assessment

<b>Event</b>	<b>Cause</b>	<b>Probability</b>	<b>Consequence</b>	<b>Preventive Measures</b>
Bugs	Coding errors	Very high	The application has unwanted behavior and can then be frustrating to use. In the worst case, parts or the entire application may be unusable.	Testing, Continuous Integration, documenting code, carefully evaluating how code and architecture work, version control.
Corrupted/lost code	Lack of version control	Low	Missing/corrupt code can, in the worst case, result in parts of/the entire project being lost/unusable.	Version control like git, preferably linked to a remote repository.
Unexpected permanent dropout, changes in group composition	Illness, internal conflicts, private reasons	Low	In the event of a permanent dropout, the workload will have to be redistributed, and the workload increases considerably for the remaining members.	Maintain group cohesion, be considerate of others' needs and perspectives.
Short-term dropout	Illness, unpredictable events	High	In a case where a group member cannot work, it can lead to unevenly distributed workload, and in the worst case, cause a deadline not to be met.	Notify as early as possible in case of any dropout, have clearly divided tasks so that they can be easily distributed.

## 6.1 Schedule

We use Confluence to structure the different phases of the project. With the help of various tools like Confluence and Jira, we can more easily get an overview of meetings and work that is taking place and/or planned. We structure these plans in tables and diagrams that further visualize the workflow.



(For instance, a Gantt chart, made here in "Click-Up", visualizes our work process for the upcoming weeks. Here, the planning phase, sprints, and meetings are well-organized and laid out in relation to each other.)

# Sequence Diagram

The sequence diagram displays the process of launching the application, a client/user need successful authentication with Microsoft azure active directory before being redirected to the application. Upon initial load, server requests queried data from the database to display when page loads. This process is the same between client-server-database when client/user request changes in both production area, diseases and/or date.

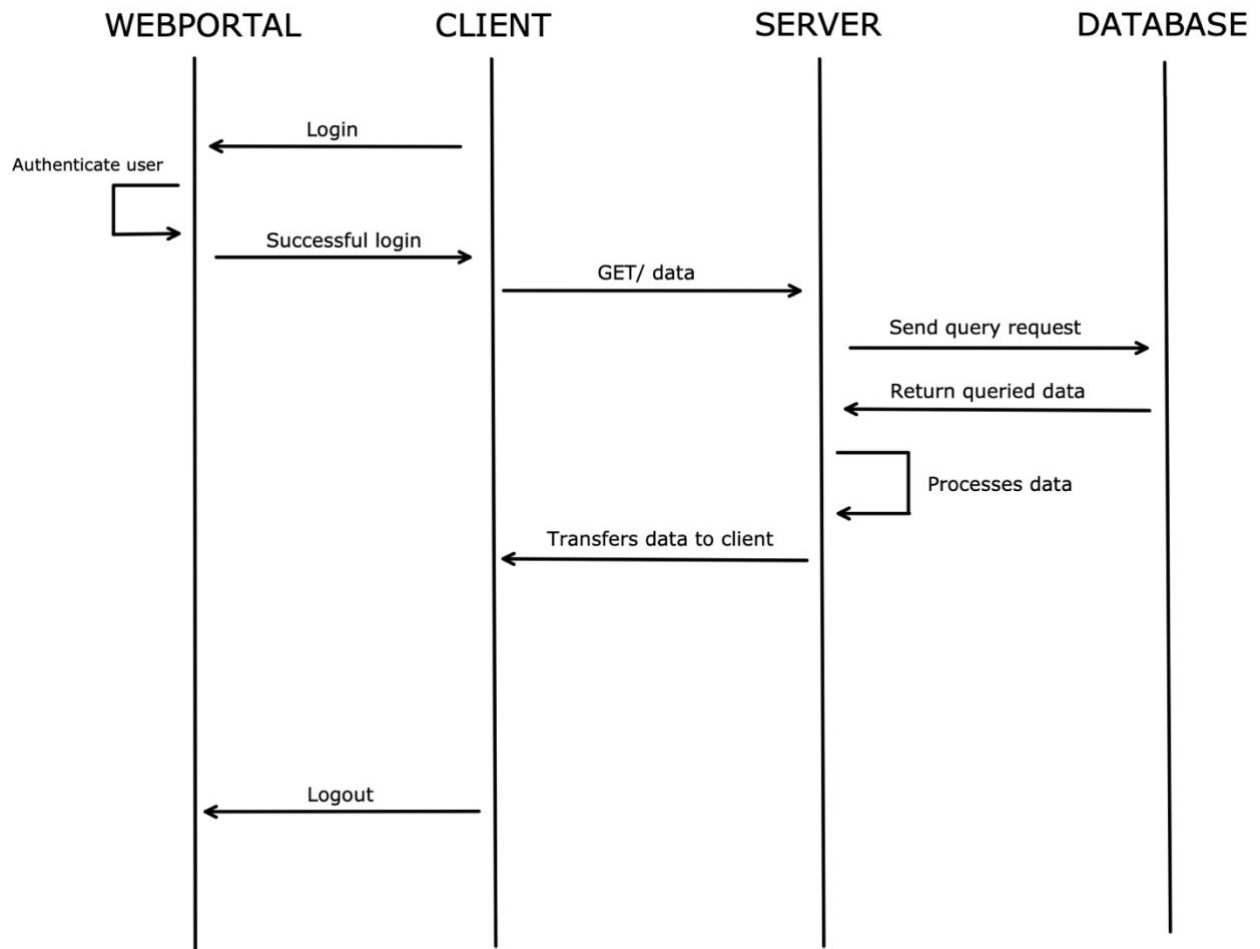


Figure 1 (Sequence diagram for loading application)

# Wireframes



Figure 2 (1. Iteration digital wireframe)

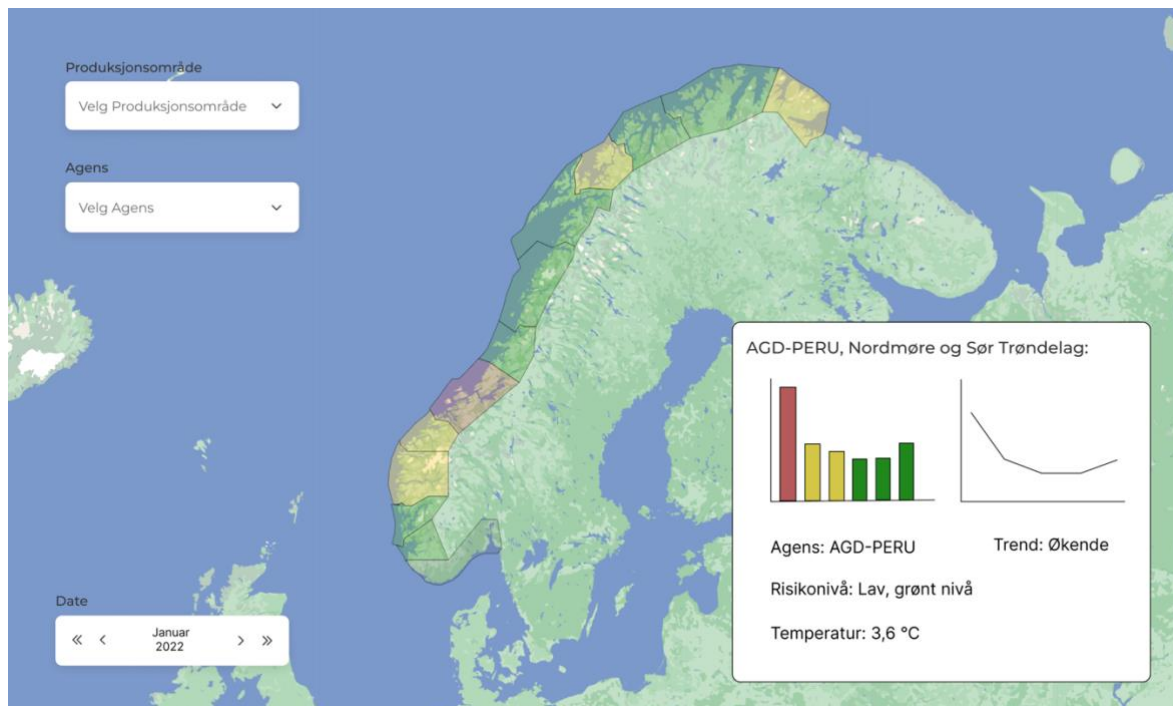


Figure 3 (2. iteration digital wireframe)



The wireframes shown below is the ones that were decided on during a team meeting including PatoGen. As the visualization methods and data representation was not declared, the side window was put on hold and would be picked up again during the later stages of development.



Figure 4 (Wireframe sidewindow collapsed)

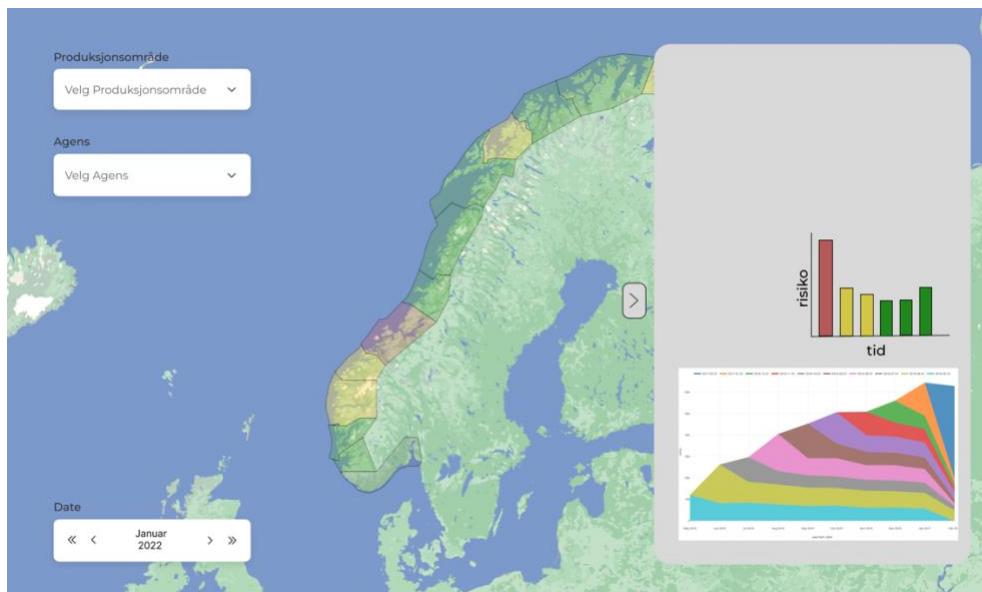


Figure 5 (Wireframe sidewindow opened)

# Application Result Overview



Figure 6 (Result sidebar collapsed)

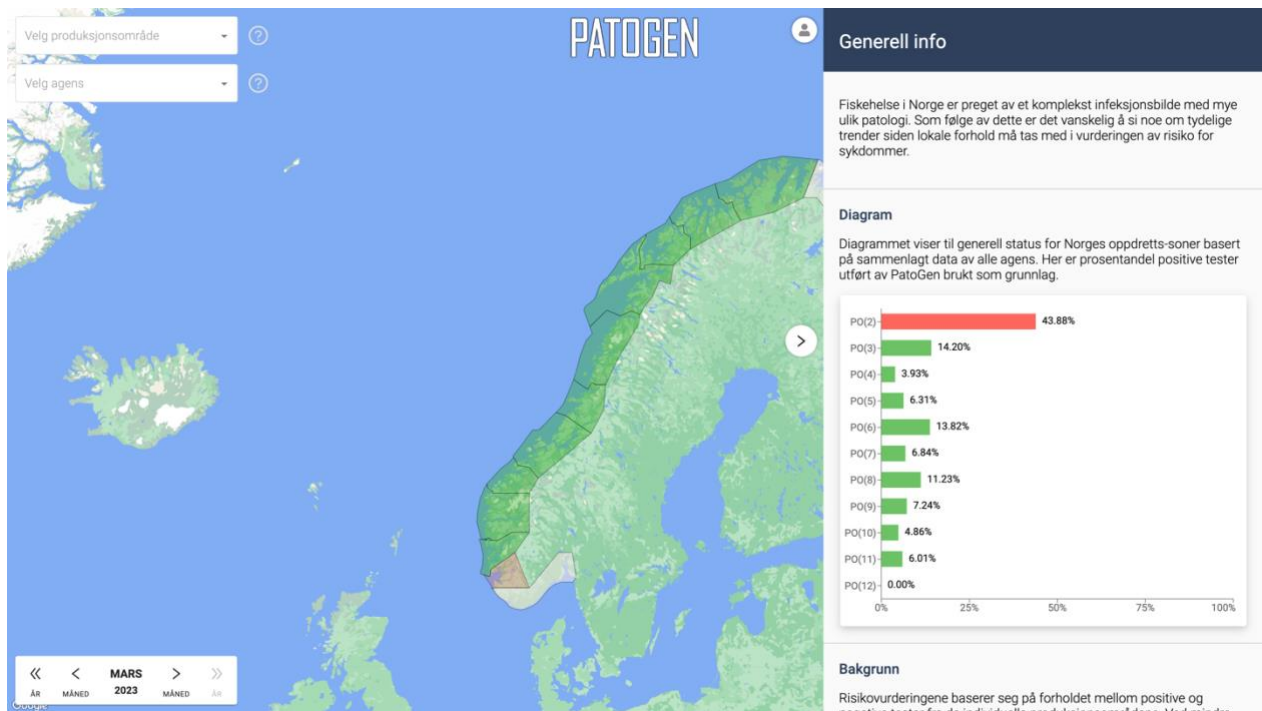


Figure 7 (Result sidebar opened)

