IT2901

Final report

10.05.2019

01 eCatch Kyst Pilot

Student name:	Student number:
Petter Grø Rein	766072
Balázs Orbán	257287
Håvard Bergheim Olsen	476709
Morten Falstad	258588
Tord Standnes	756451
Ebba Fingarsen	476762
Tore Stensaker Tefre	476757

Contents

1 Abstract

The eCatch Kyst Pilot project is the work of a group participating in the course IT2901 at the Department of Computer Science, at the Norwegian University of Science and Technology, in Trondheim, Norway.

The team has on behalf of their clients Dualog in Tromsø, Norway, and Bekk in Trondheim, Norway, developed a Progressive Web Application (PWA) that enables fishermen to report their catch to the Norwegian Directorate of Fisheries, through the systems of Dualog.

Figure 1: Lofoten Fishing Boats - Jonybraker/flickr.com - CC

Functionality-wise, the application comprises state-of-the-art features described in the PWA guidelines defined by Google [19], such as the possibility to use the application offline, and the possibility to create a shortcut to the application on the home screen of personal devices.

The report shows how the development team has been working with the product and the process. These are exciting subjects in themselves, and the team discusses aspects such as architecture, user experience, risk management and terminology throughout the report.

Working with the eCatch Kyst project has been enlightening. We have learned that fishermen is a quite varied demographic group, where astonishingly enough everyone of them travel to Lofoten during Spring, and that they have strong opinions on how things should be designed. This has fuelled discussion within the development team, documented in our report.

2 Introduction

2.1 The project

This project was written during the course IT2901 at the Norwegian University of Science and Technology, referred to as NTNU, in Trondheim, Norway. The team consisted of 7 Informatics students on their third year. The project had limited lifespan and resources, and roughly followed the spring semester at NTNU with project start at the 19th of January and a final delivery on the 10th of May.

2.2 The problem

Given Norway's position as the second largest seafood exporter in the world, fishing is a vital part of the nations economy [32]. A big reason for this success, and a prerequisite for even further growth, is that Norway focuses on managing the marine natural resources in a sustainable way and keeping the productive sea areas well maintained. One way to control this is by having quotas for fishing[35].

Today every fishing vessel over 15 meters has to report the catch to the Directorate of Fisheries[30]. In the future however, this might be expanded to include boats under 15 meters of length. This brings new challenges as these boats have fewer workers on board who already have a lot of daily tasks. If the catch is not reported correctly they risk fines or even facing imprisonment. This means that it is very important that an application for reporting the catch guides the user through the reporting process and protects the user from making mistakes.

The core technical problem we face in this project is delivering 3 messages to the Directorate of Fisheries with full integrity.

2.3 Introduction to core concepts

2.3.1 Progressive Web Application (PWA)

Progressive Web Applications, referred to as "PWA", are web applications built to meet a specific set of criteria[19]. The criteria is set by Google Developers as of now, with more actors like Mozilla and Microsoft, starting to adopt and develop their own PWA criteria. The idea is that the web application should behave and feel like a native application. The most important criteria is the ability to interact quickly and reliably with the application,

even when the internet connection is bad or fully absent. Furthermore, the application should provide a way to add a shortcut to the application on mobile devices. When using this shortcut, the application should open in full-screen and support the use of push notifications. This way the user can interact with the application like any other mobile application.

2.3.2 User experience (UX)

The User experience, abbreviated "UX", is a term that in this project describes how the product feels for the end user. It involves every aspect of the product, from how the user experiences the daily use of the application, to how potential errors are handled.

The International Organization for Standardization defines UX as

> *A person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service*

in ISO 9241-110:2006 [40]. However, we have based our use of the term on the words of interaction design professor Don Norman[8] describes UX as

> *User experience encompasses all aspects of the end-user's interaction with the company, its services, and its products.*

as we feel it gives a better interpretation of the term.

2.3.3 User interface (UI)

User interface is a broad term that involves everything from hardware to software, which in software development focuses on making the graphical user interface as user-friendly as possible. In this project, we wanted to pursue the best compromise between functionality and simplicity.

In ISO 9241-110:2016 [41], User interface is defined as

> *All components of an interactive system (software or hardware) that provide information and controls for the user to accomplish specific tasks with the interactive system.*

2.4 Stakeholders

The primary parties with interests in this project, referred to as stakeholders, are briefly described as:

- Bekk, our main customer.
- Dualog, the client that ordered the project from Bekk.
- The Norwegian Directorate of Fisheries, hereby referred to as the Directorate, that Dualog provides the service for.
- The fishermen, both Norwegian and foreign, the end users of the application.
- The Norwegian coast guard, that uses the information provided by the application to run inspections on the fishing vessels.

Within Dualog there are also multiple departments that can be listed as stakeholders. These include the sales department, that needs to market the application, as well as tech support that will need to maintain and further develop the application.

The users of the product are fishermen on smaller fishing boats. These fishermen are usually a bit older [2] and reporting on a digital device can therefore be a challenge they feel reluctant to accept.

Statistics from the Directorate, informed us about the presence of 5564 fishing boats under 15 meters of length in February 2019 [15]. This means the application had to fulfill a diversity of expectations and not only be catered towards one generation or group. This required the solution to be intuitive, yet complex. By this we mean it should be simple enough to be easily understood by someone with little technical background, while still implementing all the sought features beyond just the basic functionality.

2.5 The solution

eCatch Kyst Pilot is a solution that makes it possible for fishermen to easily and intuitively report their catch from whatever device they prefer, ranging from smart phones to computers. This is a large improvement from similar solutions for larger vessels of today, which often require specific hardware and online connectivity. Persistent online connectivity is difficult because of limited coverage out at sea and may be expensive. With all of this in mind we developed a solution that fulfill the PWA criteria which corresponds with the needs the fishermen have at sea.

2.5.1 The wanted impact of the solution

Our goal for the application was to simplify the report of the catch and make it easy to implement as a part of the daily routine. If we could make the reporting easier, the fishermen would feel more comfortable that they are following laws and be less opposed to reporting the catch.

The application would also make upholding of the law easier for the coast guard. They would more easily get an overview over the fish currently onboard the vessel through the application. As this process is quicker, the coast guards could get more time for other assignments.

The Directorate would get more thorough data about fishing in Norwegian waters by including smaller fishing boats. They have already proposed a plan for when the different subcategories of smaller fishing boats are required to start reporting[36]. With our application they will get the statistical information they need. This would simplify the task of monitoring the marine life in Norway.

2.6 Assignment description

The project evolved around creating a brand new React web application, that fishermen on fishing boats shorter than 15 meters could use to report the catch at sea. Our main task when creating this web application was to provide new perspectives and approaches to already solved problems, as the customer already develops a similar application for vessels longer than 15 meters.

As the customer aimed to utilize the ideas from our product, and not necessarily the application itself, aspects like following the new General Data Protection Regulation was not of great importance. However, we needed to design the application with reliability, maintainability and usability in mind as discussed in section 3.4.

3 Product

3.1 Preliminary study

In order for the team to better understand the problem, preliminary studies were conducted to gather information about the problem and the solutions available. The focus was put on architecture, user interviews, security and available technology.

3.1.1 Study of technologies

Before we started our prestudy on technologies, our customer had some wishes for what technologies we should use during development. They expressed a desire that we used React as the frontend framework. For the backend we were entirely free to choose what we thought was optimal for the project, as the backend framework was not a priority for the customer.

Considering the customer wanted an application that fits the PWA criteria, it was wise to choose frameworks that would help us accomplish this.

The frontend framework needed to work while offline, and we solved this with a Service Worker [3] (More about this in 3.5.1). Right now the top 3 frameworks for web development are React (A library but considered by many as a framework), Angular and Vue [38]. All of them have support for Service Worker as Service Worker is only a JavaScript function. React and Angular have built-in tools for implementing a worker where as in Vue you would have to implement it to the project yourself.

From a PWA perspective the only requirement is that the website supports HTTPS. There are multiple ways of implementing a backend, e.g. cloud services or a self-hosted server. We chose a cloud service as it provides us with more functionality than having a self-hosted server, but without concerns about scalability, reliability or cost. Scalability is provided from a few test users to a fully deployed product with thousands of concurrent users without having to upgrade the hardware. It also provided us with reliability in regards to up-time, with the cloud provider maintaining the servers, and it enabled us to focus on the product itself.

Some of the biggest cloud services are Amazon Web Services, Microsoft Azure and Google Cloud [7]. Google is currently contributing to the PWA standards, so we looked at their service, Firebase [12]. It hosts the website with HTTPS and provides the required functionality through Firestore and Cloud

Functions and authentication, as well as a persistent local storage. The SDK fully supports the JavaScript language, the same as React JS. This keeps the technology stack simple.

3.1.2 Business context

Businesses have different ambitions and motivations. In this project we performed a prestudy of our customer's business goals. With this information we were able to make better decisions, and prioritize our effort spent on the most important goals. We wanted to specifically use this information to determine architectural requirements and map the possible security risks. The business context was filled out by interviewing the customer.

Below are the result of the business context analysis, the full version with business goals and business assets can be found in Appendix C.

BG1	Uphold legally required electronic fishing reporting for professional fishermen, fishing fleets and other actors in the marine coastal environment.
BG2	Securing and regulating the fishing resources in the ocean, through reporting and registration of fishing operations.
BG3	Make sure Norwegian ocean resources stays in the hands of the Norwegian people, for now and for future generations.
BG4	Extend the product line to include a product for smaller boats, shorter than 15 meters.
BG5	Create valuable software for fishermen, and continuously simplify the reporting process.
BG6	Have a reputation of making sustainable and reliable products, that are always available when needed.
BG7	Have a pleasant user experience with the use of our products, that promotes reliability and usability.
BG8	Reduce support cases.
BG9	Reduce product maintenance costs.
BG10	Expand into a new market.
BG11	Reduce technical debt.
BG12	Meet the needs and expectation of our stakeholders.
BG13	Meet business agreements with the authorities.
BG14	Become the preferred supplier of electronic catch reporting.
BG15	Become the market leader with a 70% market share.

3.1.3 The effect of the prestudies

Our preliminary studies were fundamental for many of the choices we conducted throughout the development phase. As the application got further developed and more complex, the prestudies guided us through problems and obstacles we encountered during development.

The prestudies were also of great importance to discover where our focus should be and what we could reduce the prioritization of. An example of a feature that reduced in prioritization, and was ultimately not fulfilled, was the introduction of maps. This was because it could possibly require a lot of development hours, and was not a vital part of the basic functionality of the application.

3.2 Requirements

The requirements stated below were gathered in different ways. Some requirements came directly from the customers while others were defined by the team members as a suggestion to what would be required and discussed with the customer at meetings. The abbreviations L, M, H denotes Low, Medium, High respectively.

3.2.1 Functional requirements

ID	Description	Priority
FR1	The messages needs to be signed to maintain integrity	M
FR2	The solution needs to be able to send messages several times if needed to make sure the messages are sent even if the network may be unstable.	H
FR3	The solution needs to handle an array of edge cases based on equipment like crab-fishing, setting and fetching of fishing nets and overnight stays.	L
FR4	The solution needs to send out descriptive errors to relevant actors.	M
FR5	The solution should integrate Automated Identification System (AIS) tracking to view the GPS location of the ship.	L
FR6	It should be possible to edit the catch-messages after they are created/sent.	M
FR7	The solution should propose standard settings to ease use.	H
FR8	The solution should provide a dark mode option.	H

FR9	The solution should provide a full overview over the catch currently on board.	M
FR10	The solution should provide all relevant choices for messages such as types of fish, weight and more.	H
FR11	The solution should provide an overview of previously sent messages.	H

3.2.2 Non-functional requirements

ID	**Description**	**Priority**
NFR1	The solution should be a PWA.	H
NFR2	The application should be Dualog themed (design, colors).	L
NFR3	The solution should work on all devices (cross-browser and cross-platform).	H
NFR4	The solution should provide support for multiple languages.	H
NFR5	The application should aim to generate little data traffic as an internet connection can be costly.	M
NFR6	The solution can be used without requiring a lot of attention and focus.	M
NFR7	The solution should help the fishermen report correct and on time.	H
NFR8	The solution should provide a guarantee that the captain has provided adequate information in his reports, and that does not risk breaking any laws.	H
NFR9	The solution use words, logic and symbolism that the user is familiar with and understands.	H
NFR10	The solution should be usable with bigger hands.	M
NFR11	The solution should guide the user as much as possible as the user may have little to no technical background.	H

3.2.3 Architecture significant requirements (ASR)

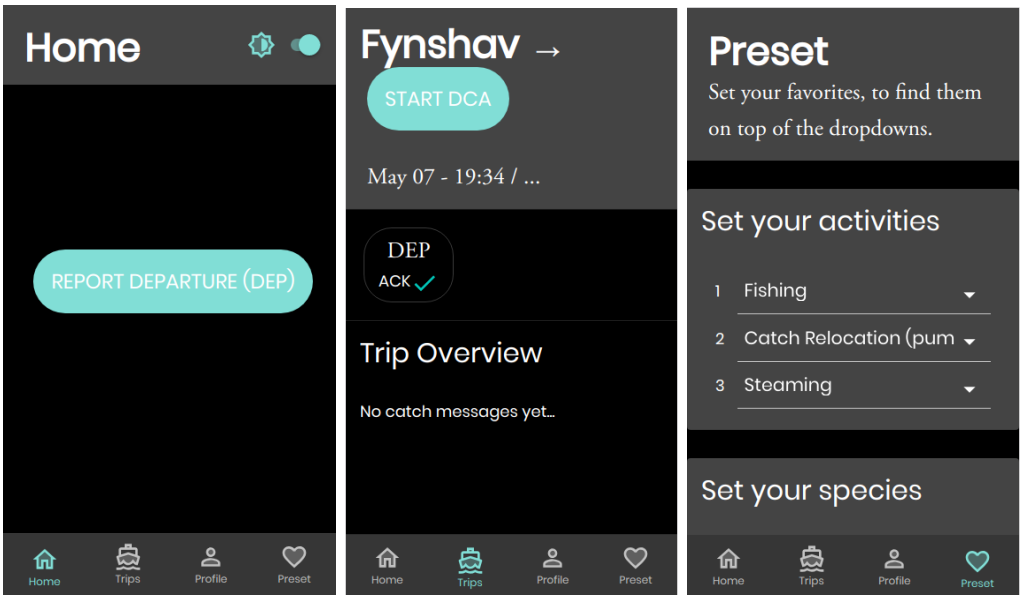
ASR are requirements that affect the design of the architecture and are used to verify the architecture of the application. See the process section for more information.

ID	Extracted from	Description
1	Requirements documents	The fishermen should be able to use the application and all it is functionality offline.
2	Requirements documents	The application needs to handle the sending of messages, and give user feedback.
3	Requirements documents	The fishermen should be able to get the GPS location in the application.
4	Requirements documents	The messages sent to the endpoint needs to be formatted correctly.
5	Requirements documents	The messages send to endpoint needs to be authenticated, to validate the correct user.
6	Requirements documents	The fishermen should be able to use the application on all devices he owns.
7	Requirements documents	The fishermen can send his messages without thinking about signing the message.
8	Requirements documents	The fishermen should be able to store information locally on the device.
9	Requirements documents	The fishermen should only have to log on one time.
10	Requirements documents	The fishermen should not have to use a lot of MB using the application.
11	Requirements documents	Dualog needs to get sensible information from error messages.
12	Requirements documents	The technicians should be receive bug reports from the app.
13	Requirements documents	The fishermen should be able to view himself using the AIS systems.
14	Stakeholder meetings	The application needs to be a progressive web application (PWA).
15	Stakeholder meetings	The key for authentication is stored by Dualog and should be provided at login.
16	Stakeholder meetings	The main focus of the project is frontend development.
17	Stakeholder meetings	The frontend application needs to be portable to Dualogs servers and services.

18	Stakeholder meetings	The user group needs to be able to use and understand the application with little effort.
19	Stakeholder meetings	The customer wants to focus on high reliability.
20	Stakeholder meetings	The customer strongly suggest using React.
21	Stakeholder meetings	The customer suggest using client-server architecture.
22	Business context	Uphold legally required electronic fishing reporting for professional fishermen, fishing fleets and other actors in the marine coastal environment.
23	Business context	Securing and regulating the fishing resources in the ocean, through reporting and registration of fishing operations.
24	Business context	The user should not be able to make mistakes in the reporting process.
25	Business context	Have a reputation of making sustainable and reliable products, that's always available when needed.
26	Business context	The reporting process should be simple to complete, without confusing interfaces, and easy to learn.
27	Business context	The application should be self explanatory to reduce support cases.
28	Business context	Reduce product maintenance costs.
29	Business context	Reduce technical debt.

3.3 Design and UX

Given the proposed design in appendix D we came up with this design in our final product.



These are the three main pages the users interact with, as they use the application. There are elements with big buttons and reduced opportunity for errors by displaying only the possible choices from appendix D.1, the finite state machine mechanism and possibility to add your own options as a preset from appendix D.2.

3.3.1 User group description

The expected user group is predominantly men in the age group 20 - 65[2]. It was desired from Dualog to assume that this group did not want to have anything to do with our product. The design of the application therefore needed to have this challenge in mind.

Our design was heavily influenced by our diverse target group, comprising of fishermen of all ages and technological competencies. This can be seen for example through our simplistic user interface and our choice of larger text and buttons. One of the biggest focus areas when designing for groups with

lower technical experience was making sure everything in the application is consistent. We tried being consistent in regards to having similar colors on buttons with the same functionality. This would give the user an indication that each time similarly colored buttons appear, they would have similar actions or have similar level of importance. To solve this the application utilizes two colors, a primary and secondary button color.

Figure 2: Primary and secondary button color

The user gets feedback through a series of different messages depending on what they do in the application, as a way of telling them that their action did something. When these actions fail or they try to do something wrong, they get a red warning explaining what went wrong.

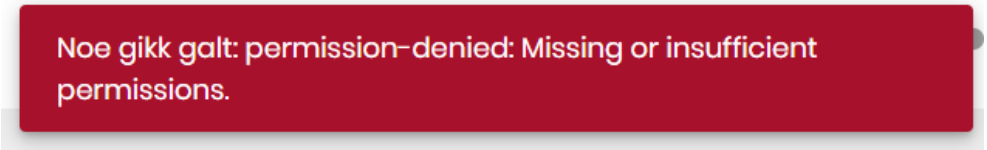


Figure 3: Example of error message to the user

3.3.2 Design description

Navigation

An important focus in the application was to minimalize the possibility for user errors, and ensure that all required user input is present in all messages. To accomplish this, the design focused on limiting the possibilities and choices the user had where necessary.

An example would be that the user is not able to send a catch message (DCA) or report to harbour (POR), unless they have sent a prior departure message (DEP). The options to send one or more DCAs and a POR would only be visible after a trip had been started with a DEP message.

Messages
When creating a message, most of the fields are filled in by the application. It fetches data from previous messages or preset values. This is to help the user limit the time it takes to carry out the operations required, as well as help find the correct input based on previously given information.

Prior to first departure the fisherman can preset most of the required information. When out at sea the fisherman will have to perform minimal amounts of clicks and has easy access to his favorite choices. After sending a message, the following messages will be based on the information from the previous ones of equal type. This is an advantage as the fishermen usually have routines and for instance returns to the same port every trip.

After the message is sent, the fisherman gets feedback from the system if the message is pending, acknowledged or rejected. This is to make sure the status of the message is clear for the fisher, which in turn could make them more informed and trusting of the application.

We analyzed the messages and the required fields of required, as seen in appendix F. Some of the fields were generated, some come from the backend (when a admin creates a user), some could be preset and the rest were set by the user. From the rather heavy raw messages, we now had a fast and easy way to generate full messages without extensive user interaction.

User guidance and error protection
All input fields have a describing label showing the user what type of input it expects. If the user tries to input something on a wrong format, the application will provide useful feedback that explains what went wrong and how to correct this mistake.

3.3.3 Design principles of implementation

The design methodology we decided to use during and after the mock-up and development phase, was an agile approach to design thinking, where we continuously assessed and improved the design. This allowed us to better adjust to new features and change, as well as providing the customer with the opportunity to give feedback all the way through development. Furthermore, establishing and using design principles are essential for developing a good product. The principles we utilized were Don Norman design principles and The Gestalt principles.

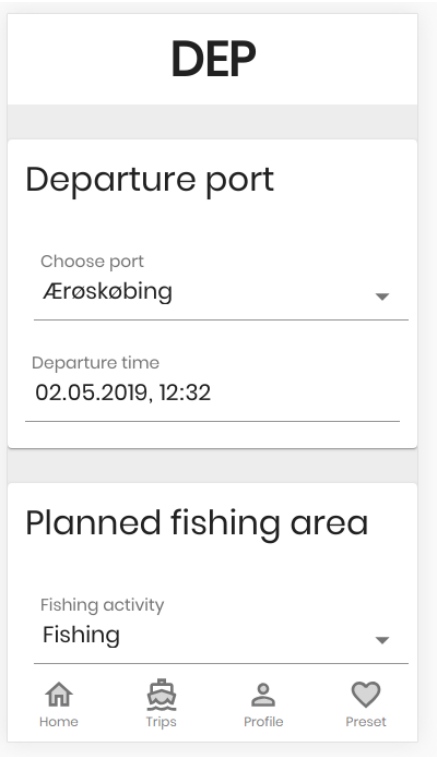


Figure 4: Example grouping

The way we implemented these ideas and principles into our application can be seen in the way similar elements in the forms are grouped together following the gestalt principle of proximity. The icons follow Don Normans principle of mapping elements to their real world equivalent, and the colors on buttons and actions are the same throughout the application. This helps the user increase the rate at which they learn to use the application. As we can see in figure 3, the feedback provided in the application follows Don Normans principle of feedback, as actions give immediate feedback. The feedback is presented to the user in the form of pop-up messages and changing icons when sending a message.

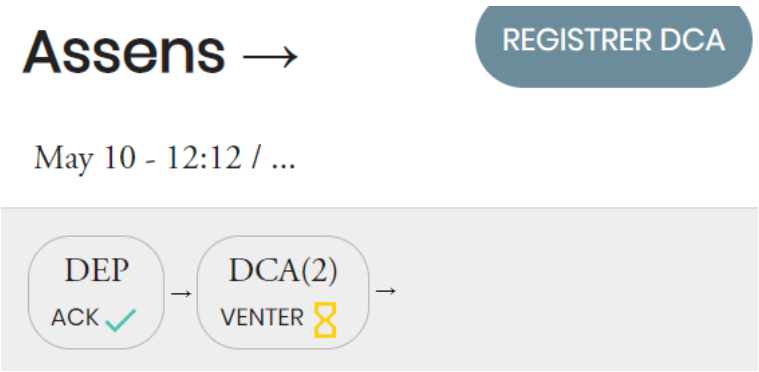


Figure 5: Example of icons changing in the application

One of the huge pitfalls we could, and might still have fallen into, is designing for the proposed user group. Even though we have tested the icons and design on some test persons, they might not reflect the entirety of the user group and might have more or less technological insight than the average user.

As we developed the application, choices were made to alter the proposed mock from Figma, as we saw that it gave the application something positive.

An example of this is the rounding of the buttons and use of a bottom navigation bar. In the three messages we send with the application, we apply the law of proximity to group menus that are related to each other, time, fish etc, so that the user can relate. It improves the general feel of the application and user experience.

3.4 Architecture

One can never achieve the perfect software, because there are trade offs to every architecture. The qualities desired in software, affects the way it is written and designed. A software focused on performance is very different to a software focused on testability.

The quality attributes we have focused on are Reliability, Usability and Maintainability [27]. These attributes align with our customers wishes, our project scope and the team's goals.

3.4.1 Architectural patterns

Our software system was created using a client-server pattern. The PWA React frontend is on the client and the "server" in this case is, the serverless services that Firebase provide. It is in this environment we have developed the product.

The whole ecosystem this application exists in can be seen as a multi-tier pattern. As you can see in the physical view 9. We have multiple clients connecting to our backend which in turn sends "correctly" formatted messages to the customers endpoint, which in turn sends it to the endpoint of the Directorate. We do not know what happens with the messages in the last two tiers of the system, but we receive responses.

Reliability is the main reason we added our own tier into the ecosystem. The only way to develop and test with reliability in mind is to have a complete backend. With the backend we have one source of truth as clients can have unreliable connection and transactions far out at sea.

The backend can also be seen as a small, shared data structure as both clients and other applications can use it as an access point.

3.4.2 Architectural tactics

The architectural tactics we have implemented origins from the book "Software Architecture in Practice" [34].

Reliability
Reliability refers to a property of software that is available when needed [34].

Tactics

- Use timestamp on messages
- Exception handling
- Continuous re-sending messages with the use of PWA in offline state
- Active redundancy with the use of Firebase

Usability

> *Usability is concerned with how easy it is for the user to accomplish a desired task, as well as the kind of support the system provides to the user [34, p. 198].*

Tactics

- Separating the user interface from the rest of the system. We do this with the component based structure in React. Where the interface and data is separated in different components.
- Automatic maintaining of models, task, user and system. Which in layman's terms means, do not let the user perform actions that produces errors in the operations they are executing. This is mainly described in the UX portion of the report.

Maintainability Therefore we want to make the system easy to change and further develop over time.

Tactics

- Reduce components
- Split components
- Increase cohesion
- Reduce coupling
- Encapsulate code
- Abstract common services

3.4.3 Views - the 4+1 model

All the views here are based on the 4+1 view model [22]. The views represent different models of the software, based on different needs. The 4+1 view model is a collection of the most relevant and complete views. It focuses on representing useful information and the notations reflect this. Therefore there are multiple notations used in the views, these include derivatives of the UML and the Booch notation [22].

Logical view
This is a high abstraction logic view. We chose to go with a high abstraction

as it was more efficient at communicating the logic of the system. The logical view uses a notation derived from the Booch notation, as described in 4+1 model [22]. The entities represent a class category and the dot arrow represents the use of a class (entity "with" dot uses the class it points to).

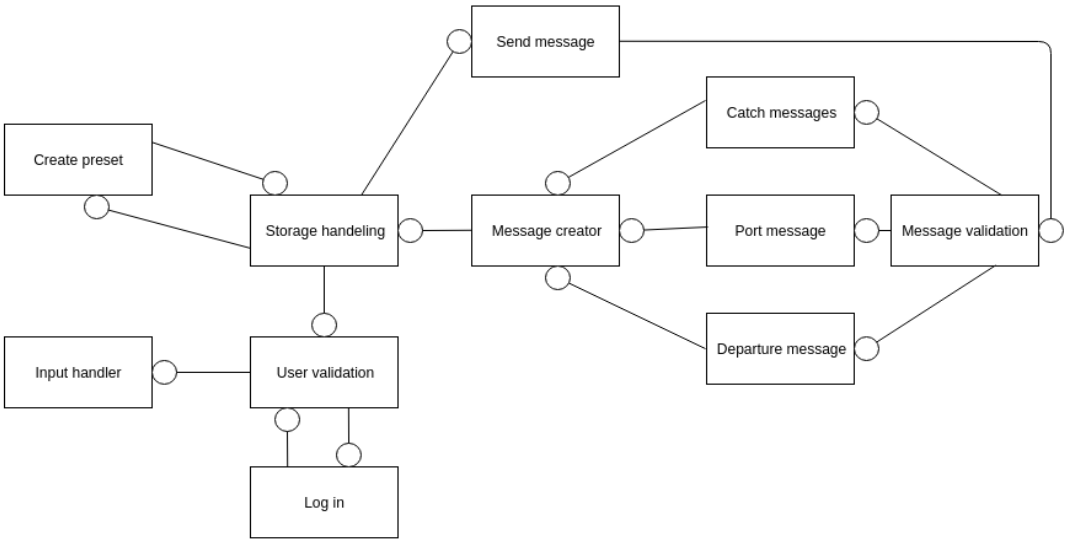


Figure 6: Logical view

Development view
The development view is a layered view, representing the development environment. This is made to be used by future developers. The layers are on the left side.

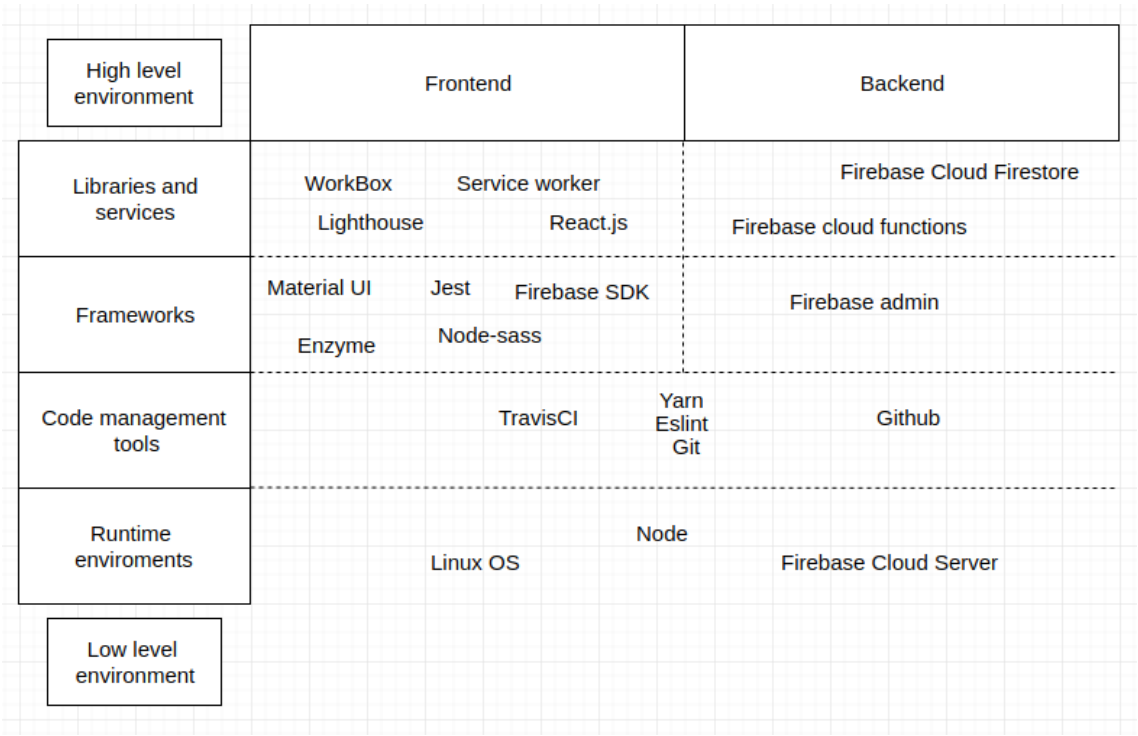


Figure 7: Development view

Process view:
The process view represents all the code in runtime, and all the code it interacts with. It uses a derivation of Booch notations to represent the processes, described in the 4+1 view model [22].

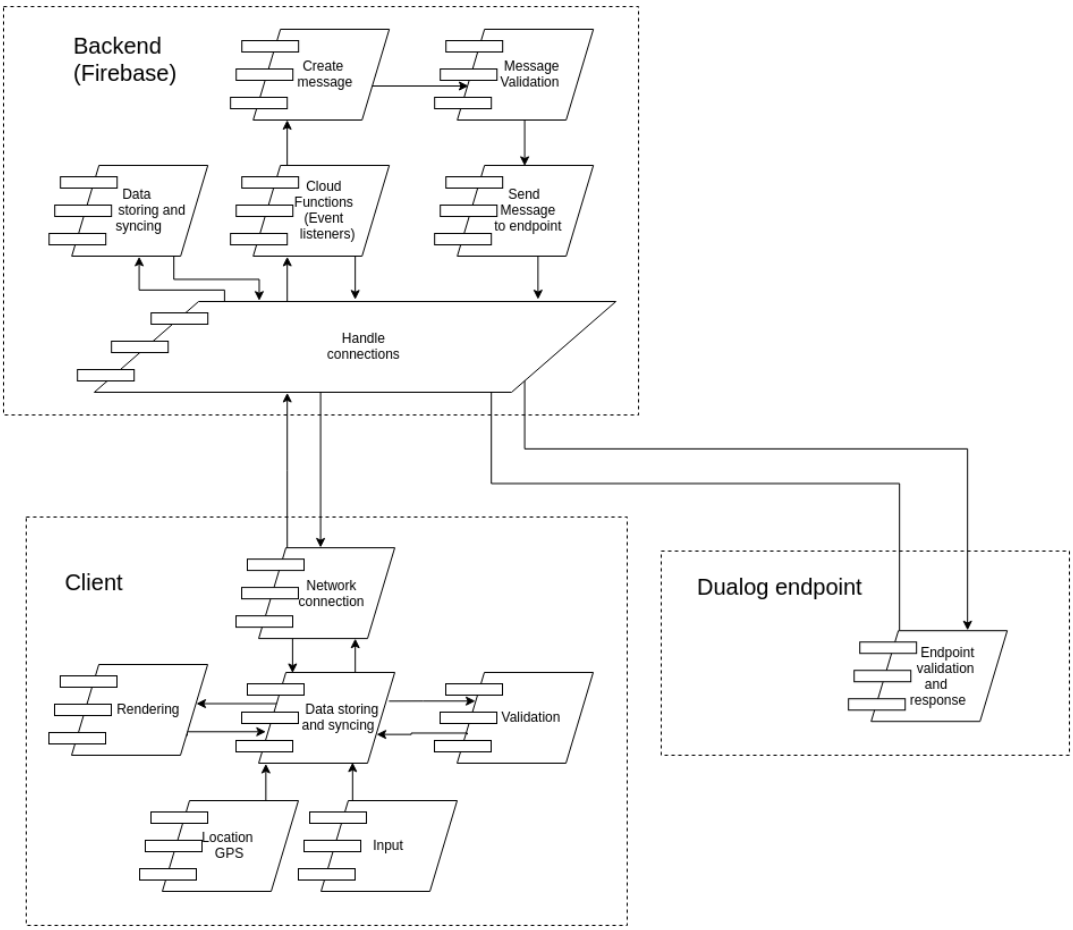


Figure 8: Process view

Physical view
The physical view represent the hardware involved and the connection between them.

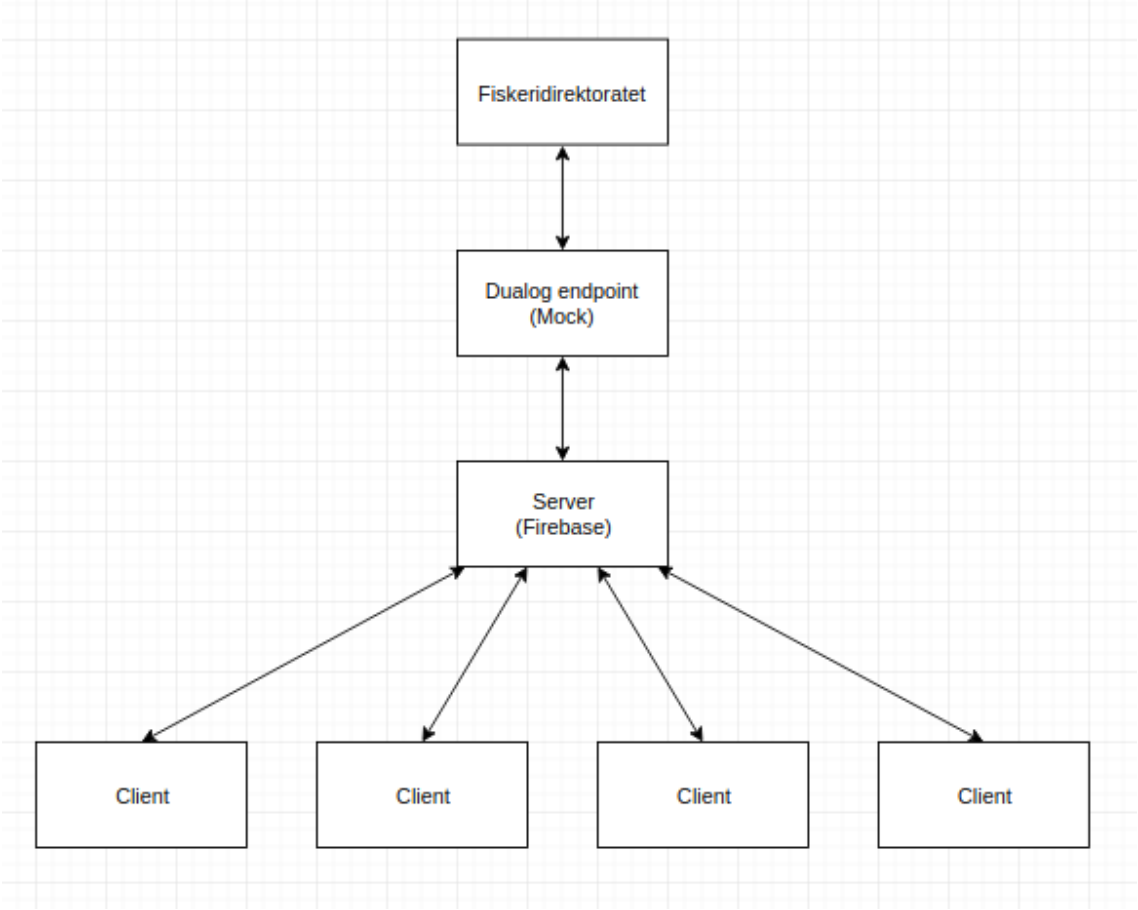


Figure 9: Physical view

Scenarios
Here are two use scenarios of the application. The one on the left shows a person sending a message, and on the right someone edits a preset tool.

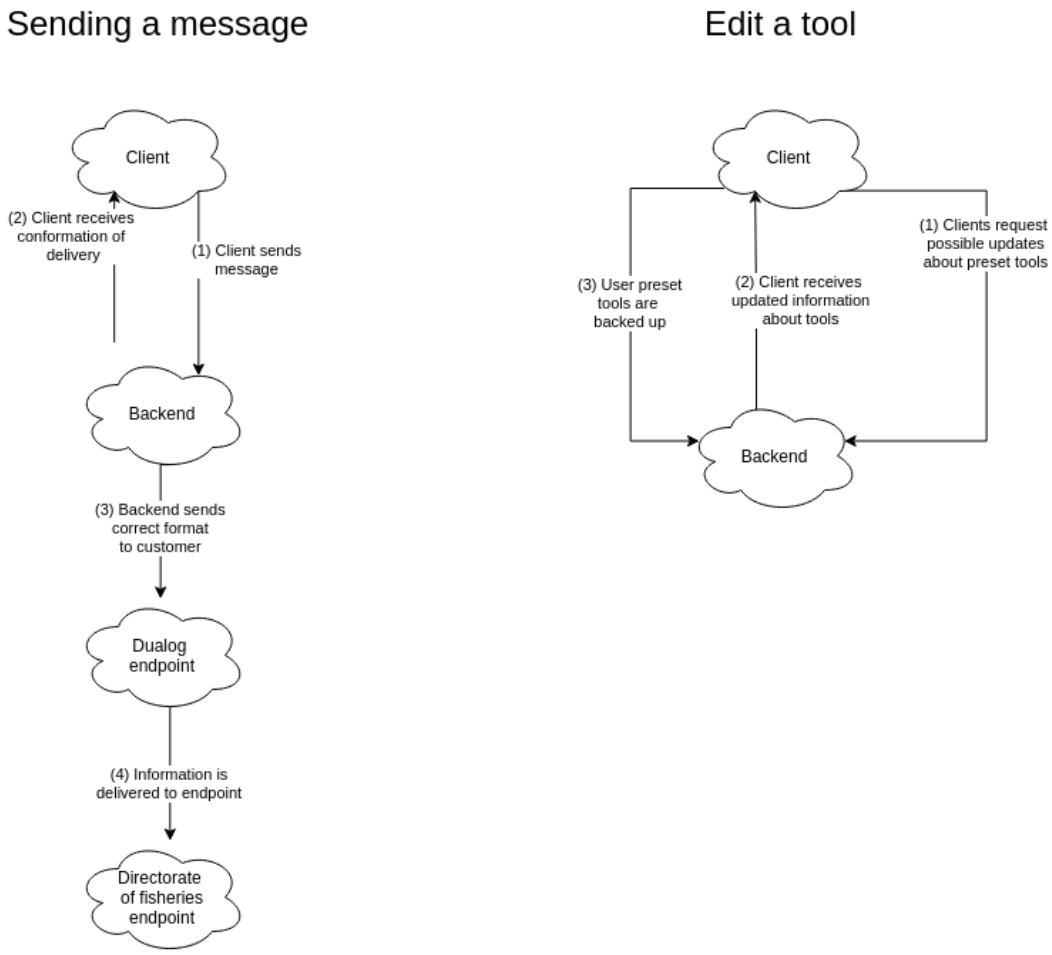


Figure 10: Scenarios

3.5 Implementation

3.5.1 Technology stack

The desired programming language was JavaScript, as we were required to create a Progressive Web Application. For the JavaScript syntax we used the modern ES6 syntax [11]. We were asked to use React JS [28] as our JavaScript library for the customer, and from our prestudy we found React JS to fit our needs. For code documentation JSDoc [21] was used if needed.

For the backend, we went with Google Firebase [12], which works very well with PWA (for example, because of its offline capabilities [14]), and provides many extra features for such a project. For example, the authentication part is totally out of the way, as Firebase did all that for us in the background (meaning it turned the authentication process into a "one liner").
It also featured a hosting solution, which helped us to set up a main page and a staging site. Meaning, that features under current development could be deployed to a different URL than the finished site. This gave us an extra layer of security in terms of users getting errors or bumping into bugs while using the application.

Our database is hosted on Firebase's Cloud Firestore [13] which is a scalable NoSQL cloud database. Backend code for validation of messages is hosted on Firebase's Cloud Functions [4]. Cloud Functions is an event-driven, serverless computing platform that triggers when changes are made in on our Firestore.

For the UI elements of the application the Material-UI library was used [26]. Material-UI is React components that implements Google's Material Design. The library provides us with UI-elements that are tested extensively and work on all browsers. Using an existing styling library gives us more time to focus on the code.

For code sharing and developing we used GitHub, as discussed later in section 3.5.2. This was a tool the entire group has used before, which made collaborating a lot easier. We registered our application under an organization, to be able to separate the codebase into two sections, a frontend and a backend.

The master and develop branches were locked for direct committing, the only exception being the repository owner. Code could only be merged into these branches through Pull Requests, often referred to as PRs, that required to go through our testing pipeline. The pull requests then had to be reviewed by other team members. If accepted, and all the tests from the pipeline had

been successfully ran, the code would be merged into the develop or master branch. Merging into the master branch required two approvals from other team members, while the develop branch needed one. By approving changes, we ensured a higher quality and more ownership of the code for the whole team.

A guideline for contributions was set up in the repository's root so that anyone could easily start contributing in a manner that is consistent with the rest of the project. All this was to ensure not only a high quality end product, but to make it easy for future contributors to build upon what we have done.

3.5.2 Code repository

The git code repository is currently hosted publicly by GitHub, and the team is pursuing to follow GitFlow [17] for usage of git. We did one change to the flow of GitFlow, instead of making release branches we continuously tested on the develop branch and merged the release into master when every bug was fixed and we were finished with a Sprint. After every sprint we tagged the release with a version number following Semantic Versioning [33]

We also utilized underlying features of GitHub such as issues, boards, code reviews and pull requests. Some members also benefited from the software GitKraken [18] to ease this work.

3.5.3 Code editor

We decided that throughout this project every member should use the same editor to simplify cooperation. The choice fell on the open source editor Visual Studio Code, commonly referred to as VSCode. The reason being it features an extensive ecosystem of extensions that could help the development process. To improve the structure and visual appearance of the code, we used ESLint for linting. This was done both locally in VSCode and in the pipelines.

3.6 Testing

For testing, we used Jest [20], Enzyme [10], Cypress [6] and Lighthouse [23]. The first two were used for unit testing, Cypress for end-to-end testing, while the latter ensured that our application met the requirements of a PWA.

3.6.1 Pipeline and testing

Our pipeline was built using Travis CI [39]. It took care of our integration and deployment together with the Google solution Firebase Hosting. The code was linted using ESLint and checked during the Travis CI build. This was to enforce a common code style.

When pushing code to the repository, each commit triggered a Travis CI build, which ran our tests, and uploaded a coverage report to CodeCov [5]. If everything went fine, and the trigger came from the develop branch or a Pull Request to the develop branch, the code was then deployed to our staging site. If the trigger came from the master branch, the code was published to our main URL. The Cypress tests were not run in the Travis CI build because of the long process of going through every Cypress test.

We also used Lighthouse to verify that our application was complying with our selected threshold for measurable standards of PWA set by Google [25]. These thresholds were set to:

- Performance score: NaN - This score was not set as Lighthouse had a bug [24].
- Progressive Web App score: 90 - 100 in a ideal world but set at 90 so the test would still pass during development, should have been changed to 100 in the end.
- Accessibility score: 80 - We wanted to focus on accessibility but achieving a score of 100 could actually have decreased the quality of the application.
- Best Practices score: 80 - We wanted to follow best practices, but as the criteria are vague, it is difficult to achieve the maximum score of 100.
- SEO score: 1 - SEO was not a priority in this project.

To set a realistic threshold we looked at other well known PWAs.

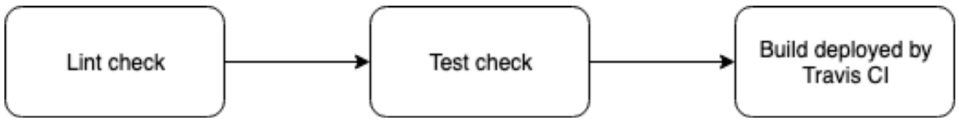


Figure 11: Pipeline

3.6.2 Unit testing

Jest and Enzyme was used for unit testing. When a team member worked on a new feature they had to write tests for the feature before a pull-request into develop was accepted. This extent of testing was in the beginning a challenge to follow when we only wanted to get a prototype going. There were several times we skipped the testing to get some code into the development and master branch. Later in the project we started to write test for features that were already merged. Later on test were a requirement to get new code merged into develop.

3.6.3 End-To-End testing and Integration testing

For End-To-End testing framework Cypress. We used Cypress to test that the frontend and backend worked together. Cypress allows you run test and display in the browser what every step of the test does. This made it easy to debug and see what was wrong. Cypress came with an existing API we could use to mimic navigation and text input in the project. This way we could see if we got the intended information we expected and wanted. When the input was submitted it would be sent to the database, processed and an answer would be sent back. Hence we could check if the feedback on the application was correct corresponding with the input. Only a few Cypress test was written, login, logout and when you login in for the first time you only get to option to send a DCA message 3.3.2 and not the rest of the messages. Future work is to implement tests for the whole process of going on a trip. End-To-End testing consists of multiple integration tests after each other. In this way we have done some integration testing.

3.6.4 User testing

Throughout the project we wanted to conduct user-testing on potential users. This proved to be more difficult than anticipated as we had our project while all the active fishermen we could get a hold of were in Lofoten or similar places further north in Norway due to the annually migration of Atlantic cod. In the beginning we created some sketches in Figma, appendix D, that we planned to use for user-testing using the Wizard of Oz method, but because fishermen were hard to get a hold of we could not to this without traveling far.

> *Wizard of Oz test is a user-based evaluation of unimplemented technology where, generally unknown to the user, a human or team is simulating some or all the responses of the system. [43]*

After conversations with Dualog about user-testing we agreed that they should try to get one of their test tablets and present our application to

some of the people available in Tromsø. Through a video conversation we could get an overview over the actions made. To make sure every user went through the same actions we created a set of tasks the user should try to perform. In the meantime we would take notes on what they were thinking, how they interacted with the application and where they met difficulties. This meant we could easily notice common difficulties. It was a great method to get feedback on what was working and what needed to be improved upon. Because of the difficulty of finding users to test on this was done fairly late in the project. We originally aimed to this parallel with the coding, but this was not possible.

To register an account for testing purposes our application has an open endpoint called register [29]. This allows anyone to sign up with email, password and a display name and thus be able to conduct tests. In the future, this will be handled by Dualog and they will simply forward the user details to the user.

3.6.5 Architecture validation results

We consider 25 of the 29 architectural significant requirements (ASR) listed in 3.2.3 as fulfilled. This indicates the architecture we have chosen solves the problem adequately. This is especially true when we take a look at the reasons why 4 of the ASRs failed.

- ASR 15 changed during the project, authentication from the customer will not be implemented by us.
- ASR 11 and 12 are about error feedback, and are not completed. We do have sensible error handling with messages. But there are not reports generated and sent to technicians or Dualog. So we do not consider these fulfilled.
- ASR 13 about AIS, it is a feature we have not prioritized and is therefore not completed.

The failed ASR are not failed because of the architecture, but because of changed requirements and unfinished features. The ASRs does not work against our architecture and could be implemented without problems in the architecture, given enough time.

3.7 Summary and future work

Throughout the project we had a lot of thoughts and ideas about what would be beneficial to add to the application. The following list shortly explain the main ideas that could improve the user experience and be beneficial as an incentive for fishermen to use the application.

- Making it possible for users to add all typical side catch with the click of a single list element.
- Push warnings notifying the user that he should send a DCA message, to further guide the user from making mistakes.
- A page to display different fishing statistics.
- More user testing as the tests we had were helpful, but more feedback is needed for further development.
- Cypress is integrated into the projects but more tests needs to be made.
- Automatically sending an empty DCA before midnight, as this is a requirement with overnight stays at sea.

Map
Through our meetings with Dualog it became very clear that implementing a map into the application should be of high priority seeing as it could potentially greatly enhance the user experience. The map would serve multiple purposes. Firstly, it would be used as an alternative method for the fishermen to input their positions, e.g. expected fishing spot. Secondly, it could be combined with a heat map layered on top to give a visual representation of the fishermen statistics. We believe this would be a valuable addition and it might give the fishermen an incentive to use the application.

3.7.1 Known bugs

During testing of the application we became aware of some bugs that were present in our application. As this was towards the end of the project fixing these were not a prioritization. For future work, fixing these would be benefitial.

- The user is able to land more fish than what currently is onboard the boat
- Our prefill() method is sometimes overwriting timestamps and locations that are manually filled in by the user
- The user can set the expected fishing start timestamp to be before the departure timestamp
- When the device changes orientation the application zooms in on some devices
- If the application refreshes while on the DCA part 2 form it loses input values from DCA part 1
- Editing message is not working as expected and it is not sent to Dualog on the correct format

4 Process

4.1 Project timeline

We decided to divide the project into 5 Sprints with a duration of 2 weeks each, where the first Sprint started the 28th of January. We also had a Sprint 0 where we organized the team and discussed some of our goals for the project. The workflow is shown in the figure below.

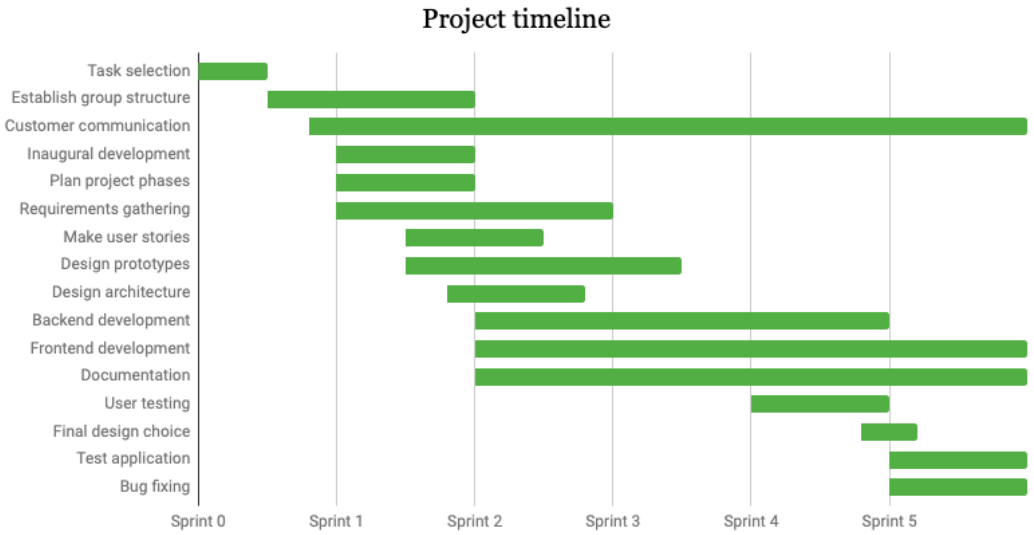


Figure 12: Gantt project timeline

Due to the short timeline of the project, a lot of work has been carried out in parallel. This enables rapid progress, given that there is a long term plan to avoid bottlenecks and inefficient work sequences. Some of the things we worked on concurrently were, code development, design, prototyping, architecture, prestudies, writing documentation and report.
In the first sprints we focused more on the prestudies and planning. Towards the end of the project the focus was more towards testing, validation, and writing documentation. The main focus during the entire project period has been writing code and developing the product.

The target was to finish the programming by the 3rd of April. The reason for such an early code freeze was that the majority of the team members

would participate in a school excursion that lasted about two and a half weeks starting the 4th of April. This posed a big challenge when it came to communication and work. A lot of websites were blocked in China, including our main communication channel Slack. As the schedule would be full of activities, there would be little to no time for further development during this period. With this in mind, we wanted to finish as much as possible of the project before this time.

4.2 Team organization

4.2.1 Group contract

One of the first things we did as a group was to make a team contract. The purpose of the team contract was to clarify expectations and to have a binding contract that every member of the team had to follow. If one or more members of the team would start to lose perspective or focus, the group contract could help guide the teamwork and project back on track.

The group contract contains:

- Our ambitions for the project
- Attendance and meetings
- Expectations to the individual team members
- Disagreements and deviations
- Communication
- Signatures

4.2.2 Time management

In our group contract we specified that it was expected that every team member worked approximately 20 hours every week. We chose hours worked as a fair comparison to measure the team members in this project. One of the main reason for this is that it was easy to keep track of and it was easy to plan with.

There were also non-practical reasons for choosing effort comparison over a performance or result based comparison. It compensates for the difference in knowledge and programming skill. This promoted teamwork as it is in every team members best interest to help others be productive. Contradictory

to a performance based measurement where one's best interest is in spending the least amount of time completing the task given. In such an environment there might not be such a big interest in helping others in the team that needs it.

4.2.3 Roles and competencies

In a team of 7, making decisions can easily result in longer discussions and problems with agreeing. To shorten down decision time the team agreed on creating areas of responsibilities and assign these to members of the team.

Decisions that would affect several parts of the project or were considered important would be run through everyone at meetings. When we made these decisions in plenary everyone would have a say in what should be done. This way every member would be able to participate and get ownership of the project no matter what area they were responsible for.
If we were not able to find common ground on a decision that would affect the application the group would consult with the customers, preferably in one of the meetings. This way we were sure the customer had the final word and the application would be more catered towards their wishes.

When there were smaller, less significant decisions to be made, the person responsible for each area was free to make a decision on behalf of the team. A clear trade off with using this approach was that the speed at which the project was developed would be traded for democracy and thoroughness.

Each person would be responsible and accountable for the planning, execution and documentation of the delegated area. This did not mean that this person had to do all the work single-handed, but that they had the ability to delegate work and were expected to have a general overview. This made the work more structured as we knew who to ask within each area.
Each role is described in detail in Appendix A.

Areas of responsibilities:

Leader: Tord Standnes

Minutes: Ebba Fingarsen

Frontend lead: Balázs Orbán

Scrum: Ebba Fingarsen

UX lead: Morten Falstad

Backend lead: Håvard Bergheim Olsen

CI/CD lead: Balázs Orbán

Customer relationship manager: Morten Falstad

Test lead: Petter Grø Rein

Architecture lead: Tord Standnes

Security lead: Tore Stensaker Tefre

4.2.4 Customer relationship management

The relationship with the customers was good throughout the entire project. After contact was established with e-mail, all communication happened through Slack where the customers were delegated a shared channel. This made it easy to keep in contact, and share documents and other relevant information. One of our customers were based in Tromsø which made meetings a bit more difficult. To carry out these meetings the best way possible was with video conferences. This way we could have a more genuine conversation despite the distance. This worked well aside from some occasional technical difficulties.

4.2.5 Relation with the supervisor

To keep in contact with the supervisor we made a separate Slack channel. Here we could ask questions that were easily answered and express any uncertainty we had connected to the report. If we had more extensive questions or concerns we would express these in any of the set meetings. To keep the supervisor updated on the status of the project we delivered status reports. These explained what user stories had been done, what was planned forward and if we had encountered any problems.

4.3 The UX process

Dualog expressed that user experience should be of utmost importance. To make this happen we analyzed the problem, made prototypes and conducted user tests to improve upon the overall user experience.

4.3.1 Analyzing the problem

The first step in creating a good user experience is understanding the problem and the user. To accomplish this we gathered as much information as possible from Dualog.

The first meetings with Dualog were mostly about understanding the legal aspect, logical details of the task and which expectations the customer had towards us as a team, and the end product. Dualog gave us some goals regarding the user experience, but withheld their previous applications and products, as they did not want us to get influenced by the previous designs. They wanted us to rethink how the whole reporting process could be realized. Based on the information provided by the customer and our own studies, we started developing.

One of the techniques we used to extract more information was the business context mentioned in chapter 3.1.2. This gave us insight about the task, which context it was created for, and the intention for the product after this project.

4.3.2 Prototyping

During brainstorming, each member was asked to come up with some ideas on how they envisioned the application. When we went through these in plenary we noticed there were both positive and negative aspects to every idea. From the ideas we developed two designs to show the customer.

When developing the sketches we tried to follow a continuous double diamond (Figure 13) strategy. Here we brainstormed and created different designs, then reviewed them as a group, before we decide to proceed to develop the design further using the agreed upon components or elements.

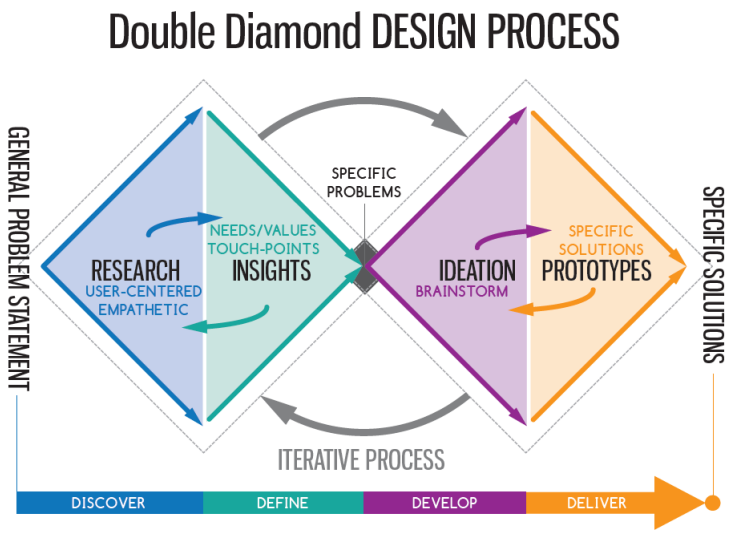


Figure 13: Double Diamond [9]

The Double Diamond model helped us have an agile approach to the design, and made it easier for us to detect errors or solve problems. After having multiple iterations internally, we presented it for representatives from Bekk and Dualog and received comments and ideas from them on how to further develop it. After presenting, they said they wanted a combination of the two solutions, and the group started working on implementing the design ideas in a functional product.

By working on the sketches in correlation with development, we could save time as the customer already had given an input on ideas we had in the sketches, so that we knew how the customer wanted it.

4.4 User testing

4.4.1 Figma user test

To ensure the team and the end users understood each other, we wanted to conduct user testing of the sketches and the final product. In order to find relevant test applicants we reached out to the local fishing association [16], and asked if they were interested in helping us. This proved difficult because of the fishing seasons all fishermen that seemed relevant were in Lofoten, so we only had test applicants from the customer as a viable option.

As mentioned in 4.3.2 we presented the sketches to these test applicants by sending them the sketches a few days before a meeting. This way they had some time to gather their thoughts before we went through our ideas and thoughts on the meeting. The feedback from this meeting helped us understand the essence of limiting the number of clicks a user has to make. In order for this to happen, we proposed that D.1 should be combined with D.2.

During these meetings translations and understandable wording was also a issue. There are a lot of fishing terminology in the application, that none of the team members were familiar with before this project. We ended up having multiple conversation with our customer throughout the project about wording. Some examples of the translations and terminology can be found in appendix E. This is also something we were aware of during user testing and changed the wording where it was commented on. Though we have implemented a lot of fishing terminology, we recommend our customer to further elaborate the terminology in the application, as this is important for the fishermen.

4.4.2 Phone interviews

We conducted multiple phone interviews with fishermen from different parts of Norway in order to get input on our current ideas surrounding the application and what features they desired.The results of these can be found in Appendix G. After multiple interviews it became clear what the fishermen feared the most, e.g. being overrun by rules and having to use an excess amount of time doing paperwork, while still doing the same amount of work at sea. Multiple fishermen mentioned that they wanted a category when reporting fish called bycatch. Essentially it means all fish that is not part of a quota, should fall under one category in the catch report.

The phone interviews provided us with multiple ideas and things to be aware of when developing the UX, with the biggest feature being the preset page. We also got an understanding of what terminology the fishermen were familiar with, and decided we should use abbreviations they knew from before. This meant that the different message types ended up being called the same in the application as in the regulations; DEP for departure messages, DCA for the catch messages, and POR for the port arrival messages.

User test
We held one user test on the product at the end of Sprint 4, for an employee of Dualog that now also works as a fisherman. It was originally planned to

hold a Wizard of Oz test, providing the user with tasks without assisting more than necessary. This proved to be problematic as the product was in too early development phase, so the test did not provide us with as good of a result as we would have hoped. We did however get input on several things such as how the use of language and icons could be improved.

4.5 Architecture

4.5.1 Plan

In the start of the project the architecture lead wrote this plan for the completion of the architecture.

Sprint 1

- Make an execution plan for architecture
- Present the architecture plan for the team
- Decompose the problem into understandable material
- Design the architecture

Sprint 2

- Present and validate the architecture to the customer

Delivery phase

- Finalize the architecture views
- Finalize the documentation for the implemented architecture
- Finalize Documentation

4.5.2 Process

We encountered multiple problems during the design of the architecture, but all of these were resolved. The original plan was to use a method called attribute driven design (ADD)[1]. Where we find out what qualities we want in our software fist, create requirements for the architecture and then design it.

We followed through with the decompose phase with business context and the creation of Architectural Significant Requirements, but dropped ADD in

the end. This was due to the architecture lead was sick for 2 weeks. We solved this by the team looking at the requirements and finding a suitable architecture, that meet all the requirements.

The validation of the architecture was postponed to Sprint 3 due to illness. During validation with the customer we discovered some misunderstanding about the backend. The customer wanted little focus and work on the backend, while the team wanted a layer between the client and the customer endpoint. After some back and fourth we found common ground and a common understanding of the customers needs.

4.6 Security and risk analysis

The application was designed with security in mind. This included establishing a security report based on the Risk Management Framework, as used by the US Government[31]. Furthermore, we tried to be critical when introducing new features, and sought to identify and address possible security issues early, so that we could take measures to mitigate them.

4.6.1 Security report

In the security report we identified and addressed security issues that may be present during the use of our application, limited to the PWA. The servers that receives our data is not within the scope of this report, as well as security issues on a low level such as Device OS and browser.

The security report features a test plan, and it also includes a risk analysis, based on the Risk Severity Matrix, where all relevant risks were graded based on likelihood and consequence. The risks were also analyzed in order to find countermeasures to mitigate them. Misuse cases were also used to show how the application may fail. This way we could reduce the threat of application security failure.

To ease the identification of the potential business risks for the application and our customer, we utilized the overview of business assets and business goals established in the business context part of this report. These are referred to in the report as BG1 for “Business goal 1” and BA1 for “Business asset 1”, and can be found in section 3.1.2.

4.6.2 Understanding of risks

In order to best identify our risks, we needed to first establish an understanding of the term risk. Throughout this report risk has been defined as the compound of the two concepts probability and consequence. This implies that the probability of an event is the chance that an event will occur. The consequence is a measure of how severe the outcome of an event will be and how much cost would be incurred. This means that risk would translate into a measure of the potential cost and is found by combining the chance of an event occurring and the costs if it does occur.

The Risk Severity Matrix below shows how a combination of likelihood and consequences represents a risk for the business, both regarding technical risks and business risks, and has together with the Risk Severity Matrix (Figure 8) been developed by the US Department of Commerce [31].

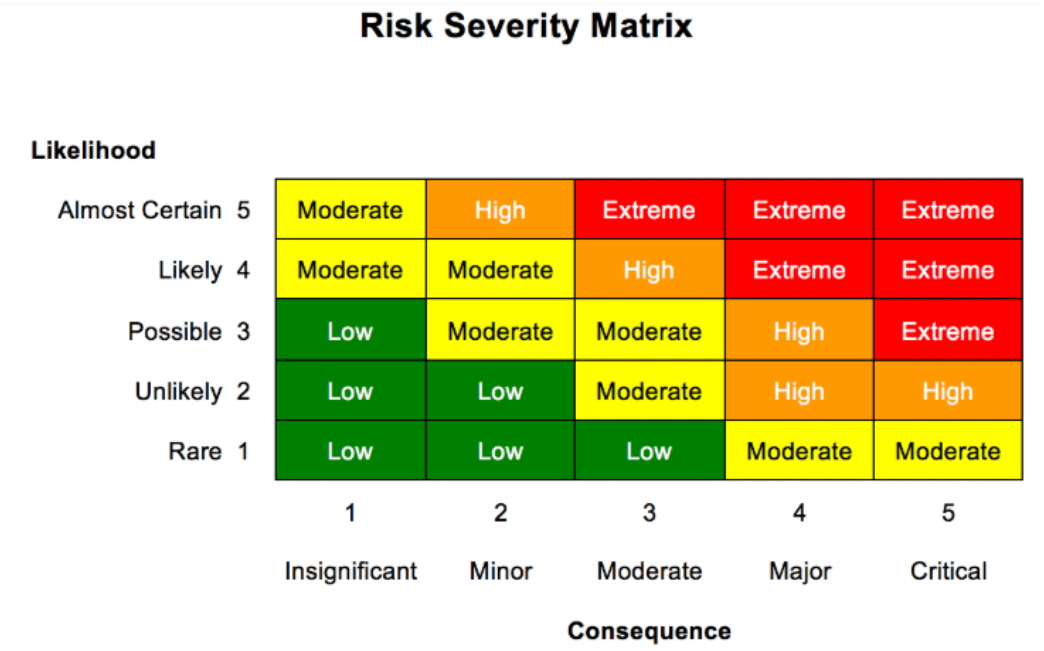


Figure 14: Risk Severity Matrix

4.6.3 Project risks

Our project risks shows the general high-level administration risks that the project faces.

In the following table, L denotes Likelihood and C denotes Consequences, scaled as in the Risk Severity Matrix above(Figure 8). The maximum risk product is 25, and the table shows what strategy we use to deal with the risk in question.

ID	Title	Description	L	C	Risk strategy	Risk product
1	Team member leaving	Team member leaves the project. May cause loss of project knowledge.	2 - 40%	4	Mitigate	8
2	Customer unsatisfied	Customer becomes unsatisfied with progress or direction of project	2 - 40%	3	Mitigate	6
3	Customer pulls out	Customer leaves project	1 - 20%	4	Accept	4
4	Project not finished	Project is not finished as agreed	3 - 60%	5	Mitigate	15

4.6.4 Project risk mitigation

The project risk mitigation table shows how the team aims to reduce the consequences of the events mentioned in the previous section.

Project risk ID	Counter measure	Description
1	Share knowledge	- Use GitHub for issue description and code distribution - Make sure the workload is distributed - Pair programming and cooperation
2	Communication with customer	- Avoid misunderstandings by having good communications with customer - Discuss important issues with customer - Receive feedback from customer
3	Trust customer	Customer would have lost reputation if leaving a bachelor project
4	Agile project methods	- Start with Minimal Viable Product(MVP), and extend project based on importance and available time - Keep customer informed about progress and what to expect

4.6.5 Business risks

The business risk table is an attempt to map how events and errors related to our product, may influence the business of our customer. The business risks are connected to the business goals listed in section 3.1.2.

ID	Description	Likeli-hood	Cons	Risk	Connected business goal
BR1	Application is unavail-iable	4	2	Moderate	BG6
BR2	Application is difficult to use	2	3	Moderate	BG8
BR3	Application weakens reputation of Dualog	2	3	Moderate	BG6

BR4	Another company makes a better application	3	3	Moderate	BG5
BR5	Application data is compromised	2	3	Moderate	BG12
BR6	Dualog can not trust user authentication	3	4	High	BG12
BR7	The system seems unsafe or unsecure	3	1	Low	BG7
BR8	Loss of data stored on clients	5	2	High	BG6
BR9	Laws are changed and reporting is unnecessary	4	4	Extreme	BG10
BR10	The application reports illegal or wrong fishing	2	4	High	BG13
BR11	Target group does not use our application	1	5	Moderate	BG14
BR12	Application difficult to maintain	1	3	Low	BG9
BR13	Application is not availiable for international markets	1	2	Low	BG15

4.6.6 Technical risks

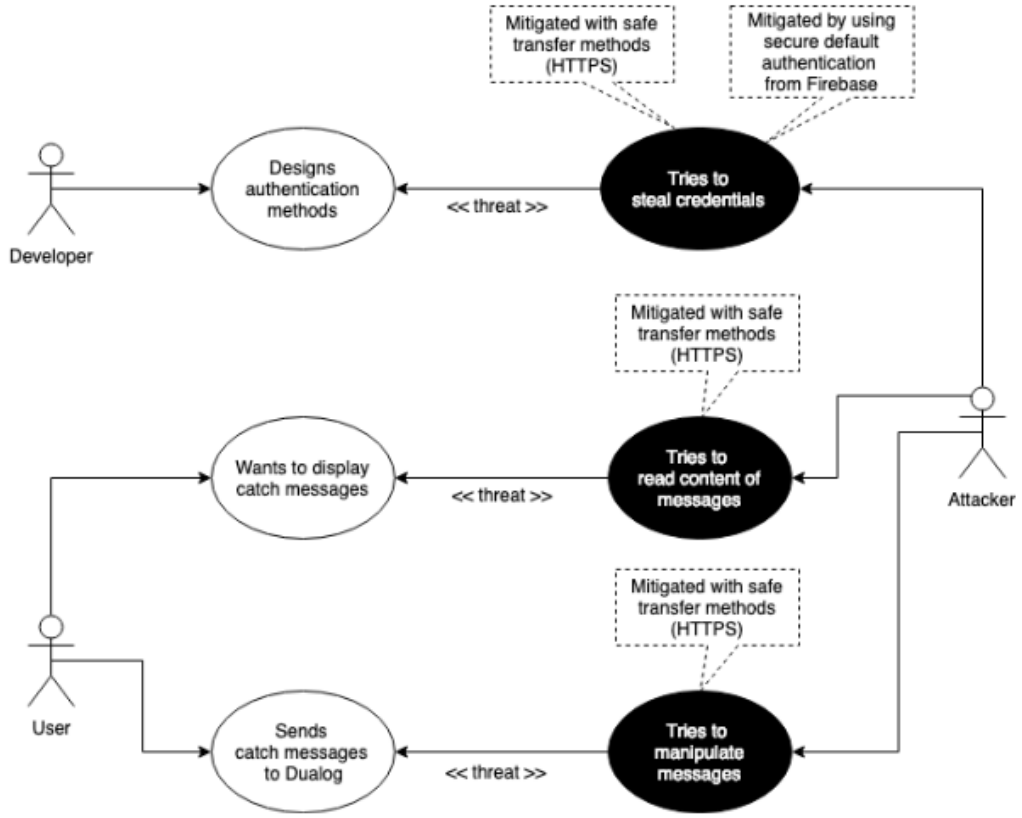
The technical risk table attempts to map how technical errors may introduce the business risks of the table in the previous section.

ID	Description	Likelihood	Cons	Risk	Security measurement	Connected Business Risk
TR1.1	Server malfunction	2	3	Low	Ensure server is safely hosted (Google Firebase), and make application work offline.	BR1

TR1.2	Server data is deleted	2	3	Moderate	Backup and database access control	BR1
TR1.3	Cached data on device is lost	3	2	Moderate	Upload things to server as fast as possible	BR5
TR5.1	Security keys are compromised	2	3	Moderate	Store and transfer keys in safe ways	BR5
TR5.2	Broken admin authentication	2	4	High	Restrict admin functionality, and make sure those functions are secure	BR5
TR5.3	Unencrypted communication with servers	1	4	Moderate	Enforce HTTPS, as is standard with Google Firebase.	BR5
TR5.4	Data is compromised	2	1	Low	Data can already be constructed through publicly avalilable data such as AIS	BR5
TR6.1	Broken user authentication	2	2	Low	Limit number of login attempts and put users into quarantine	BR6
TR7.1	Users think our system is unsafe	1	1	Low	Make sure users are informed about events and that the application is safe	BR7
TR9.1	Fishermen are fined because of us	2	4	High	Application must ensure fishermen work legal	BR9

4.6.7 Misuse cases

Our misuse cases demonstrates how an attacker may try to find vulnerabilities in our application, and the steps we have taken to mitigate these possible threats.



4.6.8 Security test plan

The test plan is based on the business risks and technical risks. The plan shows some of the ways our product may be tested to ensure safe operation, and is connected to the technical risks of the table in the previous section.

Test ID	Related tech risk	Test detail	Expected outcome	Result and explanation
T1	TR1.1	DDoS attack bringing down server	System should be unaffected	As the application is cachedriven, and Firebase as a cloud provider has nearly unlimited capacity, a DDoS attack did not have any effect on our product.
T2	TR6.1	Try to log in with credentials "admin" and "pass"	Should be denied login	The user was denied login as Firebase is unable to authenticate the credentials.
T3	TR6.1	SQL Injection during authentication	Should be denied login	The user was denied login as Firebase authentication system prevents SQL injection.
T4	TR6.1	Brute force authentication	Unable to access	The admin interface of the application is located at Google servers, separate from the application, and protected on IP-basis by Google. Thus, simple brute force attacks will not be successful.

4.6.9 List of vulnerabilities in the final product

During the process, the team has tried to log and identify all possible security and privacy issues. The issues which has not been mitigated are listed in the following section.

V1: Manual creation of user credentials
As our customer required, user accounts are currently created manually through human interaction with the database. This means that user credentials, including passwords, are transferred from our customers call centre to the customer. In turn, this means that credentials may be known by someone other than the user, and that credentials may be transferred in unsafe

ways, vulnerable to data sniffing etc.

Safe functionality where the user creates their own credentials can be enabled with only small changes in the source code, but currently, this may be a vulnerability.

V2: Cache is not deleted on log out
During tests prior to the final delivery of the product, we discovered that data saved to cache is not deleted on log out. This means that users may be able to display the catch data of other users in the very rare occasions where multiple fishermen use the same device.

However, we do not think that this represents a major vulnerability as the data is inaccessible without an user account created by Dualog, due to the enforcement of Firebase security rules.

4.7 Development methods

4.7.1 Process model

During this project we decided to work with Scrum. The main reason for this was that the entire team was familiar with this process model before the project started and it would help us finish out project on time. Scrum is highly agile and would allow us to easily adapt to feedback from both the customer and users. This way we would always have a minimum viable product that could be tested and shared with the customer and users.

At the start of every Sprint we would have a meeting with Sprint planning. Here we would choose the a new Sprint goal, choose user stories and make issues. The Sprint goal would be a general goal we would work to reach during the 2 weeks. This would have a connection to the chosen user stories that defined what we had deemed as important for the Sprint. We chose to make issues as a part of the Sprint planning to make sure everyone were aware of what needed to be done, and to define the issues in such a way that every issue was understood.

At the end of each Sprint we would have a Sprint review followed by a Sprint retrospective. The main purpose of the Sprint review was to get an overview over how the project was going according to plan. This entailed figuring out what was finished, what was not finished, if we faced any problems and how we solved these.

During the Sprint retrospective we would focus less on what was done. Here we would instead focus on how it was done and how it worked out. We would mainly answer 3 questions.

- What went well?
- What could have been better?
- What can we do to improve upon these challenges during the nest Sprint?

Here every member would be able to explain their challenges and we could work to solve these as a group and improve upon our work-dynamic.

To be able to define what was finished or not during the Sprint review and during the project we decided to use a definition of done. This was a list that had to be fulfilled for a feature or issue to be considered finished. For a pull request to be accepted, and code merged into develop or master, one (develop) or two (master) other team members had to accept the request. This involved looking for typos, logical mistakes, linting, going into the branch and checking everything compiled as it should, testing the sought features, checking if the code was documented, the necessary code was tested and the code followed the "best practice".

4.7.2 Communication tools

Our primary communication tool was Slack and stakeholders such as the customers were represented in their respective channels. Our supervisor was also present in a separate channel where we would keep contact regarding upcoming meetings and ask shorter questions. We also had a channel in Slack where Travis CI informed the team if there were something unexpected happening with tests and pipelines. GitHub notified about issue and pull request updates in the same channel.

4.7.3 Organizing tools

We organized team meetings through a joint Google Calendar, which had a bot in Slack informing the team about upcoming events and deadlines. Meetings with stakeholders such as customers were either organized through Slack, or through email should the stakeholder not be available in Slack.

4.7.4 Documentation

Minutes and relevant documents were archived in a Google Drive folder that was shared with all members of the development team. In instances where documents and minutes needed to be accessed by people not part of our group are granted access through shareable links.

Google Drive made all documents easily accessible to every member and made it possible for everyone to edit documents and leave comments for others to see if needed.

4.7.5 Distribution of time

Throughout the project we used the software Toggl[37] to track our weekly work hours. Toggl is a cross platform time tracking application, that allows us to specify what we were working with. The categories we used were:

- Meetings, lectures and communication
- Product
- Report and process

The use of Toggl allowed us to see how much time we spent on each category. On top of this we would have weekly checkups within the team to see if everyone worked the required hours. One team member chose to use a tool called Wakatime [42] that only tracks hours spent in the code editor. This is fully automatic and creates no overhead. All team members were required to track hours as per the team contract, and both Toggl and Wakatime were defined as acceptable tools to use.

The work distribution as of the delivery date the 10th of May between of the different project parts shows, combined with figure 18, that we have spent 733 hours in total on the product, approximately 350 hours on the report, and approximately 300 hours on different kinds of meetings and lectures.

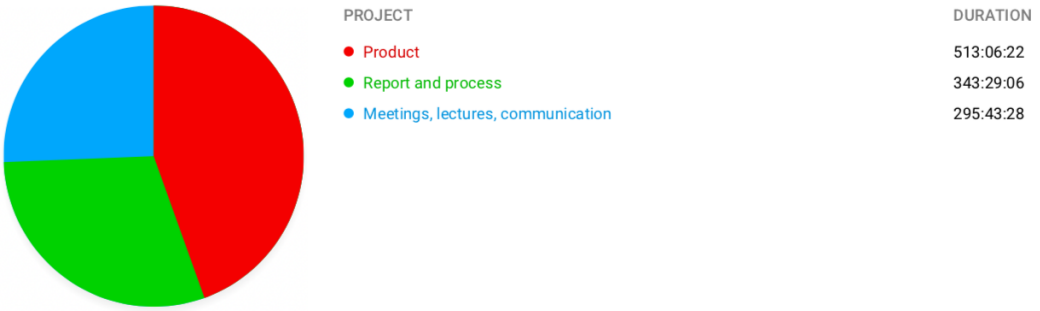


Figure 15: Task distributed work hours - Toggl

According to what ended up being the grading criteria, we should have spent more time writing the report, however, it seems that we misunderstood how the project would be graded from the start.

Toggl also made it possible to see when we worked the most, which worked out to be during the months of February and March. This was natural as we were not assigned a task until the 20th of January, and that the majority of the group left the country for large parts of April. The finishing touches on the report during the last week generated a lot of work hours, which was natural due to preparations for the demonstration the week before.

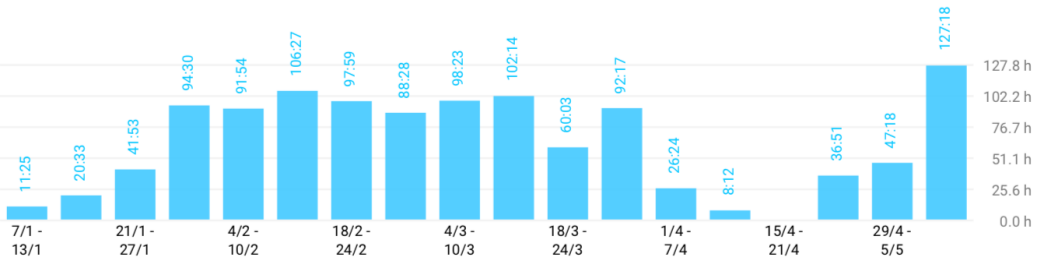


Figure 16: Weekly work hours - Toggl

We can also see that the total hours spent by the group members averages to about 200 hours each. This is a bit below our target specified in section 4.2.2 where the group members agreed to spend 20 hours a week on the project, as 200 hours divided by 15 weeks equals about 14 hours. If we remove the excursion weeks, we get 200 hours divided by 12 weeks which equals to about 17 hours, which is also below our aforementioned target of 20 hours.

When looking at the figure below, it is worth mentioning that team member Petter was ill for some weeks during the project.

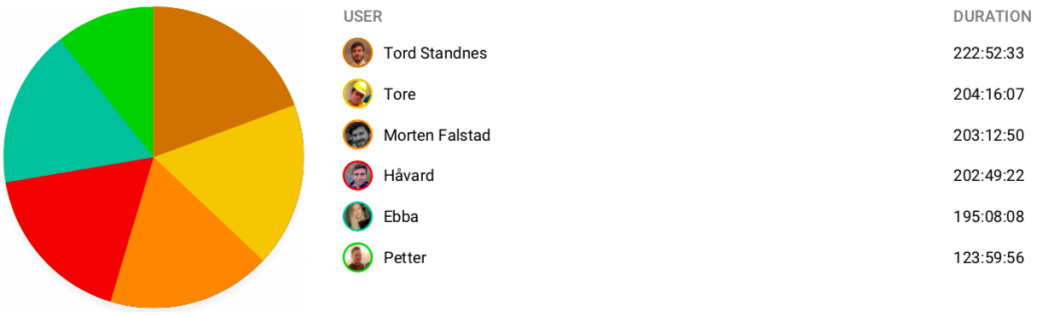


Figure 17: Personal work hours - Toggl

The following figure shows the hours Balázs spent on coding. As a result of him using WakaTime to track his hours they are not reflecting the time he spent researching or being in meetings. The time spent researching is sadly untracked and therefore unknown, but it is safe to say it was a substantial amount. Adding his coding hours with the group's average time spent in meetings he spent a total of 265 hours.

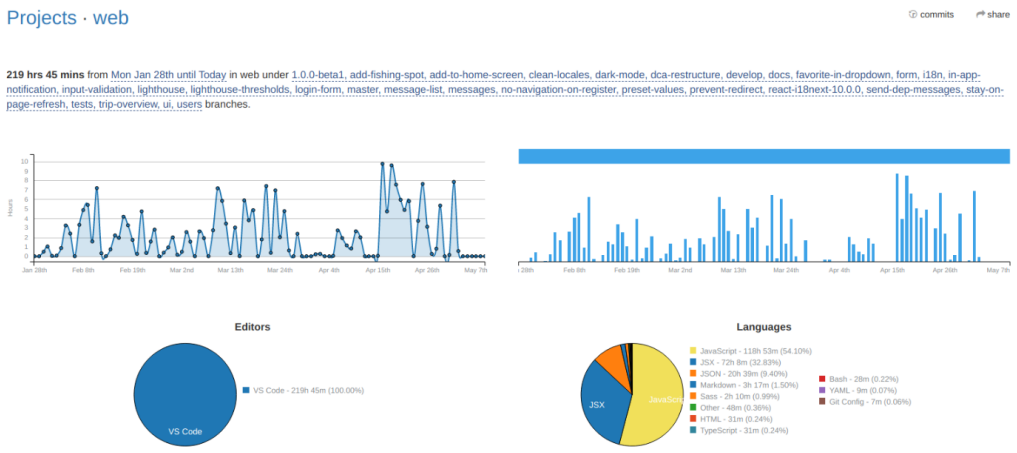


Figure 18: Personal work hours Balázs - WakaTime

4.7.6 Meetings

In addition to two set team meetings every week, Monday and Thursday, the team communicated mostly over Slack. Here we had different channels for each topic like frontend, backend, testing and similar. This was to have more organized conversations and the information decided upon waS easier to keep track of. The team also had some joint workshops to work closer as a group. Here we could work in a less formal setting while still being productive.

In the beginning of the project we had weekly meetings with the customers outside of our communication on Slack. Towards the end there was no need for weekly meetings and the frequency was set to once every two weeks. The meetings took place towards the end of the Sprint to show our process and get feedback for the next Sprint.

4.8 Changes during the project

As the project went through a lot of changes that greatly impacted the process, we decided to list some of the changes in this section.

4.8.1 Sprint 0

During this Sprint the team focused on defining the scope of the project, the wanted structure, making templates for minutes, retrospectives and reviews and getting all technologies in place for the initial software development start. In this Sprint the prestudies mentioned in section 3.1 were conducted to make sure decisions affecting later development and choices were educated and thought through.

We also gathered information on the level of competencies each group member had, and what expectations they had for the project, in terms of learning and grade. This was to make sure everyone was on the same page and there would be less misunderstandings.

The biggest challenge during this Sprint was to find common hours for meetings. As every member of the team was a student and several members had part-time jobs at different times of the week, it was difficult to find common ground.

4.8.2 Sprint 1

The Sprint goal for Sprint 1 was defined as "Have an MVP product. By this we mean that the application should send a test message and the application should be able to format and send the 3 messages required.".

The table below contains the chosen user stories for Sprint 1 and their status. The status indicates if the user story was finished, started or not yet started by the end of the Sprint.

ID	Description	Status
22	The fisherman can send his messages without thinking about authentication of the message	Finished
26	As a customer i would like the application to reflect the company design colors	Finished
20	The fisherman should be able to choose between light and dark mode	Finished
12	The fisherman should be able to choose between Norwegian and English language, and have support to later add other languages	Finished
31	The fisherman should be able to use the application offline	Started
13	The fisherman needs to know if the request has been delivered and is ok	Started
15	The fisherman should be able to change or cancel their fishing estimation report	Not started
25	Dualog needs to get sensible information from error messages	Not started
8	The Directorate should know the messages have integrity	Not started
23	The fisherman can rely that the message gets sent	Not started
14	The fisherman should be able to report leaving the dock/shore	Not started

Some of the finished user stories had to be continued to be implemented as new features were added, like more translations (20) for new pages and design for new elements (26 and 12).
We realised that our Sprint goal was very optimistic and that the startup phase of the development would take some time before it got more effective. This meant that there were user stories that did not get finished on time. In future Sprints we aimed to have less user stories and work towards finishing

a higher percentage of these.

We encountered some challenges during this Sprint. The division of labor was not as even as we wanted it to be, members needed to take more initiative when it came to working with the code, and communication and focus during meetings could be better.

To solve these problems we agreed on some countermeasures:

- Everyone should look through the meeting agenda before a meeting to be alert as to what will be discussed.
- Everyone should work towards meeting the 20 hours every week.
- Keep the GitHub board and issues better updated to make sure everyone was up to date.

4.8.3 Sprint 2

The goal for Sprint 2 was formulated as "frontend should be able to take input from users through forms, and send complete messages." and the user stories for this Sprint were the following:

ID	Description	Status
31	The fisherman should be able to use the application offline	Finished
23	The fisherman can rely that the message gets sent	Finished
14	The fisherman should be able to report leaving the dock/shore	Finished
13	The fisherman needs to know if the request has been delivered and is ok	Finished
8	The Directorate should know the messages have integrity	Started
15	The fisherman should be able to change or cancel their fishing estimation report	Started
25	Dualog needs to get sensible information from error messages	Started
21	The fisherman should be able to get an overview over the fish currently on his boat	Started

During this Sprint we got some logic for the messages in place and finished the departure (DEP) message. The UI for report of the catch (DCA) and port arrival (POR) messages were begun but the logic was not yet finished. The display of messages and their status were in place. We were yet to finish the logic behind editing a message and displaying the fish currently on board.

Some technical challenges we met during this Sprint were that Service Workers were not as easy to customize as we initially thought. This meant we spent more time than expected to solve this problem.
Our decision to make the frontend as simple as possible meant there would have to be done more work on the backend in regards to authentication of geographic positions, timestamps and messages.

The team were more pleased with this Sprint overall. We finished more of the user stories, the meetings were more effective, members took more initiative, the communication was better and there was written more code. However there were still some points to improve upon, in particular regarding testing. The team agreed to be stricter and not approve Pull Requests that did not have the testing required for it to be considered finished.

4.8.4 Sprint 3

The goal for this Sprint was that the application should follow the Directorates' rules for how messages should be sent, and should be able to send one DEP message at the time, multiple DCA messages and one POR message per trip.

ID	Description	Status
8	The Directorate should know the messages have integrity	Finished
17	The fisherman should be able to get an easy overview over previous messages sent	Finished
10	The fisherman should be able to read realtime GPS location in the application	Finished
5	The user should be able to fill out the form offline (one time)	Finished
9	The fisherman should be able to set GPS location from the application	Finished
15	The fisherman should be able to change or cancel their fishing estimation report	Started
21	The fisherman should be able to get an overview over the fish currently on his boat	Started
4	The fisherman should be able to preset settings to their specific situation	Started
16	If the fisherman stays out overnight, a message should be sent before midnight	Not started

During this Sprint the Sprint goal was almost met. This was something we were pleased with because it entailed that the base functionality was nearly in place. DCA and POR messages were fully done, but DEP messages had a to get a little more logic in place before it could be considered done.

This Sprint we also decided it would be beneficial to change the date of the code freeze. It was changed from 3rd of April to the 30th of April. Most of the application would still be done before the original date, but this way we had some more time to make minor changes to the application even after the excursion

We noticed that the communication had gotten a bit worse compared to earlier Sprints. Members were not fully aware what everyone was working on and smaller misunderstandings and discussions appeared within the group. On top of this we wanted to make sure more code got pushed to Master. This

meant the group had to make sure all Pull Requests met the set requirements in the 'Definition of done'.

These were the measures set to improve on the challenges:

- Petter, as test responsible, should remind people to make tests and verify the test coverage.
- Everyone should read up on testing and take more initiative to help each cother out.
- Everyone should be stricter when approving Pull Requests.
- Make sure everyone is present when the Issues for GitHub.
- Disclose any problems we face early and discuss it early on. If they can not be solved via Slack, take it in person on one of the set meetings.

4.8.5 Sprint 4

The goal for this Sprint was to have working functionality for presets in the application to reduce clicks and ease use.

ID	Description	Status
7	The fisherman should only have to log in one time	Finished
42	As a fisherman i want the application to suggest standard settings that are feasible	Finished
15	The fisherman should be able to change or cancel their fishing estimation report	Started
36	As a fisherman, i would want the application to know the law (updated quotas), so i do not need to think about it	Started
4	The fisherman should be able to preset settings to their specific situation	Started
35	As a fisherman, i would want push-warnings when i need to do something	Started
21	The fisherman should be able to get an overview over the fish currently on his boat	Started
3	The fisherman should be able to quickly and easily report fishing	Not started
41	As a fisherman i want the application to help me send messages on time	Not started

25	Dualog needs to get sensible information from error messages	Not started
16	If the fisherman stays out overnight, a message should be sent before midnight	Not started

There were discussions regarding validation of messages on the backend, how much time this had taken so far and how much should be prioritized going forward. The impression we got from the customer was that they wanted and expected validation on the server as well as on frontend. The misunderstanding that arose was that the customer only wanted an acknowledgement from the server and that every message should be accepted regardless of validation. Had this been clear earlier would the time have been spent elsewhere.

We were pleased that we were able to conduct a user test this Sprint. This was something we had aimed to do during development and even though conducting it in Sprint 4 was a little later than expected we were pleased. After the backend discussion we used some time to clean up loose ends and considered the backend to be done. This was beneficial as we had one less thing to focus on going forward. We also got good feedback on our representation of trips in our application. This followed Don Normans principle about mapping and made it easier for the fishermen to connect a trip in the application to a trip in real life.

Some challenges we faced this Sprint was that not all members of the team understood all parts of the code. As not everyone was on the same level, but everyone wanted to do their best, parts of the code got developed at different speeds. We noticed that the difference in knowledge was a bigger challenge than we first expected.

To improve for the next Sprint we decided to focus on making sure to delegate the "easier" issues to the less experienced team members so everyone would have something to work with. This would improve morale in the group and make sure more of the issues were finished.

4.8.6 Sprint 5

The goal for this sprint was to improve the existing UX and make it harder to make mistakes, using more validation and guiding.

ID	Description	Status
28	The fisherman should feel safe using the application	Finished
21	The fisherman should be able to get an overview over the fish currently on his boat	Finished
11	The fisherman should be able to store information locally on the device	Finished
36	As a fisherman, i would want the application to know the law (updated quotas), so i do not need to think about it.	Finished
4	The fisherman should be able to preset settings to their specific situation	Finished
19	The fisherman should be able to select multiple fish types when reporting.	Finished
3	The fisherman should be able to quickly and easily report fishing	Finished
39	As a fisherman, i want the application to guide me as much as possible, as i am not good with data technology	Finished
6	The fisherman should be able to use the application on all his devices	Finished
2	The fisherman must be able to use the application with little to no technological background	Started
1	The fisherman must be able to use the application with (big gloves and) big fingers	Started
33	The fisherman should be able to understand, and use the application without complications	Started
15	The fisherman should be able to change or cancel their fishing estimation report	Started
35	As a fisherman, i would want push-warnings when i need to do something	Not started
16	If the fisherman stays out overnight, a message should be sent before midnight	Not started

After this Sprint we were pleased that a majority of the user stories were finished. The code was more structured, several user stories started in other Sprints were finished and we had a working product with the basic functionality required. A lot of small changes had been made in regards to the feedback from the user testing. This helped the application feel more fin-

ished and tweaked things that were easy to fix but had an impact on the user experience.

The focus during this Sprint was not as good as earlier. Members of the team started to become preoccupied with other subjects and were not as motivated to prioritize the project. Several members also spent some time getting ready for the excursion which would take place at the end of this Sprint.

If we were to have one more Sprint after this we would have focused on the following:

- Have more rewards for doing well. Instead of just having punishments for not working enough we could have focused on having a "team member of the Sprint" or something similar.
- We could have had more workshops. These were both social and productive as we were working together as a group, but in a less serious setting with food and room for more talking.

4.9 Evaluation and reflection

The key when learning from group work are evaluation and reflection. Our reflection notes is therefore included in the following sections.

4.9.1 What went well

During the development of our product, the concept of Scrum Sprints, have been helpful for the team to manage our tasks, missions, and goals. We also think that the Sprint planning we did as part of working with the Scrum methodology, helped us better understand at what pace we needed to work, and estimate how much of the project we could successfully deliver to the customer.

Working with two different customers have also worked out well. The way we have done this, has been that Dualog have acted as our customer, while Dag Frode at Bekk have approved the work we have done, and helped us with technology choices when we have needed some advice. This has been helpful, as Dag Frode has been acting as our technical supervisor, and helped us with some of our technical issues.

4.9.2 Challenges we faced

In a customer oriented project, there will always be challenges related to the customer. These challenges can be everything from communication to goals, and we have experienced some of them. Our primary challenge while working with a customer, was the fact that our end customer was located far away from the development team. This led to decisions being difficult to take, and some of our communication suffered from only having video based meetings. This was both due to the nature of video based meetings, but also because we were experiencing various technical issues.

We have also experienced how it is to work with people that have tight schedules, and there has been some friction related to what we could expect from the customer. This was mainly related to the backend of our product, where there was some misunderstandings between us and our customer. This resulted in our customer giving us access to a backend system which was far from what we expected, and thus that we had to rely on a independently developed backend system made by one of the group members.

As in many other IT development projects, there has been issues with one of the finest arts in IT, development speed estimation. This has led to some delays in development, as shown in Appendix C: Sprint plan, where yellow tasks have been postponed at least one Sprint, and red tasks were postponed beyond our project deadline and therefore not finished.

The above-mentioned challenge is also part of the next one - the excursion. During the month of April, 5 out of 7 team members went to China to learn more about Chinese culture and technology. The way we solved this, was to agree on an internal deadline date which was the departure date of the excursion group. The team thinks this decision was a good one, but because of our lacking estimation skills, a small amount of work had to be carried out after the deadline.

4.9.3 Group dynamics

All teams experience issues, as did we. All the way from the start of the project, conflicts of interest have been present in the group. This ranges from the selection of technology stack, all the way to the structure of this report.

Throughout the project, there has been discussion in our group on how we should use the available time. Some of the team members have recommended

spending more time on programming, while other team members have recommended spending more time on process documentation. This was solved by programming and working with process documentation side by side. This meant that we could keep everyone motivated, and that everyone could work with the project aspects that interested them the most.

4.9.4 What would we change if we started over?

- Take more notes during the process, details are easily forgotten.
- Better understand what will affect the final grading.

4.9.5 Final words

All in all, working with the eCatch Kyst Pilot project has been very educational and fun, while it at the same time has been a lot of work. We think that this project has helped us to get more ready for the working life, and has helped us understand how it is to work with a real customer in the working life.

We would like to give special thanks to Balázs for being a central piece in the development of our product. Not only did he play a big role in improving the quality of the product, but also in guiding the other group members in the technological challenges we faced. If someone deserves a extra reward it is Balázs.

Finally, the team would like to express our gratitude to the people we have been working with and learning from, including customers, supervisors, and the academic staff.

References

[1] Attribute driven design. https://en.wikipedia.org/wiki/Attribute-driven_design/. Accessed: 2019-05-08.

[2] Average age of fishermen in norway. https://www.ssb.no/jord-skog-jakt-og-fiskeri/artikler-og-publikasjoner/kven-er-yrkesfiskaren-og-kvatener-han. Accessed: 2019-05-06.

[3] Cdn mozilla service worker api. https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API. Accessed: 2019-05-07.

[4] Firebase cloud functions. https://cloud.google.com/functions/. Accessed: 2019-05-09.

[5] Code coverage. https://codecov.io/. Accessed: 2019-03-14.

[6] Cypress. https://www.cypress.io/. Accessed: 2019-03-14.

[7] Different cloud vendors. http://techgenix.com/cloud-computing-vendors/. Accessed: 2019-05-07.

[8] Don norman user experience. https://www.nngroup.com/articles/definition-user-experience/. Accessed: 2019-03-14.

[9] Double diamond figure. https://stanwick.be/sites/default/files/inline-images/DT%20double%20diamond.png. Accessed: 2019-05-09.

[10] Enzyme. https://airbnb.io/enzyme/. Accessed: 2019-03-14.

[11] Es6 syntax github page. https://github.com/rse/es6-features/. Accessed: 2019-05-10.

[12] Firebase. https://firebase.google.com/. Accessed: 2019-03-14.

[13] Firebase firestore. https://firebase.google.com/docs/firestore. Accessed: 2019-05-09.

[14] Firebase offline data. https://firebase.google.com/docs/firestore/manage-data/enable-offline. Accessed: 2019-03-14.

[15] Fishing boats 15 m - directorate of fisheries. https://drive.google.com/file/d/1qUu6aC2YJohLOlAAp44j2hg-rj_ZBvcE/view?usp=sharing. Accessed: 2019-05-05.

[16] Fiskarlaget midt-norge. https://www.fiskarlaget.no/index.php/medlemslagene/fiskarlaget-midt-norge. Accessed: 2019-05-07.

[17] Gitflow. https://datasift.github.io/gitflow/IntroducingGitFlow.html. Accessed: 2019-03-14.

[18] Gitkraken. https://www.gitkraken.com/git-client. Accessed: 2019-03-14.

[19] Google on pwa. https://developers.google.com/web/progressive-web-apps/checklist. Accessed: 2019-05-08.

[20] Jest - javascript testing framework. https://jestjs.io/. Accessed: 2019-03-14.

[21] Jsdoc. http://usejsdoc.org/. Accessed: 2019-03-14.

[22] Kruchten, P. (1995). *Architectural Blueprints—The "4+1" View Model of Software Architecture*. Paper published in IEEE Software 12 (6) November 1995, pp. 42-50.

[23] Lighthouse. https://developers.google.com/web/tools/lighthouse/. Accessed: 2019-03-14.

[24] Lighthouse issue performance. https://github.com/GoogleChromeLabs/lighthousebot/issues/56. Accessed: 2019-05-09.

[25] Lighthouse v2 scoring guide. https://developers.google.com/web/tools/lighthouse/scoring. Accessed: 2019-05-07.

[26] Material ui. https://material-ui.com/. Accessed: 2019-05-10.

[27] Quality attributes iso. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010. Accessed: 2019-05-08.

[28] React. https://reactjs.org/. Accessed: 2019-03-14.

[29] Register endpoint. https://ecatch-kyst-beta.firebaseapp.com/register. Accessed: 2019-05-09.

[30] Reporting of catch. https://lovdata.no/forskrift/2009-12-21-1743/§7. Accessed: 2019-03-14.

[31] Risk management framework for information systems and organizations. https://doi.org/10.6028/NIST.SP.800-37r2. Accessed: 2019-05-08.

[32] Seafood. https://www.regjeringen.no/en/topics/food-fisheries-and-agriculture/fishing-and-aquaculture/rad-1/fiskeri-ny/internasjonalt-fiskerisamarbeid/rydde-internasjonalt/fish/id685828/. Accessed: 2019-05-02.

[33] Semantic versioning 2.0.0. `https://semver.org/`. Accessed: 2019-03-14.

[34] Software architecture in practice. *Software Architecture in Practice, 3rd Ed.* Massachusetts: Pearson Education, Inc.

[35] Sustainable fishing. https://www.regjeringen.no/globalassets/upload/fkd/brosjyrer-og-veiledninger/folder.pdf. Accessed: 2019-05-02.

[36] The directorates datawishes. https://www.fiskeridir.no/Yrkesfiske/Nyheter/2018/0918/OEnskmer-data-fra-fiskeflaaten. Accessed: 2019-05-02.

[37] Toggl. www.toggl.com/. Accessed: 2019-04-24.

[38] Top 3 frameworks web. https://hackr.io/blog/top-10-web-development-frameworks-in-2019. Accessed: 2019-05-07.

[39] Travis ci. `https://travis-ci.org/`. Accessed: 2019-03-14.

[40] User experience. https://www.iso.org/standard/38009.html. Accessed: 2019-05-06.

[41] User interface. https://www.sis.se/api/document/preview/912053/. Accessed: 2019-03-14.

[42] Wakatime. https://wakatime.com/vs-code. Accessed: 2019-04-24.

[43] Wizard of oz - usability body of knowledge. `https://www.usabilitybok.org/wizard-of-oz`. Accessed: 2019-05-05.

A Definition of roles and responsibility

The main responsibility for meeting-reports:

- Responsible for finding a member of the group to write down every discussion, solution, and challenge mentioned in a meeting.
- Ensure that every document is orderly and easy to read.
- Ensure that the naming of documents from meetings is logical.
- Responsible for preparing the meeting agenda before every meeting.

Leader:

- Responsible for the execution of meetings.
- Is responsible for pushing the project forward and having an overview of how the project is going.
- Should have insight into team members tasks and responsibilities.
- Should arrange social team building events.

Customer contact:

- Responsible for the contact between all customers and the group, and reporting to the group of important messages.
- Is responsible for always being up-to-date on matters that need input from the customer.

Scrum:

- Responsible to make sure every team member understands Scrum.
- Make sure Scrum is used properly and the chosen elements are used.
- Make sure all documents regarding are in place, like Sprint Retrospective and Sprint Review.

Test lead:

- Responsible for the overall test-culture of the group, and provide assistance in cases where decisions need to be made.
- Overview of tools for testing and how they work.

- Overview of what parts of the application that has been tested.

Architecture lead:

- The main responsibility for the architecture of the application.
- The main responsibility for the documentation of the application architecture.
- Responsible for informing the rest of the group about architectural status.

Security lead:

- Ensure all reasonable risks connected to the project are identified and addressed.
- Ensure security events/incidents are being taken care of within a reasonable time.
- Verify that the group develops a security culture.
- The main responsibility for the security report.

Frontend:

- Ensure that progress is being made on frontend according to plan.
- Responsible for the component structure in the frontend.
- Responsible for code convention, a clean and understandable code in the frontend.
- Responsible for informing the rest of the group on frontend progress and matters.

UX lead:

- The main responsibility for the planning and execution of user interviews.
- Designing and deciding the optimal usability for the application.
- Ensure that the frontend looks as decided by the group.
- The main responsibility for implementing changes from user testing.

Responsive UI lead:

- Ensure that the UI works as intended on all platforms and devices.

Backend:

- Responsible for server and possibly database for the application.
- Responsible to document what backend supports.
- Responsible for all storage of data.
- Responsible for safe and reliable storage.

Git / build:

- Make sure that members use Git by convention (Git Flow).
- Ensure that our Git repository always work.
- Ensure that all git extensions work and are relevant.
- Provide assistance if members are having questions or trouble related to Git.

B Sprint plan

	ID	Beskrivelse	Viktighet	Goal	Comments
	31	The fisher should be able to use the application offline	0	Have an mvp product. By this we mean that the application should send a test message and the application should be able to format and send the 3 messages required.	
	23	The fisher can rely that the message gets sent	0,49		
	8	The Directorate should know the messages have integrity	7		
	13	The fisher needs to know if the request has been delivered and is ok	7		
	14	The fisher should be able to report leaving the dock/shore	10		
	22	The fisher can send his messages without thinking about authentication of the message	14		
	15	The fisher should be able to change or cancel their fishing estimation report	32		
	25	Dualog needs to get sensible information from error messages	67,5		
	26	As a customer i would like the application to reflect the company design colors.	108		
	20	The fisher should be able to choose between light and dark mode	176		
Sprint 1	12	The fisher should be able to choose between norwegian, english and have support to later add other languages	276,5		
	31	The fisher should be able to use the application offline	0	Frontend should be able to take input from users and send complete messages.	Some user stories didnt get fully done the last Sprint so we pushed those into the second Sprint. We also added some more stories.
	23	The fisher can rely that the message gets sent.	0,49		
	8	The Directorate should know the messages have integrity	7		
	13	The fisher needs to know if the request has been delivered and is ok	7		
	14	The fisher should be able to report leaving the dock/shore	10		
	15	The fisher should be able to change or cancel their fishing estimation report	32		
	25	Dualog needs to get sensible information from error messages.	67,5		
Sprint 2	21	The fisher should be able to get an overview over the fish currently on his boat	13		
	8	The fiskeridirektoratet should know the messages have integrity	7	The application should follow the fishing departments rules for how messages should be sent, and should be able to send one DEP message at the time, multiple DCA messages, and 1 POR message.	The unfinished user stories from the last Sprint are pushed into the third Sprint. We also added some more stories that focus mostly on the functionality of messages.
	15	The fisher should be able to change or cancel their fishing estimation report	32		
	21	The fisher should be able to get an overview over the fish currently on his boat	13		
	17	The fisher should be able to get an easy overview over previous messages sent	76		
	10	The fisher should be able to read realtime GPS location in the application	173		
	5	The user should be able to fill out the form offline (one time)	1		
	16	If the fisher stays out overnight, a message should be sent before midnight	34		
	4	The fisher should be able to preset settings to their specific situation	42,5		
Sprint 3	9	The fisher should be able to set GPS location from the application	6,5		
	15	The fisher should be able to change or cancel their fishing estimation report	32	Have working functionality for presets in the application to reduce clicks and ease use.	Unfinished user stories from the last Sprint are pushed into the fourth Sprint.Also added som more stories that fit with the chosen Sprint goal.
	21	The fisher should be able to get an overview over the fish currently on his boat	13		
	16	If the fisher stays out overnight, a message should be sent before midnight	34		
	36	As a fisher, i would want the application to know the law (updated quotas), so i don't need to think about it.	22		
	4	The fisher should be able to preset settings to their spesific situation	42,5		
	35	As a fisher, i would want push-warnings when i need to do something	54		
	25	Dualog needs to get sensible information from error messages.	67,5		
	3	The fisher should be able to quickly and easily report fishing	4,5		
	41	As a fisher i want the application to help me send messages on time	45		
	7	The fisher should only have to log in one time	60		
Sprint 4	42	As a fisher i want the application to suggest standard settings that are feasible	91		
	15	The fisher should be able to change or cancel their fishing estimation report	32	Improve upon the existing UX and make it harder to make mistakes (more validation and guiding)	
	21	The fisher should be able to get an overview over the fish currently on his boat	13		
	16	If the fisher stays out overnight, a message should be sent before midnight	34		
	36	As a fisher, i would want the application to know the law (updated quotas), so i don't need to think about it.	22		
	4	The fisher should be able to preset settings to their specific situation	42,5		
	35	As a fisher, i would want push-warnings when i need to do something	54		
	3	The fisher should be able to quickly and easily report fishing	4,5		
	2	The fisher must be able to use the application with little to no technological background	5		
	1	The fisher must be able to use the application with (big gloves and) big fingers	16		
	39	As a fisher, i want the application to guide me through the report process, as i am not good with data technology	16		
	6	The fisher should be able to use the application on all his devices	24		
	33	The fisher should be able to understand, and use the app without complications	30,5		
	19	The fisher should be able to select multiple fish types when reporting.	48		
	11	The fisher should be able to store information locally on the device	17		
Sprint 5	28	The fisher should feel safe using the application	113,5		
	25	Dualog needs to get sensible information from error messages.	67,5		
	41	As a fisher i want the application to help me send messages on time	45		
	32	The fisher should not have to use alot of MB using the application	61,5		
	30	The technichians should recieve bug reports from the app	174		
	18	The fisher should get an overview over some helpful statistics/models	243		
Backlog	29	The fisher should be able to view his position using the AIS systems.	380		

C Business context

Business Context

What is a business goal?
The business has ambitions, motivations and reasons to exist in the world. We capture this into a list of business goals. The purpose of this is to have background information to make decisions and prioritizing time and effort. In this project we specifically want to use this to determine architectural patterns and map out security risks.

What is a business assets?
Business assets are all assets that belong to the company. These are a part of the security landscape and will help us connect real world risk to risks in the application.

Example from imaginary Pokedex Company
Business Assets
ID Description
BA1 Personal user information (email address, name, etc.)
BA2 Users assets (pokemon, reports, images, products, etc.)
BA3 Emails and their contents
BA4 Admin credentials
BA5 Analytic data based on user activity
BA6 The application source code
BA8 Company reputation (trust, quality of product, trademark)
BA9 Physical Pokedex's (and everything related)

Business Goals
ID Description
BG1 Gain more customers/users
BG2 Gain reputation
BG3 Be the largest Pokedex company in the world
BG4 Meet service level agreements
BG5 Deliver a reliable product of the best quality
BG6 Make a product that everyone is able to use.
BG7 Reduce cost of further development
BG8 Increase revenue

Questions for the customer - Dualog
These questions are meant to spawn discussion and insight into the business and uncover business goals and business assets. Dualog do not need to answer the questions, we understand there might be need for confidentiality. The accuracy and quality of the answers will impact the quality of the end product.

Questions aimed at business assets

What are your physical business assets?

- Servers at dualog
- Custom hardware

What are your digital business assets?

- Fishing data

In short what scenarios or events would end the organization?

- If the requirement for reporting fish cease existing

Questions Aimed at Business Goals

What is the purpose of your organization?

- Uphold legally required electronic fishing reporting for professional fishers, fishing fleets and other actors in the marine coastal environment.

What real world problems do you solve?

- Securing and regulating the fishing resources in the ocean, through reporting and registration of fishing operations.
- Make sure Norwegian ocean resources stays in the hands of the Norwegian people, for now and for future generations.

What are the short term goals for your organization?

- Extend our product line to include smaller fleets with boats less than 15 meters.

What are the long term goals for your organization?

- Create valuable software for fishers, and continuously simplify the reporting process.

What do you want your customers to think about your organization?

- That we create sustainable, reliable and always available when the customer needs them.

How do you want your customers to experience your product(s)/ service(s)?

- Reliable and useable.

What are the economic goals for your organization?

- Reduce support cases
- Reduce product maintenance costs
- Expand into a new marked
- Reduce technical debt

Who are your organization accountable for? And what responsibilities do you need to overhold?

- Accountable for the product that legally reports fishing for the fishers. Reporting is required for them to do their job.
- Norwegian Directorate of Fisheries. Make sure the authorities has the information they need.
- Boat owners and office workers. Make sure the owners and office workers of fishing companies has an overview over fishing activity onboard all their boats.

Who are your stakeholders?

- Fishers, Fiskeridirektoratet, Dualog, board members, Bekk, dualog service, dualog sales, dualog code maintainers.

Please prioritize these quality attributes from most important (1) to least important (8) (More info on the ISO/IEC FCD 25010 product quality standard).

Pri	Quality Attribute
3	Functional Suitability - This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.
7	Performance efficiency - This characteristic represents the performance relative to the amount of resources used under stated conditions.
5	Compatibility - Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment.
4	Usability - Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.
2	Reliability - Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.
1	Security - degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.
6	Maintainability - This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements.
8	Portability - Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

The Business assets:

1. Servers at dualog
2. The reputation of dualog with the fishers and the state
3. Keys for authentication
4. Fishing operations data (messages)
5. Offices
6. Software products
7. Custom hardware for boats
8. Boat / fishing insights
9. Customer insights

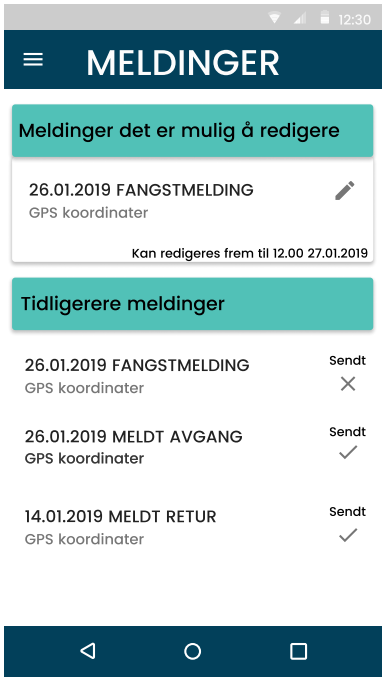
10. Good relation to the state actor
11. A business deal with the state

The business goals:

1. Uphold legally required electronic fishing reporting for professional fishers, fishing fleets and other actors in the marine coastal environment.
2. Securing and regulating the fishing resources in the ocean, through reporting and registration of fishing operations.
3. Make sure Norwegian ocean resources stays in the hands of the Norwegian people, for now and for future generations.
4. Extend the product line to include a product for smaller boats, less than 15 meters.
5. Create valuable software for fishers, and continuously simplify the reporting process.
6. Have a reputation of making sustainable and reliable products, that's always available when needed.
7. Have a pleasant user experience with the use of our products, that's promotes reliability and usability.
8. Reduce support cases
9. Reduce product maintenance costs
10. Expand into a new marked
11. Reduce technical debt
12. Meet the needs and expectation of our stakeholders.
13. Meet business agreements with the authorities.
14. Become the preferred supplier of electronic catch reporting
15. Become the market leader with a 70% market share

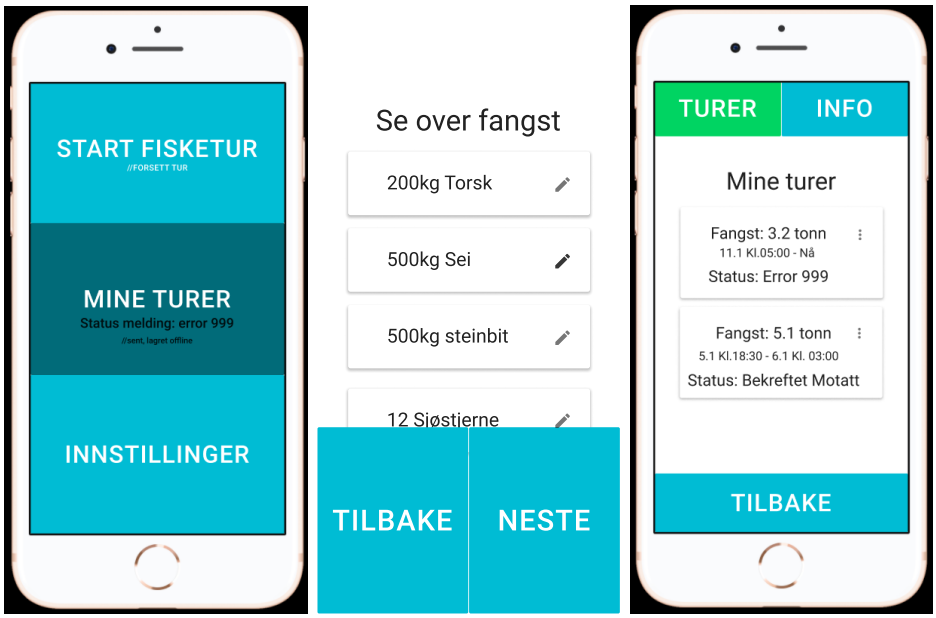
D Figma prototypes

D.1 Guided style



D.2 3 row

D.2.1 Images



Start page Step by step Trip Info

D.2.2 Design concept

Problem:
Simplify the process of reporting fishing given the specification from customer. This is done by filling out messages with multiple fields. This can be thought of as filling out the same types of form over and over again. The solution needs to be suited the user group of fishers.

Solution:
The solution is to focus on user interaction. It focuses on the types of interaction the user can have with the application. The design is follows the principle of designing from the user's perspective and not the technical requirements.

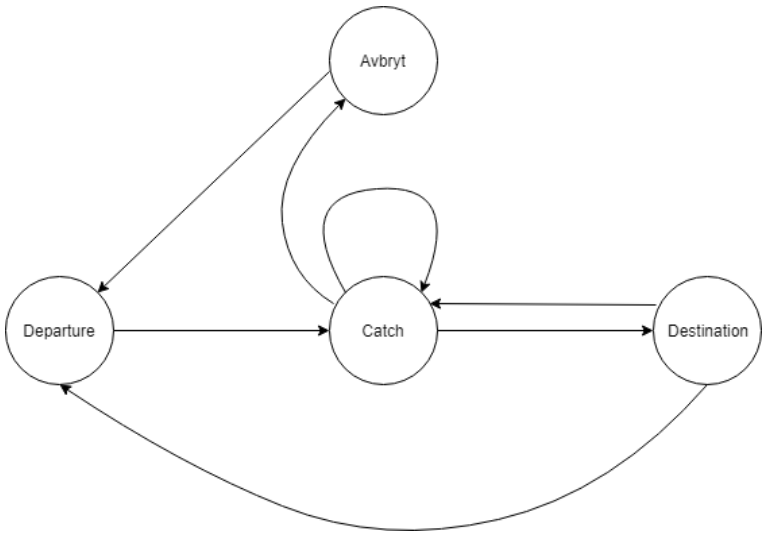
The solution focuses on reducing the time and clicks it takes to complete the forms. There is one obvious way to do this, presets. The user fills in the information in a "profile" or "settings" object. The secondary objective of the design is to have a continuous overview over the state of the process.

The problem can be boiled down to 3 types of user needs. The user needs to fill out the forms. Here is where the repetitive process of filling out forms.

The user needs to know what's going on within the application. A need for information about the state of the application is necessary to avoid mistakes and misunderstandings. The user needs to be able fill in values that always or often are the same.

Based on these three needs we separate our application into 3 clearly defined areas of use, with 3 different uses. These are barrier that should not be crossed as this will confuse the user. There might be a few exceptions to this, for example when you want a shortcut (jump to) to another area of the application. The user interface for this design starts with a login, than the main screen separated into 3 sections or rows. You enter a section and then you are able to use whatever functionality layers in that section. When you are done you can go back to the main screen and enter a new section.

The "form filling process" section The form filling process should be fast and intuitive, with as few decision points as possible. In the process the user needs to fill in a minimum of three messages as specified in the technical requirements. These represent departure information, catch information and destination port information. There can be multiple catch messages, but only one departure and destination messages. The way this is solved is to implement a state machine. Where only one of the states can be active at one time. This ensures that the user is never confused about what to fill in next in the form filling process.



As displayed in the sketch the catch messages needs to be able to go in a loop. This will be solved by having a list of catch messages, where you can either add a catch message or go to the destination state. Each state has a "form" to complete. A form is a something we can expect all users to have some experience with, ether paper or digital. We can exploit this familiarity and make the learning curve for the application better. Each form will be scrollable and displaying a maximum of 3 elements.

Information section This section contains information about the user might want to see. This must include information about the trips a user takes (the process and state) and it can include statistics and data that might be useful (calculated and graphically processed information). If both is implemented a top level menu bar choose the selected information and displays it.

The process and state information display is displayed as trips in a list format. The reason trips are used to segment the data is because it's concept tied to the users daily activity. It also fits well with the message cycle, where every trip needs a departure, a destination and one or more catch messages. Each list element is a trip, if you click the information icon the trip will appear in a pop up display. Here you will see all the messages that are created, together with their status, some information and possible actions. There can be two actions applied to the messages, which depends on the business logic, one can edit the message and view it. The view of the message will be a technical view of the message data and fields.

The statistics display shows info that the user finds useful. This can for

example be a heatmap over the best catching spots. This means using the data from the messages, storing them and visualizing them. It is important to note that this has nothing to do with the core functionality of the application (creation and sending messages), but is a way to add value to the application with the data we already have.

There will be a big back button covering the bottom of the screen, that leads back to the main screen.

The preset section In this section the user can input repetitive information. It also includes static information from the database.

The information that can be preset is listed as elements were maximum 3 items are displayed at once. The maximum limit is to reduce information "shock", where to much information results in a longer time to execute an action. The list is scrollable. The information here is used to preset the information in the "form filling process".

Why is there static information in the preset section and not the information section where information is supposed to be? Because one would assume that the information was changeable. The information is things like name, radio call, registration, etc. If the user wanted to change these things they would likely look under settings and not under status. In some way this settings are preset, but not in the application. These are preset by our customer in their IT system, as a part of their validation. With the information there will be a message to call the customer service center to change this information. So it's a preset, it is just not set by the user, but the technical staff at our customer.
Application feedback A big application feature in this design is feedback to the user. We write this as an own point as the previous sections cover functionality. The main page have two dynamic displays. The display button of the "form filling button" should give feedback about which state the final state machine is in; Start trip, continue trip. The display button for the information section should give feedback about the messages status; are some saved offline, are everything sent, is there an error, etc.

All the section should focus heavily on user feedback.

E Translations

In cooperation with the customer we constructed a translation list over important words and concepts. Bellow is a example of the translation list we used during the project.

E.1 Norwegian and English translations

	Norsk		Engelsk	Kommentarer
DEP	Avreisehavn	Avgangshavn	Departureport	
	Velg havn	Avgangshavn	Choose departureport	
	Velg avreisetidspunkt	Tidspunkt	Choose time of departure	
	Fartøyets hovedaktivitet	Hovedaktivitet	Select the vessels main activity	
	Skriv inn antatt dato og tidspunkt for fisk	Estimer tidspunkt for fiskestart	Choose estimated time for fishingstart	
	Velg blandt dine fiskeplasser	Velg fiskefelt	Select from your stored fishingspots	
	Legg inn hvor du skal fiske	Hvor skal du fiske	Add where you are going to fish	
	Legg til eventuell kvantum ombord	Kvantum om bord	Fill in quantity of fish onboard	
	Se over meldingen		Look over the message	
DCA part 1	Når startet du å fiske?	Fiskestart	When did you start to fish	
	Legg inn hvor du skal fiske	Fiskefelt	Where are you fishing?	
	Velg fiskeredskapet du bruker	Redskap	Select fishingtool	
	Velg maskevidde på garnet	Maskevidde	Select mesh on tool	
	Velg målart	Målart	Select desired catch	
	Start fiske		Start fishing	
DCA part 2	Når sluttet du å fiske?	Fiskeslutt	When did you stop fishing?	
	Har du hatt noen redskapsproblemer?	Redskapsproblemer	Have you had any fishingtoolproblems?	
	Legg til fangst		Add catch	
	Fisketype	Fiskeslag	Fishtype	
	Legg inn vekten	Vekt	Add weight	
	Registrer fangst		Register catch	
POR	Vennligst legg inn havnen du reiser til	Anløpshavn	Select destinationport	
	Hjemreisehavn	Anløpshavn	Port of arrival	
	Legg inn forventet ankomsttidspunkt	Estimert ankomsttid	Choose estimated time of arrival	
	Vennligst legg inn mottaket du reiser til	Mottaksanlegg	Choose which fish landing you arrive at	
	Legg inn total fangst sortert på fisketype	Legg inn fangst	Add total catch sorted on fishtype	Sorteringa er vel noe om kan gjøres automatisk?
	Bekreft og lever fangstmelding	Bekreft og send melding	Confirm and deliver catch report	
	Fangstmelding klar		Catch report ready	
Mine meldinger	Meldinger det er mulig å redigere		Messages that you can edit	
	Kan redigeres frem til <tidspunkt>		Are editable til <time>	
Profil	Kontakt Dualog for å endre informasjonen. Tlf: 92084000		Contact Dualog to change your personal information. Phone: +47 987 65 432	
	Min anløpshavn		My home port	
	Mitt landingsanlegg		My landing site	
	Mine fiskefelt		My fishingspots	
	Min avgangshavn		My departure port	
	Mine fangstarter	Mine arter	My typical fishingtypes	
	Min standard kvote	Min standardkvote	My standard quota	
	Mine redskaper		My fishingtools	
	Her kan du redigere profilen din, legge t	`Rediger informasjon`	Here you can edit your profile, add information and verify that all is correct	
	Bytt til nattmodus for å gjøre det bedre å	`Nattmodus`	Switch to nightmode to improve the experience of using the application when its	
Om oss	**Trenger du hjelp? Kontakt Dualog Fisknett AS på support@dualog.com eller 92084000.**			

E.2 Fishing terminology

Snurrevad - Purse seine
Linefiske - Longline fishing
Juksa - Handlining
Garn - Fishing net
dorg - Trolling
Trål - Trawling
Fangst - Catch
Fangstmelding - Catch report
Meld avgang - Report departure
Mine meldinger - My messages
Meld fangst - Report catch
Meld hjemreise - Report your return journey
Breddegrad - Latitude
Lengdegrad - Longitude
Slepepose - Towing bag
Avreisehavn - Departure port
Hjemreisehavn - Return home port
Torsk - Cod
Skrei - Atlantic cod
Sei - Saithe / Pollock
Sild - Herring
Makrell - Mackerel
Kongekrabbe - King crab
Teine - pot
Piggvar - Turbot
Kveite - Halibut
Reke - Shrimp
Flyndre - Atlantic halibut
Steinbit - Catfish
Breiflabb - Monkfish
Rognkjeks - Lumpfish
Uer - Redfish
Ål - Eel
Taskekrabbe - Crabs
Hummer - Lobster
Kolmule - Blue whiting
Hyse - Haddock
Brosme - Cusk / Tusk
Lange - Common ling
Blålange - Blue ling
Lyr - Atlantic pollock
Lysing - European hake
Hvitting - Whiting / Merling
Hestmakrell - Atlantic horse mackerel
Brisling - bristling, brisling, garvie, garvock, Russian sardine, russlet, skipper or whitebait
Blåkkveite - Greenland halibut or Greenland turbot
Rødspette - European plaice
Tunge - Common sole
Smørflyndre - Witch
Slettvar - Brill

F Decomposition of messages

DEP / DCA / POR / RET meldinger

Dataen i meldingene kan settes med:

- data fra authentisering (**DBD**) //i dev er dette firebase, i prod må det gjøres om til dualog sine databaser
- lagres i statisk profil (av fisker)(**STK**)
- skrives inn under fiske (av fisker)(**KIS**)
- genereres av applikasjonen (**APP**)

Dette beskriver alle kildene av hvor dataen kommer fra, beskriver ikke dataflyt.

DEP

SR	Start	APP - genereres
TM	Meldingstype	APP - genereres
RN	Meldingsnummer	APP - genereres
AD	Mottaksland	Hardkode til NOR //spør kunde
RC	Fartøyets radiokallesignal	DBD - fra authentisering
NA	Fartøyets navn	DBD - fra authentisering
XR	Fartøyets registreringsnummer	DBD - fra authentisering
MA	Skippers navn	DBD - fra authentisering
DA	Dato sendt ÅÅÅÅMMDD (UTC)	APP - genereres
TI	Tidspunkt sendt TTMM (UTC)	APP - genereres
PO	Avgangshavn NO = Land, TRD = havn	APP/STK/KIS - fra gps, i profil (default), kan endres underveis
ZD	Dato for avgang, antar ÅÅÅÅMMDD (UTC)	APP/KIS - genereres, kan endres underveis
ZT	Tidspunkt for avgang, antar TTMM (UTC)	APP/KIS - genereres, kan endres underveis
PD	Antatt dato for fiskestart, antar ÅÅÅÅMMDD (UTC)	APP/KIS - genereres, kan endres underveis
PT	Antatt tidspunkt for fiskestart, antar TTMM (UTC)	APP/KIS - genereres, kan endres underveis
LA	Latitude (Står på slack, antatt fiske start)	STK/KIS - i profil, kan endres underveis
LO	Longitude (Står på slack, antatt fiske start)	STK/KIS - i profil, kan endres underveis
AC	Fartøyets hovedaktivitet	KIS(/STK) - velges av fisker (m/ default)
DS	Målart	KIS(/STK) - velges av fisker (m/ default)

OB	Kvantum om bord	KIS/(APP?) //spør kunde om den alltid er null
ER	End	APP - genereres

DCA

SR	Start	APP - genereres
TM	Meldingstype	APP - genereres
RN	Meldingsnummer	APP - genereres
MV	Meldingens versjonsnummer	APP - genereres
AD	Mottaksland	APP/KIS/DBD - fra gps, i profil (default), kan endres underveis, basert på api endpoint
RC	Radiokallesignal	DBD - fra authentisering
NA	Fartøyets navn	DBD - fra authentisering
XR	Fartøyets registreringsnummer	DBD - fra authentisering
MA	Skippers navn	DBD - fra authentisering
DA	Dato sendt	APP - genereres
TI	Tidspunkt sendt	APP - genereres
QI	Fisketillatelse / hvilken type kvote	KIS/STK - velges av fisker (m/ default)
AC	Fartøyets hovedaktivitet	KIS/STK - velges av fisker (m/ default)
TS	??	empty (not null, empty)
BD	Dato for start av fiskeoperasjon	APP/KIS - genereres av gps, kan endres
BT	Tidspunkt for start av fiskeoperasjon	APP/KIS - genereres av gps, kan endres
ZO	Sone der fiskeoperasjonen starter	APP/KIS - velges av fisker (default fra gps)
LT	Posisjonens bredde ved start av fiskeoperasjon	APP/KIS - genereres, kan endres
LG	Posisjonens lengde ved start av fiskeoperasjon	APP/KIS - genereres, kan endres
GE	Redskapskode	STK/KIS - velges av fisker fra liste (m/ default)
GP	Redskapsproblemer	KIS - velges fra liste (default ingen problem)
XT	Posisjonens bredde ved avslutning av fiskeoperasjon	APP/KIS - fra gps, kan legges inn manuelt
XG	Posisjonens lengde ved avslutning av fiskeoperasjon	APP/KIS - fra gps, kan legges inn manuelt
DU	Fiskets varighet i minutter	KIS/APP - valgt fra bruker (default slutt nå, start nå-mininus 2 timer)

CA	Total fangst fra denne fiskeoperasjonen fordelt på fiskesort (FAO fiskesortkode)(SN) i kilo rund vekt (WT). Parvis angitt.	KIS - logges av fisker
ME	Minste maskevidde på redskap i millimeter (mm)	STK/KIS - fra profil, kan endres
GS	Redskapsspesifikasjon 1 = enkeltrål, 2 = dobbeltrål, 3 = trippeltrål. 4 = mer enn tre tråler	STK/KIS - fra profil, kan endres
ER	End	APP - genereres

POR

SR	Start	APP - genereres
TM	Meldingstype	APP - genereres
RN	Meldingsnummer	APP - genereres
AD	Mottaksland	APP/KIS/DBD - hardkodes for nå NOR
RC	Radiokallesignal	DBD - fra authentisering
NA	Fartøyets navn	DBD - fra authentisering
XR	Fartøyets registreringsnummer	DBD - fra authentisering
MA	Skippers navn	DBD - fra authentisering
DA	Dato sendt	APP - genereres
TI	Tidspunkt sendt	APP - genereres
PO	Anløpshavn ISO alfa 2 + 3 bokstaver havnekode	APP/STK/KIS - valgt av fisker (m/ default)
PD	Dato for havneanløp	KIS/APP - velges av fisker (m/ default eller gps)
PT	Tidspunkt for havneanløp	KIS/APP - velges av fisker (m/ default nå tidspunkt)
OB	Kvantum om bord fordelt på fiskesort (FAO fiskesort kode) i kilo rund vekt.	APP/KIS - genereres fra DCA meldinger, kan endres av bruker
LS	Navnet på landingsanlegget	STK/APP/KIS - fra profil, eller GPS, kan endres av fisker
KG	Kvantum som skal landes fordelt på fiskesort(FAO fiskesortkode) i kilo rund vekt	KIS/APP - valgt av fisker (default alt)
ER	End	APP - genereres

Svarmeldinger

SR	Start
TM	Meldingstype
RN	Meldingsnummer
FR	Fra
RC	Fartøyets radiokallesignal
RS	Meldingsstatus (ACK/NAK)
MV	Meldingens versjonsnummer
RE	Returnert feilmeldingsnummer
DA	Meldingsdato
TI	Meldingstidspunkt
ER	End

Oppsumering til “3-row”

Innstillinger = STK

Instillinger

PO	Avgangshavn NO = Land, TRD = havn	APP/STK/KIS - fra gps, i profil (default), kan endres underveis
LA	Latitude (Står på slack, antatt fiske start)	STK/KIS - i profil, kan endres underveis
LO	Longitude (Står på slack, antatt fiske start)	STK/KIS - i profil, kan endres underveis
AC	Fartøyets hovedaktivitet	KIS(/STK) - velges av fisker (m/ default)
DS	Målart	KIS(/STK) - velges av fisker (m/ default)

QI	Fisketillatelse / hvilken type kvote	KIS/STK - velges av fisker (m/ default)

GE	Redskapskode	STK/KIS - velges av fisker fra liste (m/ default)

ME	Minste maskevidde på redskap i millimeter (mm)	STK/KIS - fra profil, kan endres
GS	Redskapsspesifikasjon 1 = enkeltrål, 2 = dobbeltrål, 3 = trippeltrål. 4 = mer enn tre tråler	STK/KIS - fra profil, kan endres

PO	Anløpshavn ISO alfa 2 + 3 bokstaver havnekode	APP/STK/KIS - valgt av fisker (m/ default)

LS	Navnet på landingsanlegget	STK/APP/KIS - fra profil, eller GPS, kan endres av fisker

Velges gjennom prosessen / input = KIS

DEP

PO	Avgangshavn NO = Land, TRD = havn	APP/STK/KIS - fra gps, i profil (default), kan endres underveis
ZD	Dato for avgang, antar ÅÅÅÅMMDD (UTC)	APP/KIS - genereres, kan endres underveis
ZT	Tidspunkt for avgang, antar TTMM (UTC)	APP/KIS - genereres, kan endres underveis

PD	Antatt dato for fiskestart, antar ÅÅÅÅMMDD (UTC)	APP/KIS - genereres, kan endres underveis
PT	Antatt tidspunkt for fiskestart, antar TTMM (UTC)	APP/KIS - genereres, kan endres underveis
LA	Latitude (Står på slack, antatt fiske start)	STK/KIS - i profil, kan endres underveis
LO	Longitude (Står på slack, antatt fiske start)	STK/KIS - i profil, kan endres underveis
AC	Fartøyets hovedaktivitet	KIS(/STK) - velges av fisker (m/ default)
DS	Målart	KIS(/STK) - velges av fisker (m/ default)
OB	Kvantum om bord	KIS/(APP?) //spør kunde om den alltid er null

AD	Mottaksland	APP/STK/KIS/DBD - fra gps, i profil (default), kan endres underveis, basert på api endpoint

DCA

QI	Fisketillatelse / hvilken type kvote	KIS/STK - velges av fisker (m/ default)
BD	Dato for start av fiskeoperasjon	APP/KIS - genereres av gps, kan endres
BT	Tidspunkt for start av fiskeoperasjon	APP/KIS - genereres av gps, kan endres
ZO	Sone der fiskeoperasjonen starter	APP/KIS - velges av fisker (default fra gps)
GE	Redskapskode	STK/KIS - velges av fisker fra liste (m/ default)
GP	Redskapsproblemer	KIS - velges fra liste (default ingen problem)
XT	Posisjonens bredde ved avslutning av fiskeoperasjon	APP/KIS - fra gps, kan legges inn manuelt
XG	Posisjonens lengde ved avslutning av fiskeoperasjon	APP/KIS - fra gps, kan legges inn manuelt
DU	Fiskets varighet i minutter	KIS/APP - valgt fra bruker (default slutt nå, start nå-mininus 2 timer)
CA	Total fangst fra denne fiskeoperasjonen fordelt på fiskesort (FAO fiskesortkode)(SN) i kilo rund vekt (WT). Parvis angitt.	KIS - logges av fisker
ME	Minste maskevidde på redskap i millimeter (mm)	STK/KIS - fra profil, kan endres
GS	Redskapsspesifikasjon 1 = enkeltrål, 2 = dobbeltrål, 3 = trippeltrål.	STK/KIS - fra profil, kan endres

	4 = mer enn tre tråler	

POR

PO	Anløpshavn ISO alfa 2 + 3 bokstaver havnekode	APP/STK/KIS - valgt av fisker (m/ default)
PD	Dato for havneanløp	KIS/APP - velges av fisker (m/ default eller gps)
PT	Tidspunkt for havneanløp	KIS/APP - velges av fisker (m/ default nå tidspunkt)
OB	Kvantum om bord fordelt på fiskesort (FAO fiskesort kode) i kilo rund vekt.	APP/KIS - genereres fra DCA meldinger, kan endres av bruker
LS	Navnet på landingsanlegget	STK/APP/KIS - fra profil, eller GPS, kan endres av fisker
KG	Kvantum som skal landes fordelt på fiskesort(FAO fiskesortkode) i kilo rund vekt	KIS/APP - valgt av fisker (default alt)

Må displayes

AD	Mottaksland	Hardkode til NOR //spør kunde
RC	Fartøyets radiokallesignal	DBD - fra authentisering
NA	Fartøyets navn	DBD - fra authentisering
XR	Fartøyets registreringsnummer	DBD - fra authentisering
MA	Skippers navn	DBD - fra authentisering
DA	Dato sendt ÅÅÅÅMMDD (UTC)	APP - genereres
TI	Tidspunkt sendt TTMM (UTC)	APP - genereres

G Records from phone interviews and user tests

G.1 User testing

Brukertest spørsmålsark

Formålet med testen er for oss utviklere å oppdage logiske feil, og høre tilbakemeldinger ang hva de liker, hva som er uklart, og hva de ikke liker / mener er feil. Hvis de sier noe er feil, spør gjerne utdypende spørsmål om hvorfor og hvordan det kan løses. Hvis det er noe de savner i løsningen, er dette noe du må se om passer inn i oppgavebeskrivelsen vi har i dag.

Innlogging:

- Han vil ikke logge inn hver gang. Dette har vi allerede tatt hensyn til.

Dark-mode:

- knappene på bunnen av siden er små. Funker ikke så bra.
- Trykker på dashboard, ikke obv. at det er på profil.
- Trykker på nattmodus, men ikke på knappen.
- For lang avstand mellom knapper og teksten.
- Tips: finnes symbol på nattmodus. bedre å ha en knapp som er mer tilgjengelig kanskje?

Språkendring:

- Gikk greit, litt vanskelig å trykke på knappen kanskje

knappene på bunnen fungerer dårlig.

Meld avgang:

- Jobber med favorittliste. (kanskje top 5 elrno). [fiskested]
- Han liker måten å velge tid på, bra.
- planlagt fiskested, fiskeaktivitet, litt kluss her.
- Siden scroller ikke ned når tastatur kommer opp. Må fikse dette.
- Ikke lett å forstå at teksten forsvinner når man starter å skrive.
- Huske hva du ikke har levert?
- Hvorfor får man spørsmål om å sende meldingen ? (modal). Han virker ikke helt glad for dette.
- Står fortsatt meld avgang etter at han har sendt en melding. Fungerer ikke… ganske krise.

Minst mulig trykking.

Meld fangst:

- Snakker litt om edgecases, det har vi ikke enda fiksa.
- Står tittel, men den gjør ikkeno, WIP.
- Skriftstørrelsen er for liten kanskje. Mer skille mellom der man kan endre og ikke endre ting. (overskrifter og felt).
- Han vil ha preset liste, det vet vi allerede. Tabellform ? bare legge inn?
- Han sliter med rekkefølgen. han vil ikke legge inn start og stopp, det skal være automatisk… how?
- redskap, tydeligvis mye med regelverk vi ikke vet om.

- Mer klaging på tastatur, ikke noe vi kan gjøre noe med.

meldingsside:

- ikke så klart at man kan trykke på DCA meldinger.
- Vil ha siste meldinger på dashboard og ikke på egen side. han vil ikke trykke på trips siden. Avgangs og fangstmeldings ack.
- Ha CDA, POR osv i parantes bak meld fangst osv.

POR:

- Fisk ombord burde komme opp automatisk.
- kommer alltid 0 før når man skriver inn tall, fikse dette.
- Vil gjerne ha tabelloversikt.

vil ha notification på når du får ack
havnekoder sier han ingenting. Må være navnene.

kan ikke redigere etter POR.

meldingslogg er fint, det å ha en switch kan være bra, men er ikke så enkelt å forstå.
Kanskje stå meldingslogg elrno istedenfor symbol.
Nummerere turer basert på årstall. Tur 1 2 3 osv.

Bra med preset fra forrige tur, men vil allikevel ha favorittliste på ting. Kan gjøre dette på forhånd.

kan gjerne ringe han og spørre om ting.

G.2 Phone interview summary

From a total of 6 interviews, this summary reflects the most important topics discussed:

- Varierende fiskeplasser basert på sesong, men gjerne faste plasser i løpet av sesongen
- Helårsfiskere, men mest på høst og vinter.
- Blandet erfaring med J-meldinger og reglene knyttet til disse.
- Fiskerne har generelt flere arter de fisker etter, men det er forskjellige fiskemetoder knyttet til de.
- De fleste tror de er ganske flinke til å estimere hvor mye de har fanget, men ingen tror de er innenfor de 10
- Ingen tror de skal ha noe problemer med å bruke smartdevices
- Alle fiskerne har tilgang til statistikk, men det er få som bruker det aktivt. De fleste opplyser at de går på erfaring.
- De fleste sier helst at appen skal være: Enkel, simpel, kontrollerende, rask og oversiktlig. Noen etterspør nytteverdi. Men i all hovedsak er det enkelhet og en simpelhet i arbeidet de vil ha.
- Dagbok med kalender, mannskapshåndtering, bifangstkategorier er det som går igjen i hva fiskerne ville hatt hvis de fikk ønske seg en funksjon
- Noen bytter fiskeredskaper utifra sesong, men som regel fisker de med samme hele tiden.
- De fleste vil bruke telefon.
- Fiskerne tror vi som utviklere ikke klarer å få gjort det enkelt nok, og ikke klarer å visualisere hvordan forholdene er ute på havet.
- Generelt er de fleste redde for et alt for strengt regelverk

H User manual

H.1 Registration

If you do not have a user already, you fist need to register. In your favorite browser, navigate to the temporary register page (https://ecatch-kyst-beta.firebaseapp.com/register) and enter your personal information.

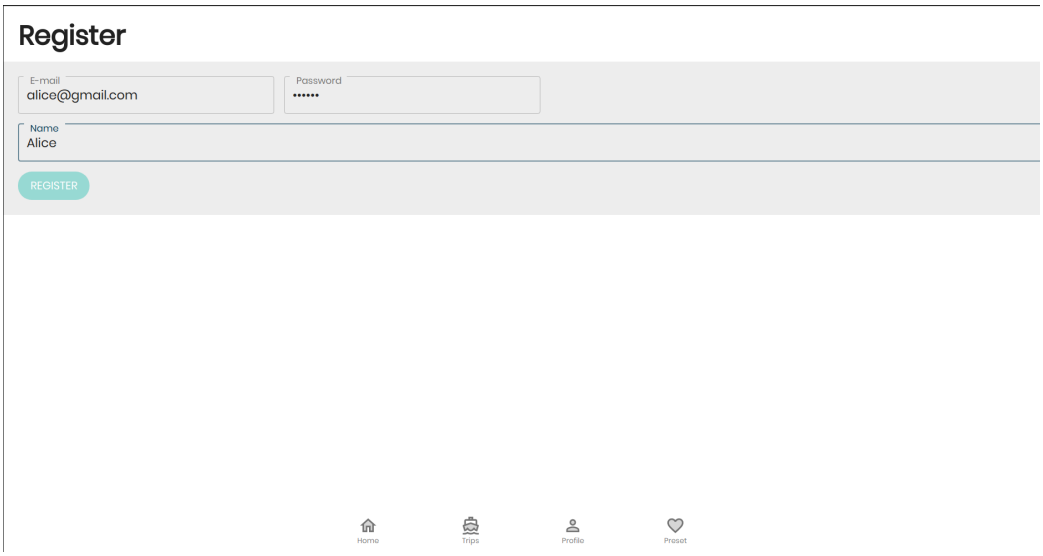


Figure 19: Register page with "Register" button

H.2 Logging in

If you already have a registered user, and you are not already logged in, please navigate to the application (https://ecatch-kyst-beta.firebaseapp.com).

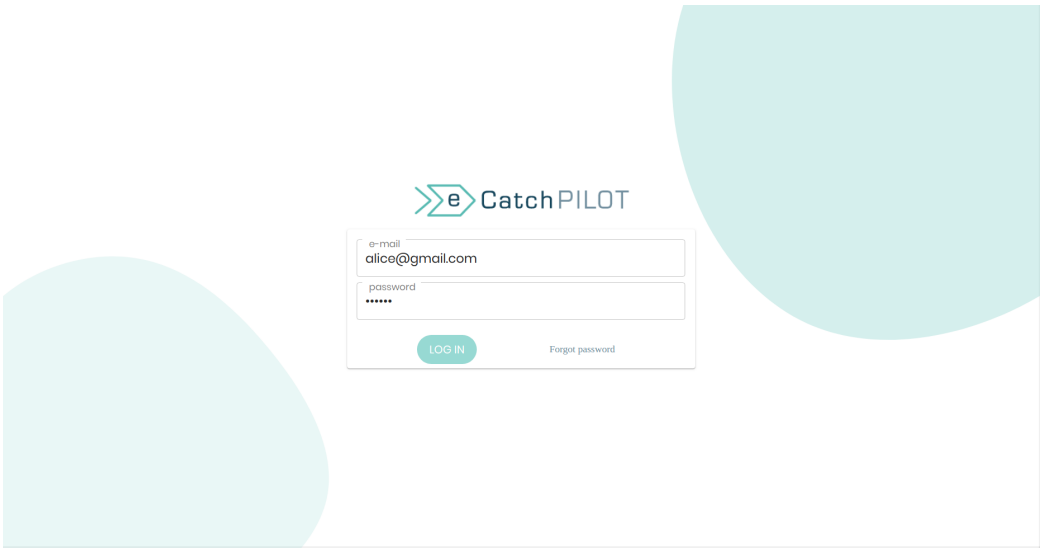


Figure 20: Login page with "Login" button

H.3 Home

You are now presented with the home page. This is the main page of the application and will always show the current actions you are able to do.

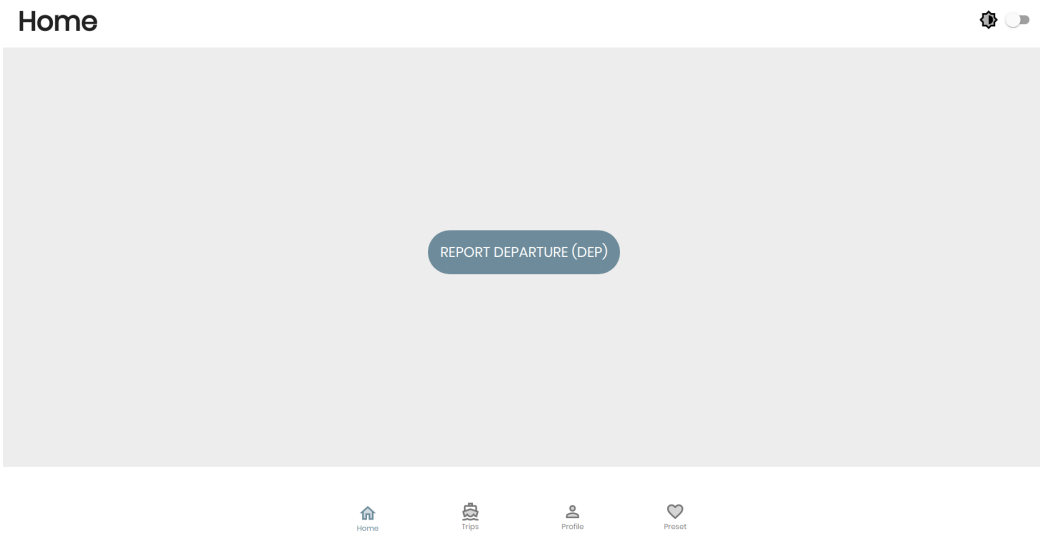


Figure 21: Home page

H.4 Setting presets

From the home page you can use the bottom navigation bar to navigate the preset page marked with a heart icon. Use the blocks to define your favorites. The values you set here will be used throughout the application.

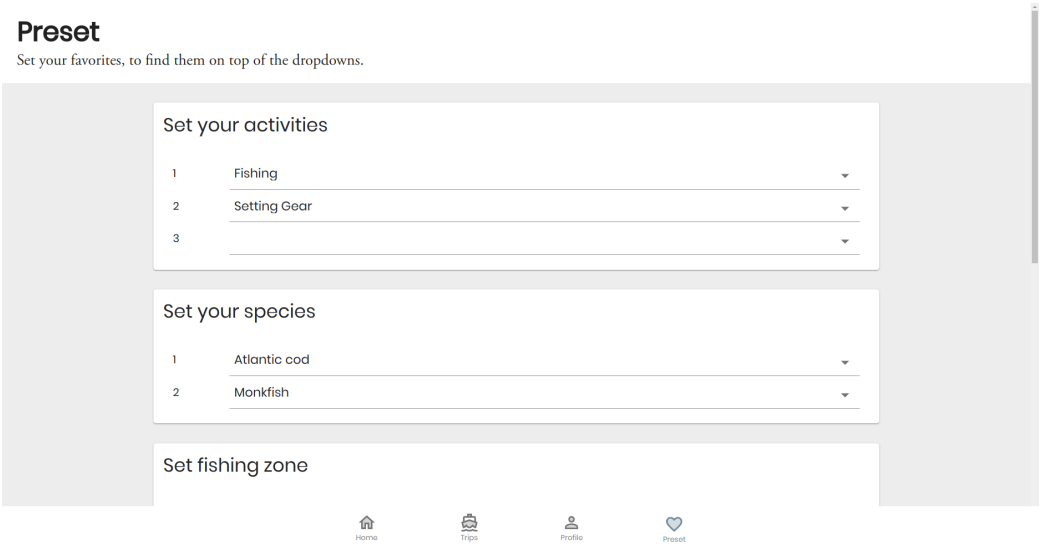


Figure 22: Preset page with multiple blocks

H.5 The first fishing trip

From the home page press the "Report departure(DEP)" button to start your trip. You should recognize that many fields have been filled automatically.

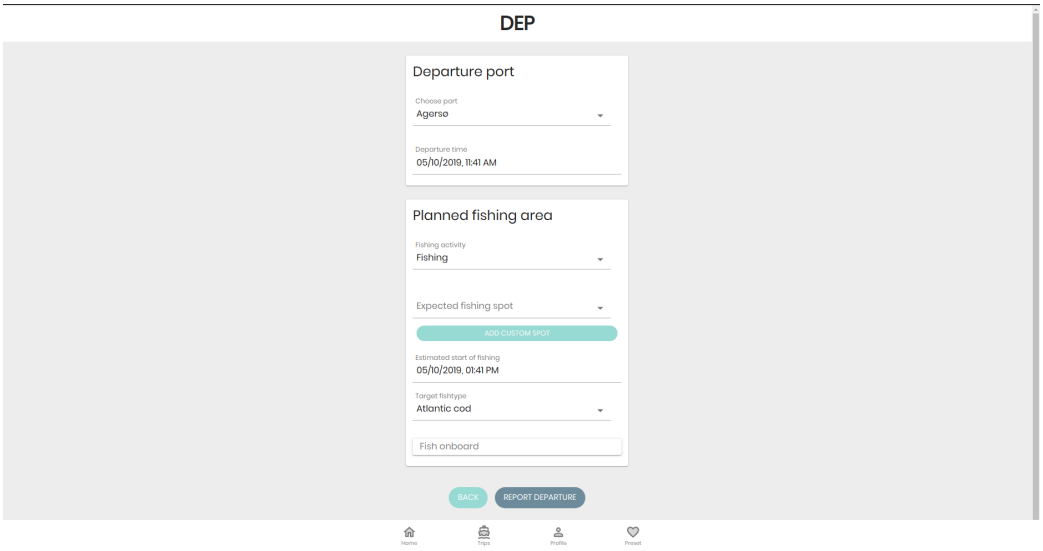


Figure 23: DEP message form

Go ahead and press the "add custom spot" to create a fishing spot. Give it a name and a position

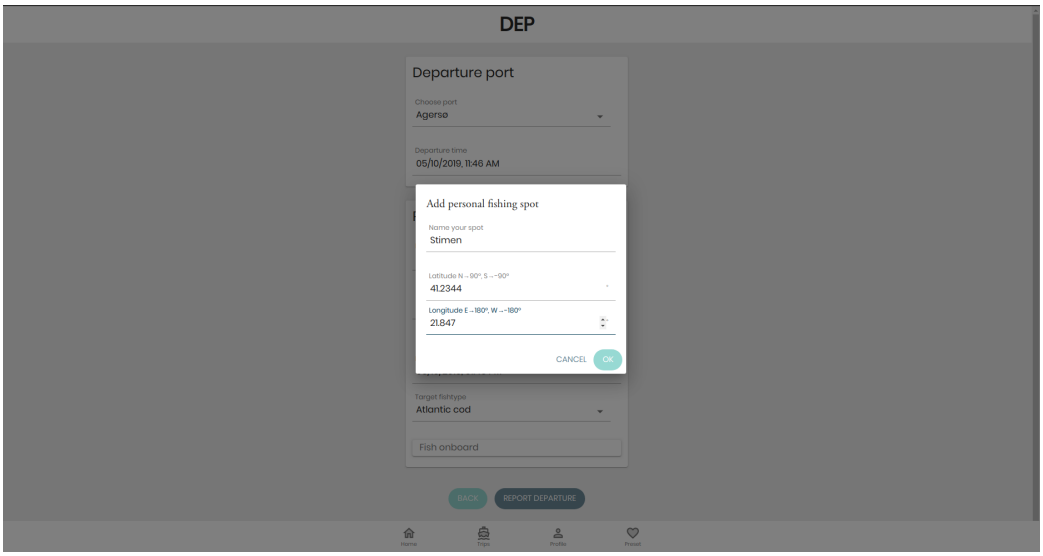


Figure 24: Add fishing spot form

If you are satisfied with the information you are now ready for departure. Press the "report departure" button.

You are now presented with the "Trip" page. Here you can see your active trip highlighted. Pressing on "Show active trip" takes you to the current trip.

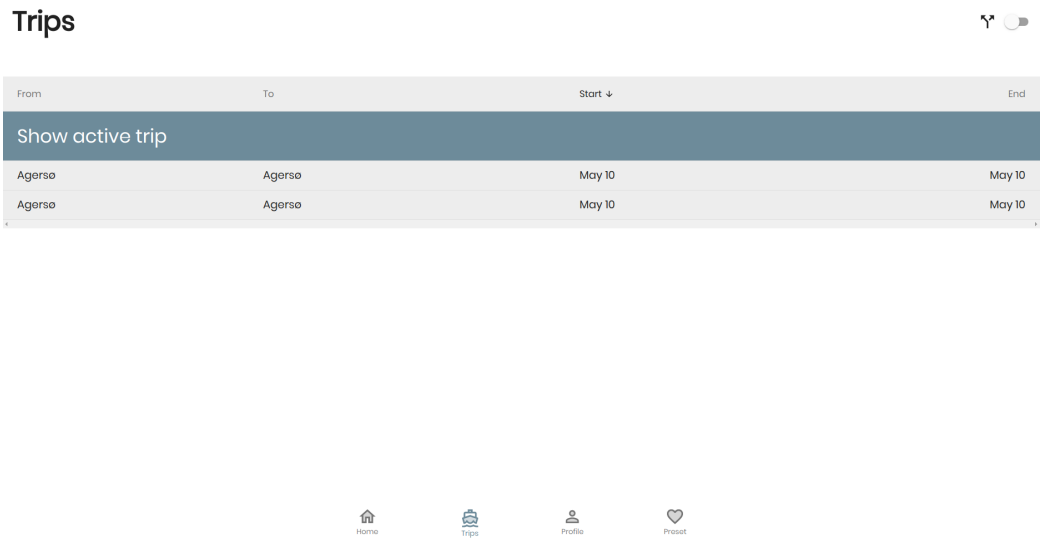


Figure 25: Trip overview of current and all previous trips

The next step is to report catch. To do this by pressing the "start DCA" button. The start DCA form is filled out automatically and should require little modification before pressing the "start" button.

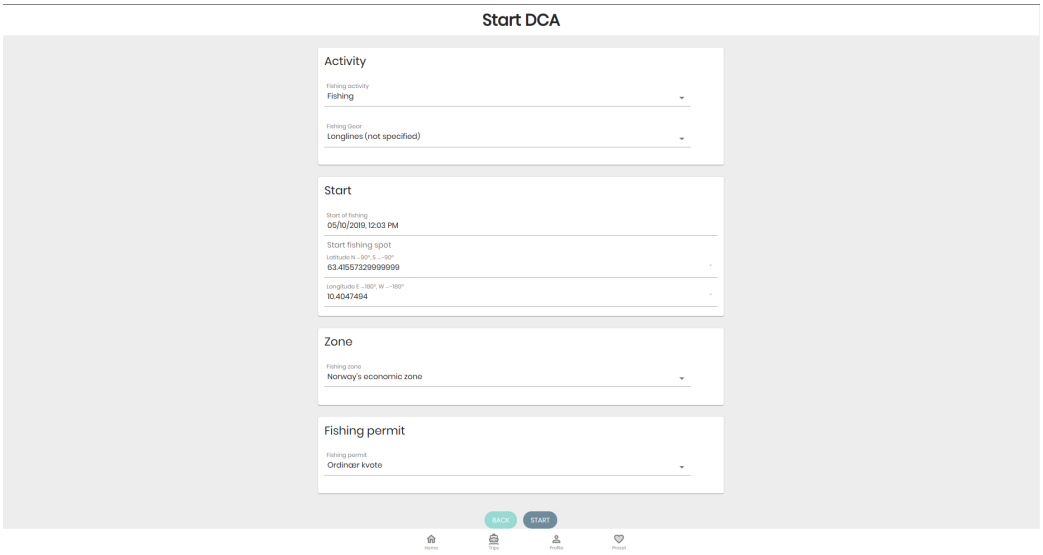


Figure 26: Start DCA form

You are now back to the home page with the option to press the "register DCA" button. Press said button to open the last part of the DCA message.

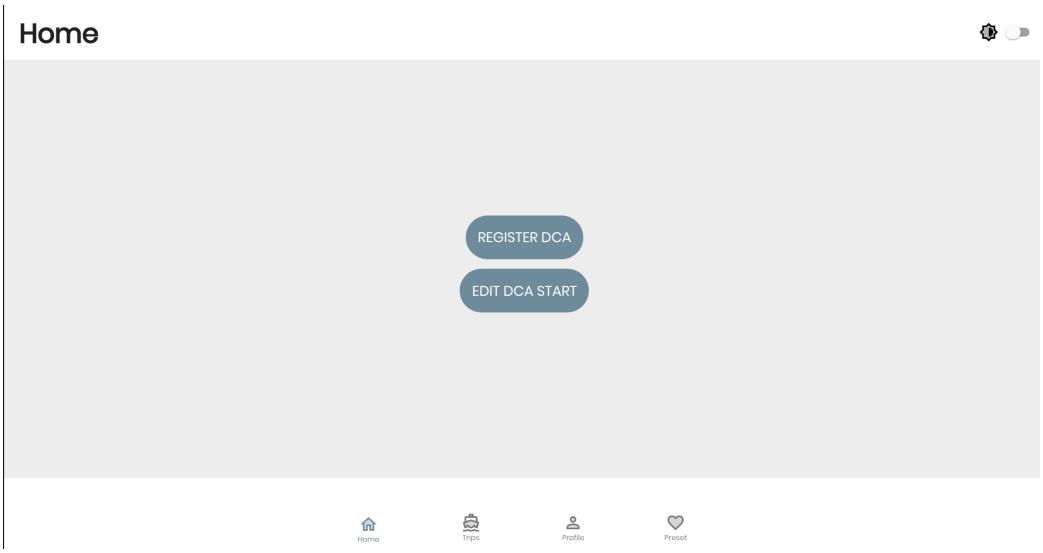


Figure 27: Home page is now filled with two new buttons related to DCA

You are now presented the last part of the DCA form. The fields should again be automatically filled in. In order to register catch, press the "change fish" button. You can select all the species you have caught and confirm by pressing the "OK" button. Following, you should see them appear in a list. By pressing in the respective fields you can change the quantity of every fish. To complete the form you press the "send DCA" button.

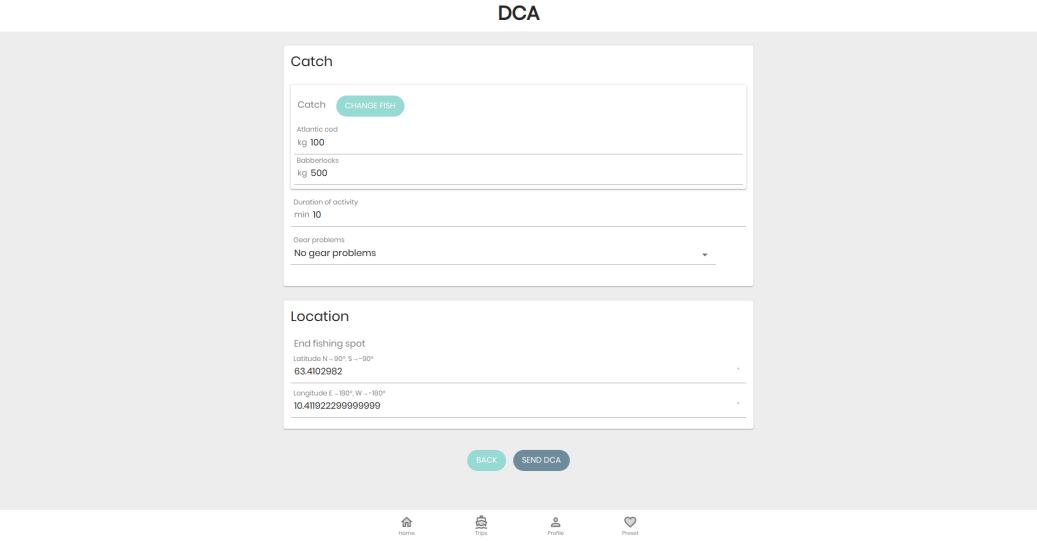


Figure 28: Home page is now filled with two new buttons related to DCA

After your DCA message you are sent back to the current trip overview. From here you can see catch currently on board, start a new DCA message or report port call POR. Let us press the "port call POR" button.

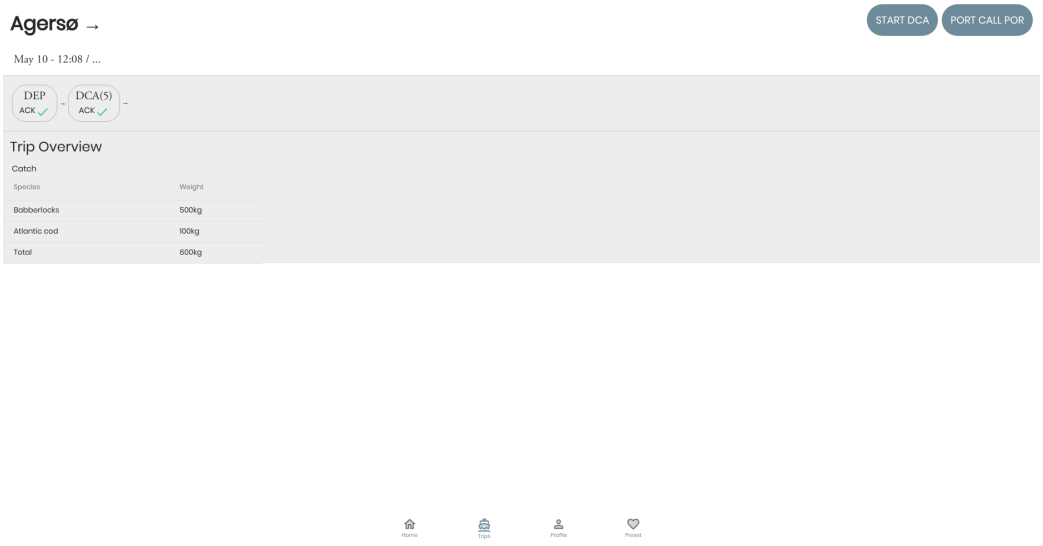


Figure 29: The overview is now updated

Again, you will notice most of the fields are automatically filled in. The "fish onboard" block shows all fish that currently is on board. After making sure every field is correct, you can press the "report port call" button. You are now directed back the trip overview and your trip is finished.

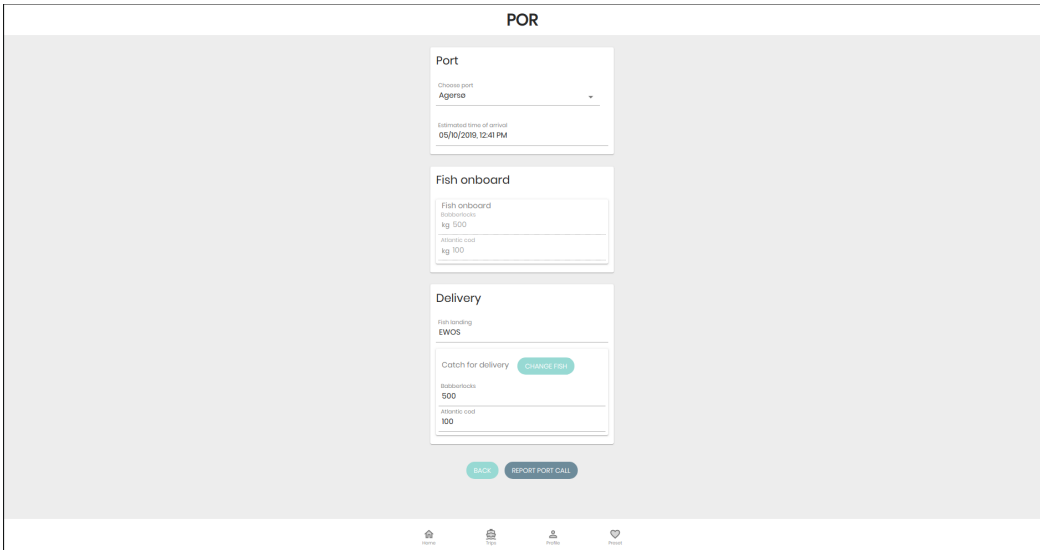
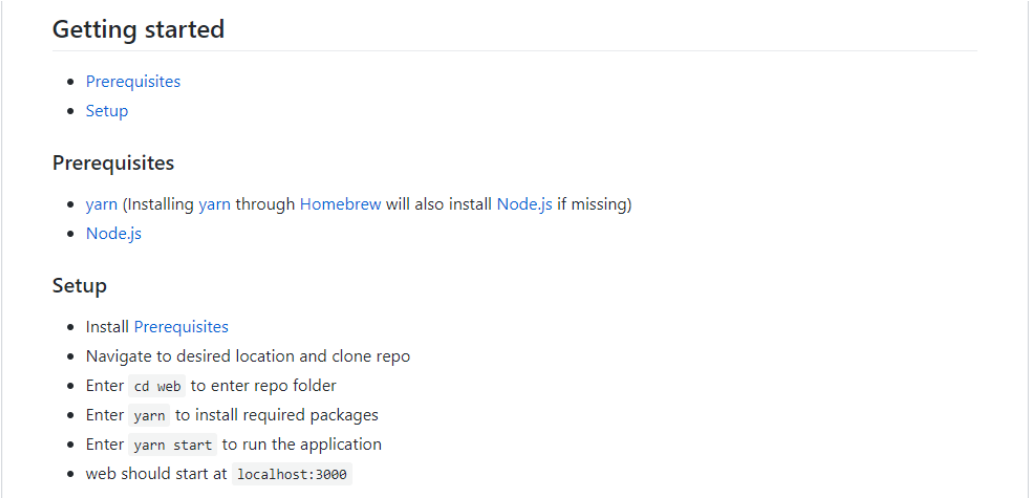


Figure 30: POR form

I Installation guide

As of the 10th of May 2019 the GitHub repository is avaiable at https://github.com/ecatch-kyst/web



Getting started

- Prerequisites
- Setup

Prerequisites

- yarn (Installing yarn through Homebrew will also install Node.js if missing)
- Node.js

Setup

- Install Prerequisites
- Navigate to desired location and clone repo
- Enter `cd web` to enter repo folder
- Enter `yarn` to install required packages
- Enter `yarn start` to run the application
- web should start at `localhost:3000`

Figure 31: Installation guide - excerpt from README.md