

Tommy Hamarsnes 8158
Ingrid-Alice Bløtekjær 8142
Krister Emanuelsen 8106
Marcin Janecki 8081
Ole Algoritme 8137

Subject Code: BA0301

Subject Name: Bachelor project

Client: Bouvet Norge AS

Delivery Date: 19.5.2021

Number of pages: 59

Number of words: 11 948

Availability: Open Limited

Kristiania University College

Mangrove - Å utvikle en app for pengeinnsamling og reduksjon av
karbonutslipp

Mangrove - Developing an application for fundraising and carbon footprint
reduction



Spring Semester - 2021

This bachelor is performed as part of the education program at Kristiania University College. The College is not responsible for the bachelor's methods, results, conclusions or recommendations.

Preface

As a final part of the bachelor's program we have completed BAO300 Bachelor Thesis as "consultants" in collaboration with a company in the industry called Bouvet. The purpose of this project is to gain work experience as well as show the knowledge and skills we have gained through the degree and the different subjects we've had.

We are using relevant research within our area of specialization, and we will also demonstrate use of technologies, tools and methods for development and project management.

Through this report we will increase the readers' understanding of how we have completed the project throughout the four and a half month timeframe, as well as how we have been collaborating as a group to gain the most out of our differences, improve our weaknesses and harness our strengths.

We want to express our gratitude to Bouvet for their confidence and trust in us, especially Andreas Rübner Johnsen and Johannes Dvorak Lagos, but also the Worldview International Foundation for proposing this project to Bouvet and therefore giving us the possibility to work with them.

Table of Contents

Preface	1
Table of Contents	1
Dictionary	5
1. Introduction	6
1.1 About the Project	7
1.2 About the client	8
1.3 About the team	9
1.4 Expectations from the client	10
1.5 Goals and ambitions	11
1.6 The problem	11
2. Literature Review	12
2.1 Progressive Web App	12
2.2 Usability	13
2.2.1 Designing the sales assistant	13
2.2.2 Designing the dashboard	14
2.2.3 Accessibility	15
2.2.4 User testing	15
2.3 Design Thinking	16
3. Process and method	17
3.1 Project Methodology - Our use of Scrum & Kanban	17
3.1.1 Managing the scrum team and maintaining the backlog	18
3.2 Project plan	19
3.2.1 Google Design Sprint	19
3.2.2 Iterative Sprints	20
3.3 Technologies and tools	21
3.3.1 Frameworks and Libraries	22
3.4 Data collection	25
3.4.1 Survey	25
3.4.2 User Testing and observation	25
3.4.3 General Data Protection Regulation (GDPR)	26
3.4.4 Norwegian Centre for Research Data (NSD)	26
3.5 Concept development	26
4. Design and user testing	28
4.1 Personas	28
4.2 UX flowchart	30
4.3 User testing	31
4.4 First prototype	32
4.4.1 Design	32
4.5 Second Prototype	33
4.5.1 Design	33
4.5.2 User Testing	36

4.6 Third Prototype	37
4.6.1 Design	37
4.6.2 User testing	40
4.7 Dashboard Prototype	41
5. Solution	43
5.1 Architecture	43
5.2 Layout and functionality	45
5.3 Database	49
5.4 Backend API Services	50
5.4.1 Authentication API	50
5.4.2 Payment API	54
5.5 Unit Tests	55
5.6 Further development	56
6. Discussion	58
6.1 Assessment of the solution	58
6.1.1 Design	58
6.1.2 Technical solution	58
6.1.3 Technical implementation	59
6.1.4 Benefit for the client	59
6.2 Assessment of process and method	59
6.2.1 Our use of Scrum and Kanban	59
6.2.2 Project plan	60
6.2.3 Risk plan	60
6.3 The research basis	61
6.4 Assessment of tools and technologies	61
6.5 Assessment of results in relation to the group's goals	62
6.6 Challenges	63
6.6.1 Everything is Online	63
6.6.1.1 Google Design Sprint	63
6.6.2 Learning about the unknowns	64
6.6.3 Developing with security as a focus	64
7. Conclusion	65
References and literature list	66
Attachments	70

Page intentionally left blank

Dictionary

1. **MVP** - Minimum Viable Product
2. **HTTP** - Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML
3. **API** - Application Programming Interface
4. **JSON** - JavaScript Object Notation is a lightweight data-interchange format
5. **GDPR** - General Data Protection Regulation
6. **NSD** - Norwegian Centre for Research Data
7. **UI** - User Interface
8. **UX** - User Experience
9. **UD** - Universal Design
10. **PWA** - Progressive Web App
11. **POS** - Point of Sale
12. **DT** - Design Thinking
13. **Native Application** - Application made in its original language for one specific platform, like iOS or Android
14. **CSS** - Cascading Style Sheet
15. **HTML** - HyperText Markup Language
16. **OTP** - One Time Password
17. **JWT** - JSON Web Token, a generated token that contains user information

1. Introduction

To finalize our bachelor program at the Institute of Technology at Kristiania University College, we are doing our BAO300 bachelor assignment based on developing a solution for a real company, solving real issues and documenting our process, thoughts, research, and results.

During the search for projects to work on we contacted Bouvet Norge AS about having our bachelor thesis with them. They presented a case about developing an application that improves the sales experience for the adults and kids in sports organizations that will be walking around door-to-door in order to sell trees to customers. The way they are supposed to do this is by having an app where the sports team members can do door-to-door sales by selling any amount of trees to anyone, rather than selling tickets or toilet paper.

The users of the app will for the most part consist of children and teenagers from 6 to 17, who play for their local sports teams as well as adults who tag along with them, particularly when it comes to the youngest sellers.

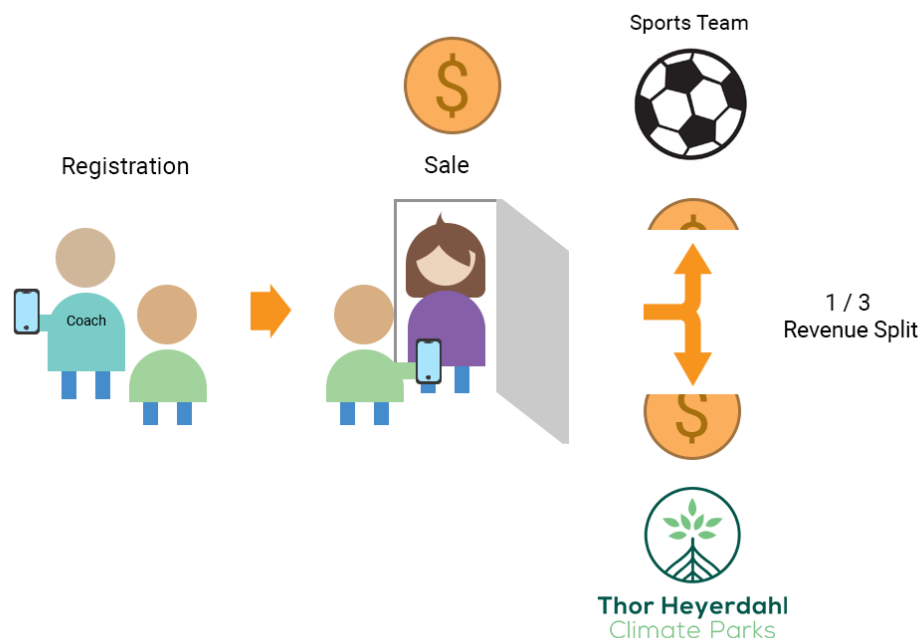


Figure 1: Simplified sales process

1.1 About the Project

Our client Bouvet Norge AS has received a request from a company called Worldview International Foundation (WIF). Part of the reason WIF wants to make this solution is to make it possible for sports teams, and later on bigger companies, to be able to write off their CO₂ emissions, by enabling them to efficiently “sell” the opportunity to combat climate change, support their favorite teams and support poorer areas of the world by adopting and planting mangrove trees. This way the solution becomes a win-win situation for all parties involved.

Our project specifically is about helping sports organizations support themselves, by enabling them to conduct their own fundraising sales, similar to how they already sell common products such as toilet papers and other small goods. However, instead of selling products the customers receive directly, we are enabling them to pay for mangrove trees being planted and grown in Myanmar. That way they can essentially purchase good conscience knowing they helped contribute both to climate change on top of supporting their favorite sports team.

The scope of our application will primarily be a sales assistant that helps the seller inform their customers about what they are selling, where the money goes and how it benefits both parties and of course the end goal: actually sell trees.

1.2 About the client

Bouvet is a consultancy firm delivering digital solutions from design and user experience to software engineering, as well as analysis, machine learning and more. Bouvet has 14 offices spread in Norway and Sweden, which enables them to understand the customer's language, culture and business. Bouvet has 1600 employees ready to help them achieve the vision and ambition of the company. See table 1 for an overview of external collaborators.

Name	Occupation Title	Association	Project Role
Andreas Rübner Johnsen	Department Leader	Bouvet Norge	Project Client Supervisor
Johannes Dvorak Lagos	Academic Administrator Mobile	Bouvet Norge	Project Client Supervisor
Tom Erik Heggedal	UX & Front-end Designer	Bouvet Norge	UX / Design Mentor
Andreas Falk Luksepp	CEO & Creative Director	WIF	Product Owner
Asbjørn Riddervold	Software Developer	Bouvet Norge	API Contact

Table 1 - Overview of individuals (excluding bachelor group members) involved in the project

1.3 About the team

This group consists of five students from the 3-year bachelor's programme "*Front end and Mobile Application Development*". Our technological insight combined with our individual areas of interests has awarded us a collectively broad knowledge span (see Table 2 below) and improved our work together as a team.

Name	Specialization	Primary Responsibilities
Krister Emanuelsen	Front end & Mobile Development	Front end authentication & security, front end tests, database management, scrum master
Ole Algoritme	Front end & Mobile Development	Back end authentication, project server management, Database management
Marcin Janecki	Front end & Mobile Development	Design, sketching, styling, front end development
Tommy Hamarsnes	Front end & Mobile Development	Vipps API, Database management, front end payment process
Ingrid-Alice Bløtekjær	Front end & Mobile Development	Project lead, user testing, front end tests, front end development
Sanchit Pawar	PhD Candidate	Internal supervisor

Table 2 - Overview of group members involved in the project

1.4 Expectations from the client

Bouvet expects us to develop a Minimum Viable Product (MVP) of what we anticipate the final solution to be. It is not expected for the group to complete a full application that handles economic transactions as this goes outside the group's area of expertise.

What is important to them is that our results are understandable, reasonable and also built in such a way that it is possible for the company to continue work on the project after the bachelor semester is over.

The expectations set for our team include a set of tasks we are meant to accomplish and develop during our time at the company to:

- **E1:** handle the design of the application with basis in published research through continuous user-testing, in order to establish a solid User Experience (UX) and adhere to the rules of Universal Design (UD) for the end product.
- **E2:** be in charge of development of the application by following established project methodology and providing sufficient documentation of our progress and end product.
- **E3:** properly establish automated unit-tests and maintain healthy version-control of our development process.
- **E4:** make educated decisions and choose our technologies based on established research and literature.
- **E5:** adhere to the rules and privacy concerns of software development relevant to the region we develop for, following General Data Protection Regulation (GDPR), and properly documenting how we store and handle data.

1.5 Goals and ambitions

We aimed for developing a solution that meets the technical requirements of the client, and ends up being a useful product for our associated companies tied to this project. The team had expectations to deliver an MVP of as high a quality as possible.

Our level of ambition was extended further by aiding in reduction of carbon footprint, decreasing direct human contact by having multiple ways of presenting information and payment options to end-customers.

A bonus goal is to get industry experience while cooperating with a well-established company on a real project.

1.6 The problem

The main issue we are solving is the lack of tooling that assists members of sports teams to raise money for their sports organization through voluntary work. Organizations doing voluntary work to raise money need to keep track of their resources, financial transactions and turnover, which existing tools are lacking.

Our task is to create a solution where sports organizations can sell mangrove trees directly to customers, keep records of sales and manage every member of the organization that is involved in the sales process.

2. Literature Review

In this chapter we will shortly introduce the main research our solution is based on. Research and information has been gathered from trusted sources such as Google Scholar, International Electrical and Electronics Engineers (IEEE), Norman Group, and Association of Computing Machinery's digital library (ACM).

2.1 Progressive Web App

During the research phase we were free to choose how we want to build our solution. In accordance with client expectation E4, we decided to develop the app as a Progressive Web App (PWA). PWA is a web application built using typical web technologies like HTML, CSS and JavaScript. PWA should be installable on the homescreen of a device, must behave similar to native apps, work offline and be accessible on different platforms (iOS, Android, PC) (Cardieri and Zaina, 2018).

PWA was chosen for a set of reasons:

- Accessible on any device with a browser.
- Lightweight and take up very little space on devices
- Faster and less complex to develop
- Easily distributable with a links
- Installable and can to a certain extent work offline

(Bjørn-Hansen, Majchrzak, Grønli 2017).

We analyzed how choosing a PWA approach impacts the user experience. Although the PWA may need more time to render the content (Bjørn-Hansen, Majchrzak, Grønli 2017), the research made by Cardieri and Zaina (2018) shows that PWAs are "*similar to native applications regarding interactions, navigations and appearance*" and that there is no huge difference in a way web mobile applications perform compared to the native ones (except for 3D games or image processing apps).

We took into consideration that our solution is not meant to be in widespread use for the general public, but instead only actively used by specific users during short periods

of time. In conclusion it was important to create a lightweight solution, accessible on devices most people use. Building native applications is more time consuming and requires more development experience (Cardieri and Zaina 2018). As we don't depend on native mobile features, we opted for a PWA approach instead.

2.2 Usability

Nielsen Norman Group (Nielsen, 2012) defines usability as a “*quality attribute*” that estimates how easy the user interface is to use. Nielsen says that usability can be measured by following quality components:

- **Learnability:** How easily users accomplish basic tasks the first time they encounter the design?
- **Efficiency:** How quickly tasks can be performed?
- **Memorability:** How easily can users reestablish the efficiency after they return to the design after a period of not using it?
- **Errors:** How many errors do users make and how easily can they recover from the errors?
- **Satisfaction:** How pleasant is it to use the design?

Since our project is a fundraising tool, it's crucial it offers features that are easy and pleasant to use which in the long run improves users' productivity.

2.2.1 Designing the sales assistant

After discussion with the customer we agreed that teenagers and young adults are the target age groups of this part of the app. Teenagers expect web-apps to let them easily accomplish their tasks. This age group has a low level of patience and when mistakes happen, teens tend to give up quickly (Joyce and Nielsen, 2019). Therefore following the advice from Norman Group we should have in mind that text for a young user should be short and understandable. Not every element of the interface has to be interactive, although teens appreciate clear aesthetics and interesting, rather neutral than childish, content. It's also important that the content isn't dense and the app is easy to navigate (Joyce and Nielsen, 2019).

The sales assistant is supposed to imitate the point of sale (POS) systems used by cashiers to check out customers. Dennis Lenard, one of the founders of award winning UX Design agency - Creative Navy, writes in his article *that “the role of the cashiers is to engage with the purchased goods and be friendly to the customer, while the POS software is an unpleasant distraction, which must be dealt with for accounting purposes.”* (Lenard, 2018).

Based on the same article we created a short list of design factors to take into consideration:

- short transaction time, system must be fast and as accurate as possible
- navigation flow must be simple and readable
- reduced interface complexity to avoid information overload
- proper error feedback

With this knowledge we are building a solid, user-friendly product that will meet users' expectations in the future.

2.2.2 Designing the dashboard

It seemed natural for us to design the administrative interface as a dashboard. Even though this part of the app isn't in the main scope, we wanted to prepare for further implementation but also see what experts have to say on how to design the dashboard.

Taras Bakusevych, a principal designer in Windmill Smart Solutions, says that a dashboard is *“the cockpit area from which a pilot controls the aircraft”* (Bakusevych, 2018). It should be a preview of the most important information for the user, with simple navigation to areas that require attention. Bakusevych mentions that choosing the right representation of data is crucial for the dashboard. He points out not to use diagrams which may be difficult to read when too many components are included and warns from using gauges and 3D charts that have lower readability. A dashboard should be the last thing designed as it is a summary view of all the data within a project (Bakusevych, 2018).

2.2.3 Accessibility

Universal Design (UD) is a composition of the environment that is accessible and used by all people regardless of their age or disabilities. The World Wide Web Consortium created a set of guides for mobile web content and apps as well (The Centre for Excellence in Universal Design, n.d.).

We wanted to follow those principles as we want our product to be accessible and beneficial for everyone.

Here is the list of the most important principles we want to implement in our solution:

- content accessible on different devices
- basic navigation at the top of the page
- ensure text and background color provide proper contrast
- provide informative error messages

2.2.4 User testing

Jakob Nielsen says that user testing is the most basic method to study and improve usability (Nielsen, 2012). Throughout the project we conducted several user tests (observation tests) to make sure we made an app with UX highly prioritized (Nielsen, 1994).

Tests are held with a group of people that fit the target demographic of the app. They should be given concrete tasks and their behaviour should be observed by the testers. It's important that test-subjects solve the tasks individually to properly observe issues the design may contain (Nielsen, 2012).

Any object, product, system, or service that will be used by humans has the potential for usability problems and should be subjected to some form of usability engineering. User testing with real users is the most fundamental usability method and is in some sense irreplaceable, since it provides direct information about how people use devices and what their exact problems are with the concrete interface being tested (Nielsen, 1994). These statements from Nielsen's book prove user testing is cost saving. As soon as we had progress on the solution, user testing had been on the agenda. User tests revealed the way the target demographic felt about how the solution works.

2.3 Design Thinking

"Design Thinking is not only seen as a motor for innovation promoted by designers, but it offers new models of processes and toolkits which help to improve, accelerate and visualise every creative process, carried out not only by designers, but in multidisciplinary teams"
(Clemente, Vieira & Tschimmel, 2016)

Design Thinking (DT) is a method to make decisions tied to what customers really want, instead of relying on historical data or making risky assumptions based on instinct instead of evidence. (IDEO U, n.d.)

The DT process merges *desirability* from people and users, *feasibility* from a technological point of view and *viability* economy wise. Desirability is what makes sense to and for people, feasibility is what is technically possible within the foreseeable future and viability is what is likely to become part of a sustainable business model.

The way design thinking is executed is in linear steps, but in practice the process is not always linear. Sometimes you start over in some parts of the process, and therefore it is an iterative way of working. (IDEO U, n.d.)

"Design thinking is not limited to a process. It's an endlessly expanding investigation."
—Sandy Speicher, IDEO CEO

DT was the method behind our choices because this process method helped us break down our thoughts and together come forward with a collective opinion and understanding about how our app should look.

3. Process and method

In this section we will be presenting our choices of technologies connected to the development of our project and how we have established it in the project. We will also show the initial layout concept.

3.1 Project Methodology - Our use of Scrum & Kanban

In order to maintain a consistent workflow and pace in our development process, we made use of Scrum as our agile project method for defining our backlog and project requirements, with kanban as a supporting method for keeping track of our specific tasks throughout the sprints. Quote from Kenneth Rubin. (2012): *“Scrum is particularly well suited for operating in a complex domain. In such situations our ability to probe (explore), sense (inspect), and respond (adapt) is critical.”*

Our scrum sprints were generally scheduled to cover two weeks (4 work days per week) per sprint. Figure 3 represents the timeline of the project. The first month we started off with a planning phase where we collectively agreed that two weeks would be for the best due to some group members having to work on certain days, and it would give us enough time to accomplish tasks within a sprint given a reasonable scope was set.



Figure 3: Project Timeline

We delegated the primary scrum roles to certain members of the team, such as scrum-master, project-lead, and general development team, and then delegated each team member a part of the product they were primarily in charge of.

Due to the ongoing global pandemic, we were unable to hold traditional daily-stand ups in an office or dedicated workspace, but worked around this by holding daily meetings using Discord as our online communication platform.

The software we agreed to use to support our scrum and kanban process was Jira, a project management software developed by Atlassian. Jira allowed us to create a roadmap for our project which we used to define the base parts of the semester. We dedicated the first month of the project to planning and research, February through April for most of the development work and sprints, the final three weeks were spent polishing our work and preparing the oral presentation.

3.1.1 Managing the scrum team and maintaining the backlog

One of the decisions we made fairly early as a team was that we did not need to document each group member's work hours.

Although requiring documented work hours is a slight deviation from what normally would happen for Scrum projects, we thought that due to the remote work situation for some members working part-time or raising children, requiring work hours would put unnecessary stress on some members of the team.

As opposed to tracking our progress through time-estimates, or through story-point estimation on tasks, we do this through a *cumulative flow progress*, represented in Figure 4, which is a visual indicator of the project's amount of to-do's, in-progresses and completed tasks.

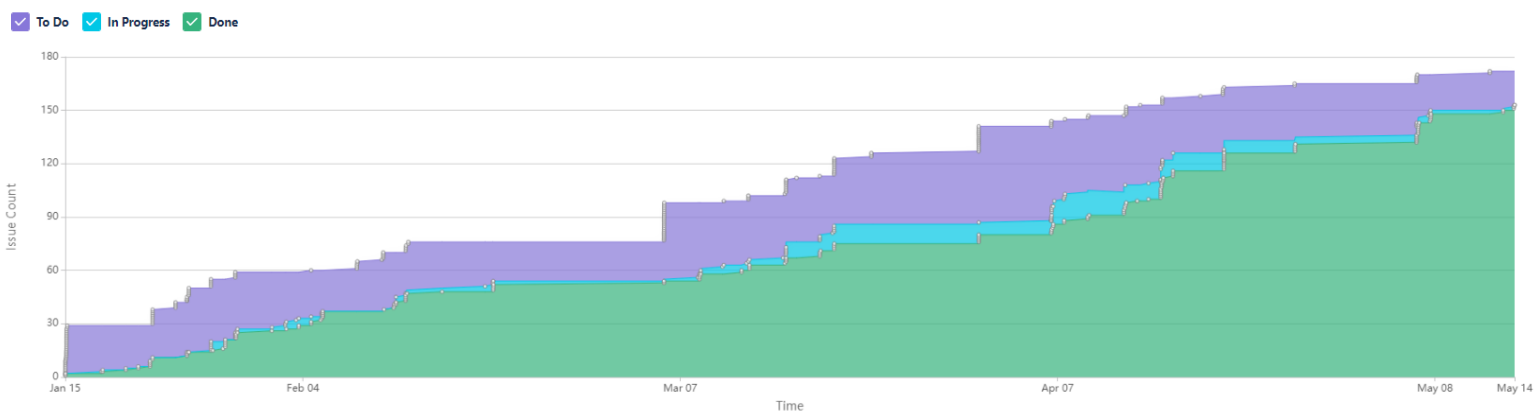


Figure 4: A cumulative flow diagram generated by Jira, tracking issues in the project

In order to ensure the backlog was maintained, we intended to periodically remind the team to give status updates on selected tasks, making sure it's updated by the end of each day to follow up on the latest updates for standups the morning after.

3.2 Project plan

The first three weeks consisted of research and planning. We have been using Jira and Confluence as tools for planning and achieving the goals we set. The project plan ended up being a roadmap in Jira (Figure 5) where all the small and big goals were visually represented.

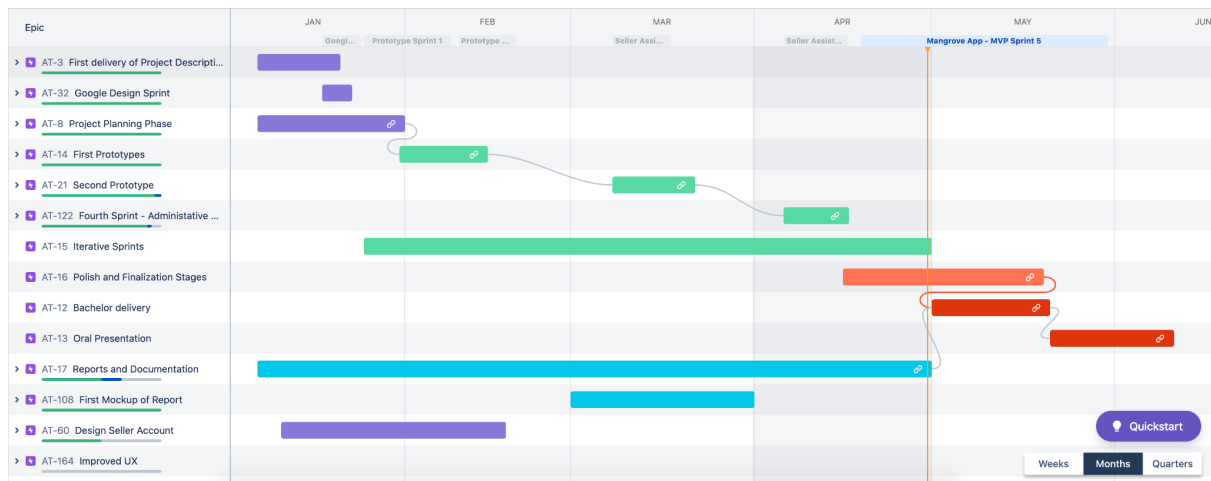


Figure 5: Roadmap as represented in Jira

The project plan has been an important part of the project due to the fact that we need some deadlines in order to achieve our goals. Having a project plan has helped us stay on track, show up every day and know what to do in order to progress. All group members also signed a group contract (attachment G) to establish a basis for making decisions and conflict avoidance.

3.2.1 Google Design Sprint

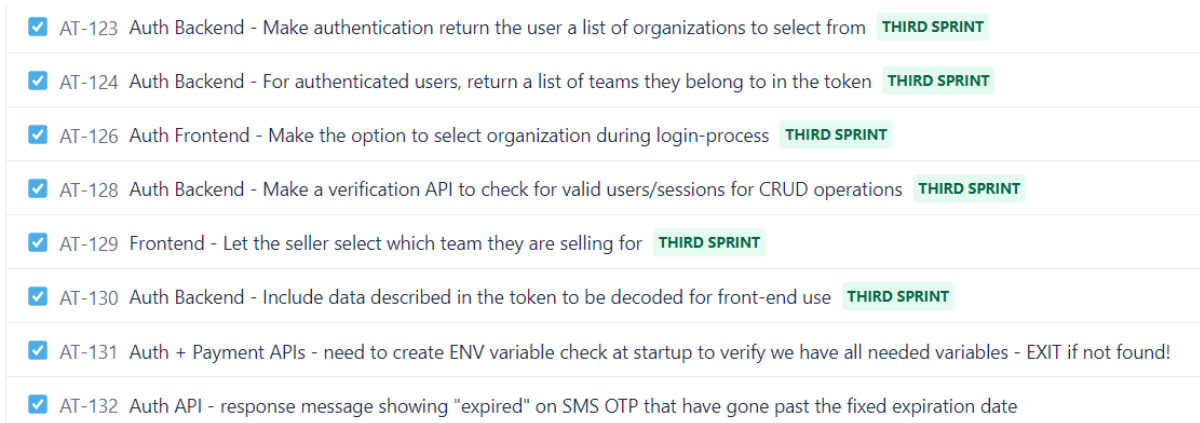
Three weeks into the project, we decided to attempt a Google Design Sprint (GDS) week. This entails going through a strictly planned out week in order to try and come up with possible solutions for the problem we were facing.

Our whiteboard, map drawings and sketches during GDS were done through Lucidchart - an online collaborative drawing tool combined with Figma and the "Sprint: how to solve big problems and test new ideas in just five days" book to follow GDS rules.

(Knapp, Zeratsky, Kowitz, 2016.)

3.2.2 Iterative Sprints

From early February all the way until May, we planned on conducting scrum sprints continually to push forward with the development of the application. As stated earlier we held two-week long sprints, with 4 work-days per week. In each of these sprints, we set a goal for what we wanted to accomplish and filled the backlog (Figure 6) with highly specified tasks we could assign individual team members to work on.



<input checked="" type="checkbox"/>	AT-123	Auth Backend - Make authentication return the user a list of organizations to select from	THIRD SPRINT
<input checked="" type="checkbox"/>	AT-124	Auth Backend - For authenticated users, return a list of teams they belong to in the token	THIRD SPRINT
<input checked="" type="checkbox"/>	AT-126	Auth Frontend - Make the option to select organization during login-process	THIRD SPRINT
<input checked="" type="checkbox"/>	AT-128	Auth Backend - Make a verification API to check for valid users/sessions for CRUD operations	THIRD SPRINT
<input checked="" type="checkbox"/>	AT-129	Frontend - Let the seller select which team they are selling for	THIRD SPRINT
<input checked="" type="checkbox"/>	AT-130	Auth Backend - Include data described in the token to be decoded for front-end use	THIRD SPRINT
<input checked="" type="checkbox"/>	AT-131	Auth + Payment APIs - need to create ENV variable check at startup to verify we have all needed variables - EXIT if not found!	
<input checked="" type="checkbox"/>	AT-132	Auth API - response message showing "expired" on SMS OTP that have gone past the fixed expiration date	

Figure 6: A sample set of tasks from our backlog on Jira - March 2021

For each sprint we set a goal and added additional tasks in case we achieved all the goals for that sprint.

This was done to ensure we would always have progression in the project, and prevent having members wait until the next sprint to continue their work.

Depending on the sprint goal, we aimed to conduct user-tests at the end to ensure consistent external feedback and maintain the quality of our work.

We did not plan to run these sprints every single week. Particularly in the early parts of the development process, we anticipated several weeks dedicated to gathering information and progressing on our reports and documents. We planned to integrate these dedicated weeks in-between our sprints spontaneously depending on our current state of the project, as well as our deadlines and schedules.

3.3 Technologies and tools

This section goes into details about which tools and technologies were chosen for our solution. Most of the decisions regarding which technologies to use are based on the fact that our app is developed in a web environment.

Vipps

Norway has the highest number of cashless transactions per capita which leads towards a cashless society (Deloitte, 2019). In 2015 DNB launched Vipps - a peer-to-peer payment service. According to the statistics 3.9 million people used Vipps in January 2021 (Vipps, 2021), which is almost 80% of the country's population.

Door-to-door sale requires use of mobile devices like phones or tablets as it's the most convenient for the sellers. Therefore Vipps was chosen as the main payment method in our app.

Typescript

As our PWA approach is a form of web development, we developed following the modern standard for web-development using HTML, CSS and JavaScript. TypeScript is an open-source language that builds on JavaScript and gives developers a way to develop JavaScript code with strictly typed definitions, helping developers keep track of variable types (TypeScriptlang, n.d.). We chose to use TypeScript because it allows us to avoid more errors during development. This scripting language provides a way to describe an object, which provides better documentation for anyone who is reading the code, as well as the TypeScript compiler giving more precise feedback on errors and warnings (TypeScriptlang, n.d.).

Hasura & GraphQL

Hasura is a modern and well-maintained service we use that controls our Postgres database. It gave us the freedom to easily create database tables with entity relationships (ER) and perform queries with the Graph Query Language. Since most of us in the group had previously worked with Hasura and GraphQL, this was an obvious choice we made as we all enjoyed the ease of use and intuitive combination Hasura & GraphQL Engine offers. On top of that, it has built-in websockets which makes it possible

to subscribe for database events and give users updated information in real-time (GraphQL, n.d.).

3.3.1 Frameworks and Libraries

React

React is a component based, declarative JavaScript library for developing user interfaces as cited from React's official page (ReactJS, n.d.). It lets us create reusable code, saving development time and simplifying manipulation of the HTML page visible to the end user. React is among the most popular JavaScript libraries, and is supported by a very large number of third-party libraries.

We looked at several articles and sources comparing React against the other two largest frameworks Vue.js and Angular and found no strong arguments for choosing Vue or Angular over React.

According to athemes.com React's popularity makes finding components and compatible libraries comparatively easier than for Vue and Angular (Pattakos, 2021; Sev, 2021, BuiltWith, n.d.).

The scale of our project is not expected to be large enough to warrant significant performance concerns when it comes to choosing a front end UI framework, but also not small enough to justify simply making the app in pure HTML, CSS and JS. We chose React mostly based on the team's existing experience with React development, and to avoid spending too much time learning unfamiliar frameworks.

React Router

React Router is a routing library specifically designed for React, which allows our app to become what is called a Single Page Application (SPA).

As an SPA, the app may take slightly longer to load when visited, but saves time when the user tries to navigate between different parts of the app.

*“A **single-page application (SPA)** is a web **application** or a website that interacts with the web browser by dynamically rewriting the current web **page** with new data from the web server, instead of the default method of the browser loading entire new **pages**”.*

- (Tomlin, 2020)

Axios

For external API calls we use Axios. It helps us write smaller chunks of code compared to JavaScript's built in `fetch()` function. Axios sends data as an object and it transforms JSON data automatically. Axios has wide browser support and provides cross-site request forgery protection.

Styled-Components

Instead of writing CSS traditionally in separate files we use Styled-Components, which is a JavaScript library that allows us to assign CSS styles directly to components we create using React. This allows us to define our styles in the same files as the components they belong to, making it easier to keep track of style definitions and re-use styles for each component.

NodeJS with Express

NodeJS is a server based application for web based applications and back end services. It allows us to code our back ends using JavaScript/TypeScript as is used in our front end, making maintaining both front and back end easier. Our backend APIs are built using NodeJS with the Express library as a way to expose our internal APIs with a web server.

Other tools

Table 3 lists the tools and programs used in the project.

Discord	Discord is a platform used for communication through texting, video calls on private servers for information sharing. Used by the group for communication.
Jira	Project management system created by Atlassian group that makes it possible to use project methodics like Scrum and documentation writing.
Github	Web platform to store code for one or multiple projects with security, version and health control in mind for the projects.
Google Drive	Google Drive is a cloud service developed by Google for users and businesses to keep hold of their documentations and assets on a cloud service.
Lucidchart	Lucidchart is an online tool for creating and sharing flowcharts and diagrams.
Figma	Figma is a tool for creating SVG images and creating clickable prototypes in different platform formats
Teams	Communication platform used for meetings between the group, Bouvet and WIF
Slack	Online communication platform to send text and post updates whenever needed to Bouvet

Table 3: List of used programs and tools

3.4 Data collection

In this section the data collection procedures used in the project will be discussed.

3.4.1 Survey

Our application is meant to be used by a wide age group, which means it's important for us to know how we can design the app in such a way that it isn't overly tailored towards a specific group of people. As a part of PJ6100 Research Methods, we decided to study this topic in particular. In this course we developed a survey in which we showed the respondents examples of designs for both popular social media platforms, as well as some "app" designs, including examples from our own app. We then asked the respondents about their thoughts and expectations.

We were mostly interested in the responses related to what users expected to happen when they pressed on a button with various icons or texts, to observe how their expectations differed depending on their age group.

We looked for patterns in responses between the age groups and tried to see if certain designs stood out.

From the survey we wanted to use the most intuitive designs across all different elements to be used in our app together with UD.

We will discuss in later sections our results from the survey (attachment F) and show how it affected design choices in our solution.

3.4.2 User Testing and observation

In order to collect more qualitative feedback on our solution, we conducted user-tests. A user-test is conducted by a group of facilitators and one test subject. One of the testers is responsible for conducting the test and the others observe the subject's actions and reactions from outside.

The subject is given access to the solution and tries to accomplish tasks given by the tester. Based on the time the test takes, the subject's testimony and actions, the observers take notes and discover parts of the solution that may need improvement (Farrell, 2016).

3.4.3 General Data Protection Regulation (GDPR)

Since our solution is developed and intended for use in Norway, we are legally required to adhere to the rules and regulations given by the EU. (Datatilsynet, n.d.)

Our app is expected to collect identifying information such as phone numbers, names and email addresses of users and customers. According to European Commission, GDPR requires us to allow involved users to know what data is stored about them, allow users to request their data be deleted, as well as inform them about what data is stored locally on their end (European Commission, n.d.). See attachment B and it's section about GDPR and how it relates to each user and product owner. However, this data will not be collected until the fully developed solution is deployed by the product owners.

3.4.4 Norwegian Centre for Research Data (NSD)

Research institutes and universities that perform research or studies that collect identifying user information in Norway must be reported to the Norwegian Centre for Research Data (NSD). We intended to send an application to the NSD, but this was deemed unnecessary by our case giver Bouvet. By the end of the project period the app was not publicly available and we did not collect any information outside of our own development team.

Any data that may have been stored during user-tests were completely anonymized, as our contracts with our participants stated we do not collect any identifying information in our tests (attachment C).

3.5 Concept development

We aim the application to be easy to access, quick to use and simple to manage. Since the app handles monetary transactions we have to keep a high level of security to prevent misuse.

In order to keep the fluid flow of the app, we wanted to have as few steps as possible when authenticating users, and wanted our users to be taken directly to their respective interfaces. We decided to develop two distinct paths depending on the user role.

A sales assistant interface and the other one for administration. Our initial plan for the app's flow can be found in attachment H.

The sales assistant part of the app, designed for the most common user-group - *sellers*, would contain the essential features to conduct sales and inform their customer about the mangrove tree product.

The administrative part of the app would contain the essential features for creating the aforementioned users and their groups. Separated into two levels of administration:

- The moderators: responsible for controlling the users within teams they are delegated to moderate.
- The org_ admin: responsible for controlling the groups and teams, and assigning moderators to teams

We created a comprehensive description of user roles in attachment B.

4. Design and user testing

In this section we will show how the design changed throughout the development process and how the usability testing influenced those changes.

To structure the design rules and consistency in our app we decided to use some of the Google Material Design principles (attachment E). Material Design is a set of guidelines, components and tools developed by Google in order to create high-quality experiences on all platforms (Material Design, n.d.).

4.1 Personas

Personas is a design technique used to map out the possible functionality different users want in your solution (Chang, Lim, & Stolterman, 2008).

A persona is a fictional, typical user of the product. Based on the user research of the Mangrove app we created a list of made-up people (Figure 7).

The main reason to use personas is to provide possible desires and behaviours different users may want in the solution (Harley, 2015).

We added possible scenarios for every user to help us understand better who we are making this app for. This had a significant impact on design decisions we were making. Therefore it was important to create personas before we started the actual design process.



Figure 7: Personas created to represent various roles in the system
 Images are fetched from Pixabay.com and applies to their License agreement
 (<https://pixabay.com/service/license/>)

4.2 UX flowchart

After discussing the initial plan of the app and GDS process, we sketched the app UX flowchart (Figure 8) to determine what components and views we will need to set up.



Figure 8: UX Flowchart

After users log in, they are directed to the homepage that presents a sales overview. Users will see how many trees they have sold, how much money they have raised and how much CO₂ has been reduced. From the homepage users can navigate to the info page that holds all the information a user may need to answer questions from potential customers. A floating action button (FAB) on the bottom of the screen takes the user to the sales page on both pages.

On the sales page users can see different sections where the seller needs to fill out the amount of mangroves being sold, customer information like name, email and phone

number and which team the money goes to if the seller is on multiple teams. After filling in the required information the seller can start the transaction.

The organization leaders and coaches also have access to the dashboard with an overview of how the sales are going inside the whole organization, or a team they are moderators in. First sketches were represented in another flowchart (Figure 9). From there they can navigate to all necessary lists to manage the members of the organization/team (add, edit, delete). On this stage we didn't plan how users can navigate from the administrative interface to the sales assistant interface.

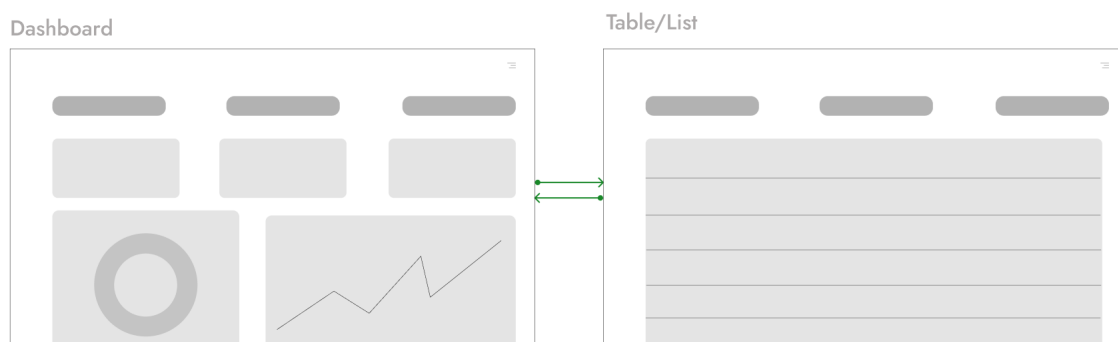


Figure 9: Dashboard UX Flowchart

4.3 User testing

In order to find any UX and design flaws we conducted user-testing on both our figma and functional prototypes. Considering the covid restrictions for social distancing, it was not ideal to have a physical meetup for user-testing. It had an impact on the user experience during the testing since it was conducted through Zoom.

We wanted to test a minimum of five subjects during each test round. After the fifth subject people usually observe the same things again so it may be ineffective to test more (Nielsen Norman Group, 2000).

The ideal user profile for our testing is any parent that has one or multiple kids in different sport teams, a child in a sport team, any young adult, a coach or an organization leader.

Every test included the same assignments for the test-subjects.

Before the start all participants were asked about their previous experience with door-to-door sales or fundraising in general. The process involves logging into the app and selling us, the testers, a set amount of trees. After the sale is done we ask test-subjects about the applications' pros and cons .

By using Zoom we could let the test subjects take over our screen while the prototype had a mobile screen output for simulating how the UI would be on a real device.

4.4 First prototype

4.4.1 Design

During GDS we sketched different alternatives for our solution. Then we used dot voting to choose the ones we found the best (Figure 10). That gave us more or less the idea and understanding of the project we wanted to develop.

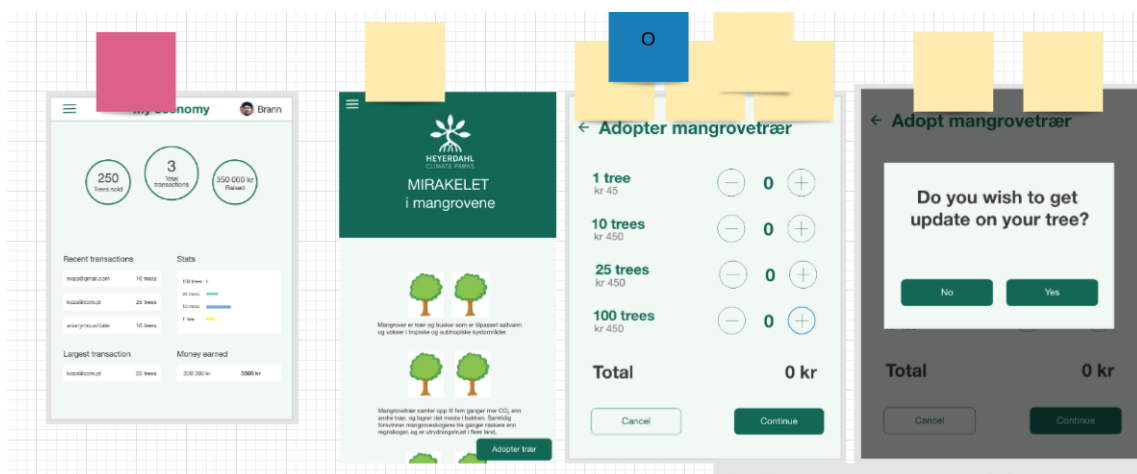


Figure 10: Example of dot voting in Lucidchart

After creating the app flow in Lucidchart we built the UX flowchart to see how exactly our app may be presented to the end user (Figure 10). We were using Figma for fast, simple prototyping and for initial user testing.

First prototype was made right after we were done with dot voting. We didn't specify any design rules yet as it was supposed to be a simple sketch of how the app should look like.

4.4.2 User testing

During the first round of user testing we only managed to get one subject. The product that was tested is seen on Figure 11. The subject gave us feedback on the design that it was nice and easy. The harder part was the navigation. The user had issues finding out that our Mangrove image at the bottom navigation bar was the button to navigate to the “Amount of trees sold” page. The Mangrove icon without labels under wasn’t intuitive enough.

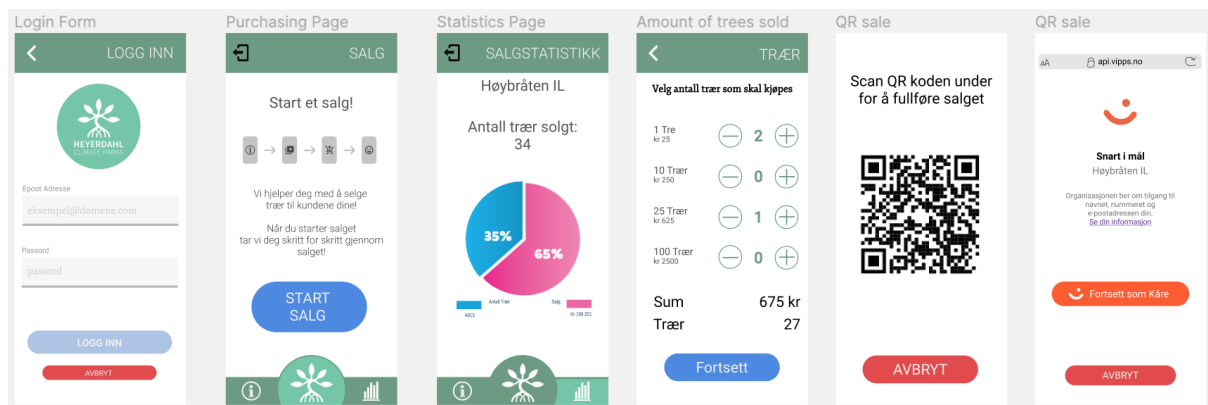


Figure 11: First app prototype

4.5 Second Prototype

4.5.1 Design

After the GDS and the first user testing were done we decided to specify the palette we wanted our app to be based on (Figure 12). During the research period we had to take a look and consider the design of the WIF’s page. The page consists of grey, white and a few different shades of green.

In an article from Vandelay Design (Sandu, 2016) green is stated as a color that stands somewhere between dark and light, but tends to find its place in the light palette. This also means green can be calm and relaxing, but being energizing at the same time.

Another article about color on Verywell Mind (Cherry, 2020) states that green is often associated with nature, and brings the mind to green grass, trees and forest. The article also says the color is restful, soothing and health giving.

In light of these reflections we decided that green would be the primary color in our app.

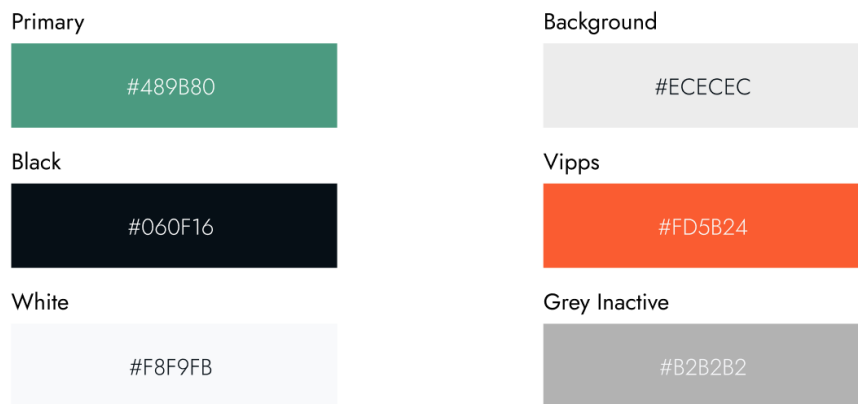


Figure 12: Color palette

To specify the layout and proper design of our app we created our own UI Kit (attachment D) that was used throughout the styling process. The reasoning behind our choices was mostly based on Material Design rules (attachment E). The statistics on the homepage are shown inside the cards (Figure 13).

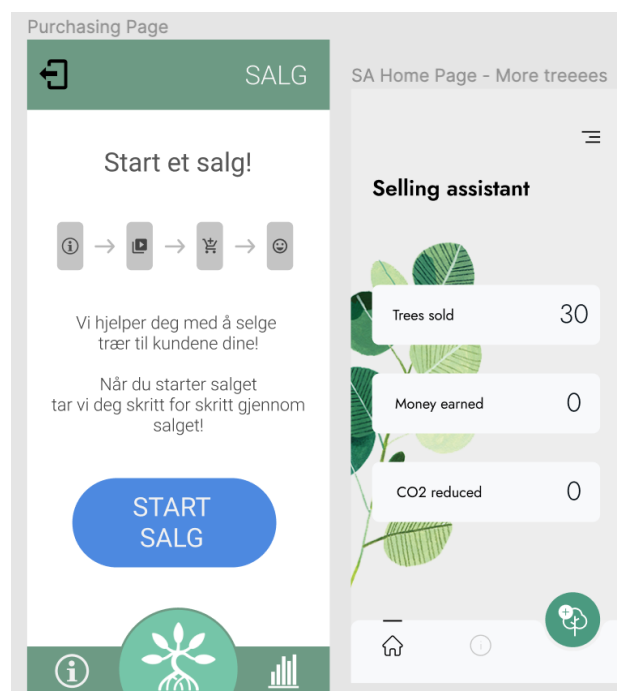


Figure 13: Left: Old Home version. Right: New Home page.

The whole selling process was conducted inside the modal bottom-sheet that “causes all content and UI elements behind it to display a scrim, which indicates that they will not respond to user interaction”(Material Design- Sheets:bottom)(Figure 14). The sales process was divided into steps so the seller could easily focus on one thing at the time. Afterwards we added a step-by-step indicator (Figure 14) that was supposed to be clickable and could take the seller back to one of the steps.

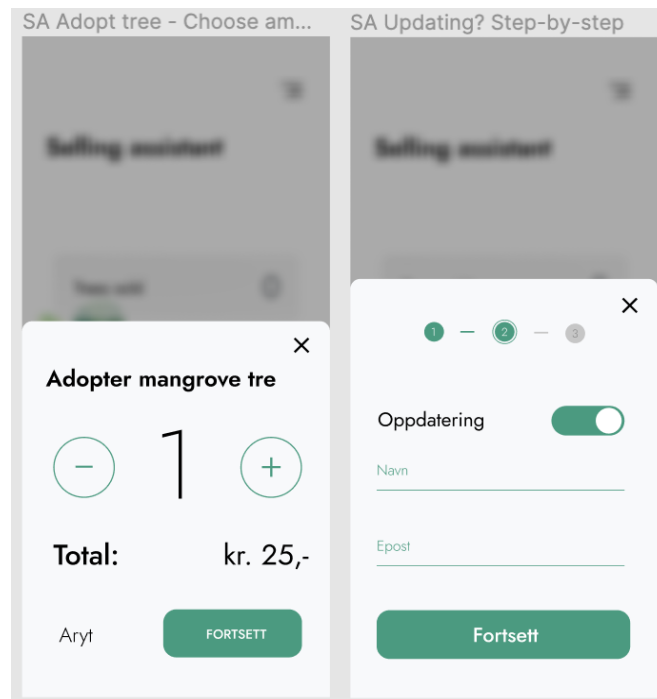


Figure 14: Sales process modal and step-by-step indicator added later

A FAB performs the most important action on a screen with the primary color is supposed to be distinguished from other elements on the screen so it immediately attracts the eyes of the user (attachment E). FAB was initially with just a tree icon inside, no label. The icon of a tree with a plus sign seemed intuitive enough to us. The test-subjects initially understood what the meaning of the button was. When asked in the survey (attachment F) 10 out of 54 participants guessed the purpose of it. However 44 participants didn't have a clue what the button does, or they thought the button would open the chat window.

We added only two destinations in the bottom navigation. Since the Material Design rules say “Don’t use a bottom navigation bar for fewer than three destinations (use tabs instead)” (Material Design - Bottom Navigation), we decided to insert the FAB button in the bottom navigation (Figure 15). Tab bar was initially with only icons without any labels below them.

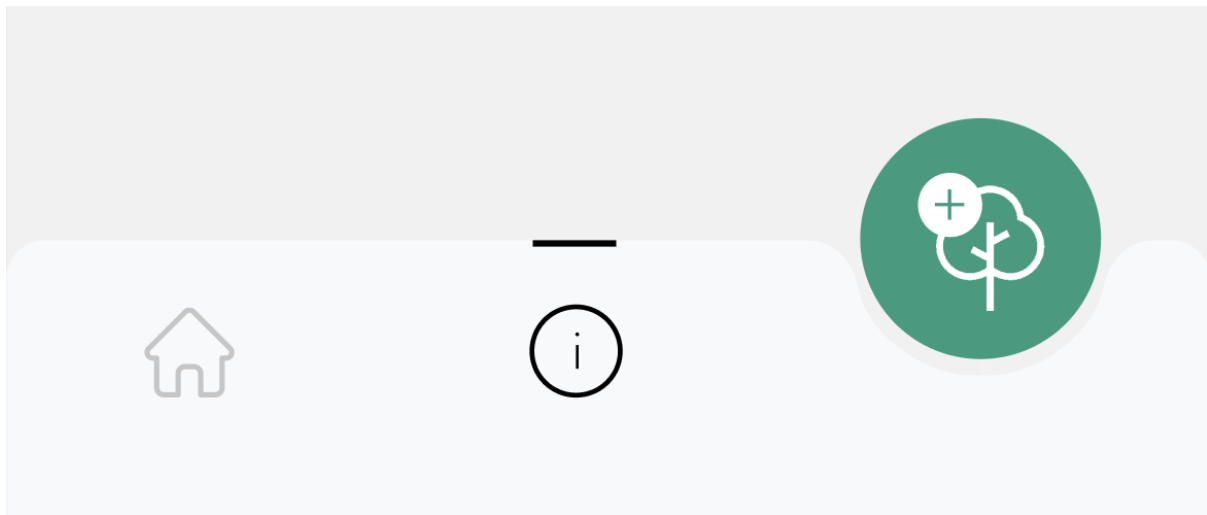


Figure 15: Tab bar with FAB button

4.5.2 User Testing

After choosing the color palette and setting up some design rules in our UI Kit we created another version of the app (Figure 16). During this usability test we got a hold of three test subjects. None had any issues on the simulated OTP login segment and easily navigated to the sales assistant. There they began encountering issues. The test-subjects stated that they had “less overview of the upcoming part”, which made them hesitate to go forwards faster and have less confidence in navigating. This meant we had to come up with a new and better overview version of the sales process.

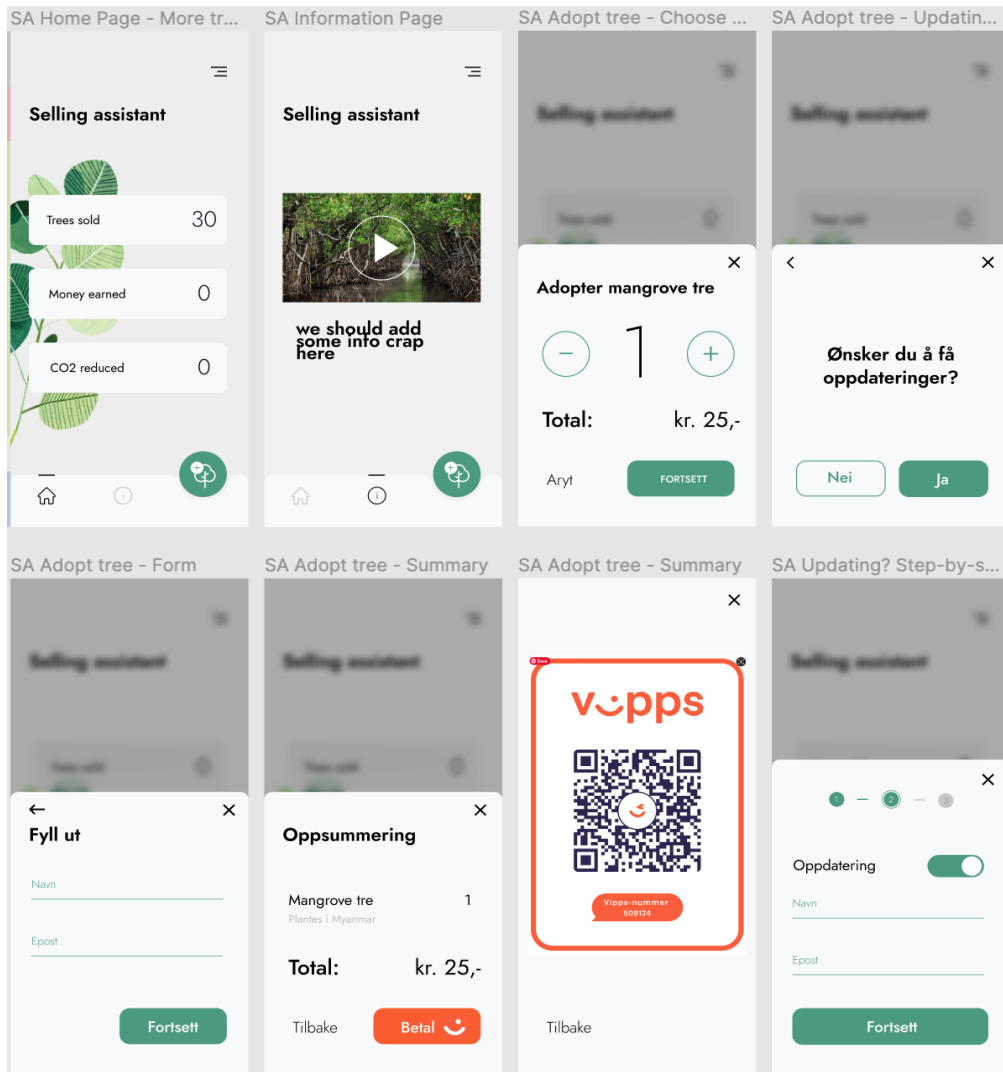


Figure 16: Second prototype

Note: QR code displayed here is only a sample link that routes to google.com

4.6 Third Prototype

4.6.1 Design

After the meeting with Bouvet's UX designer Tom-Erik Heggedal and hearing tips he had for us, we decided to introduce some changes. We removed the bottom navigation, as it wasn't needed. Instead we decided to lean towards UD principles and use tabs at the top of the screen. We removed the icons and used rounded labels instead. That inspiration came from Pinterest (Figure 17).

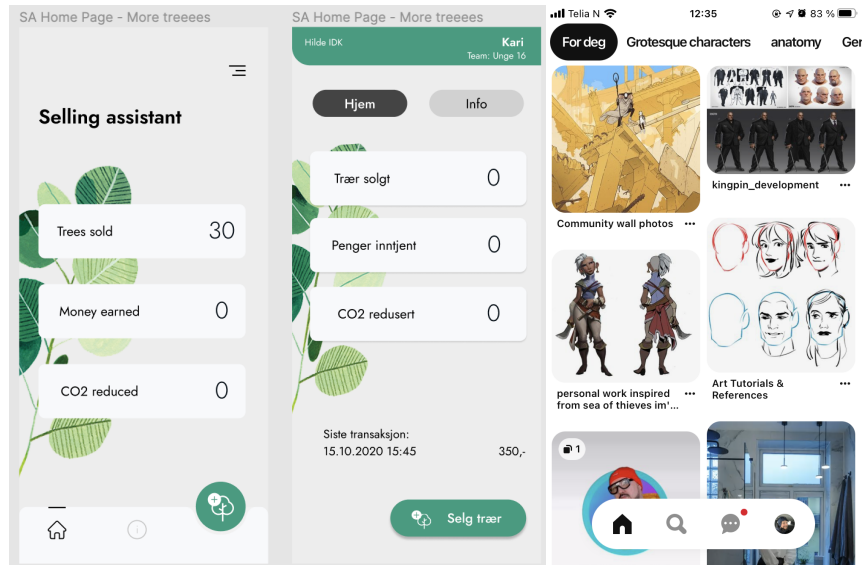


Figure 17: Left: Old tab bar vs new tab bar. Right: Pinterest tab bar

After studying results from the survey (attachment F), we decided to spread the FAB and add a text to it so it doesn't confuse the users who are not familiar with the "add tree" icon. The label however disappears when the user is scrolling the page down. Gmail app was our inspiration in that case (Figure 18). The content of the page is more visible and the wide button doesn't interrupt while reading.

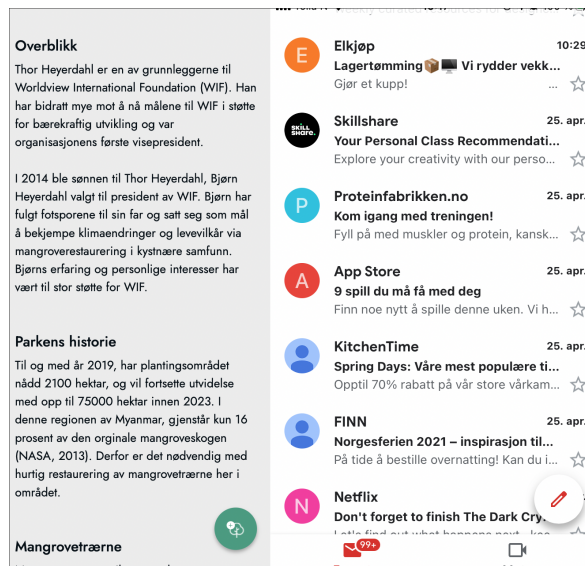


Figure 18: Left: FAB shortening on scroll in our app Right: FAB in Gmail

Major changes were included in the sales process as well. To make navigation flow simple and decrease the amount of information (Lenard, 2018) we decided to drop the

modal with step-by-step functionality. Instead, we created a separate view that covers the whole page and includes only the number of the trees and simplified version of update for the user (Figure 19). We now use the switch button to indicate whether the user wants the update or not, instead of the question where the seller has to press yes or no. The code for the second prototype was very long and complicated, the modal had too many steps. Here we have everything in one place instead of jumping between the steps (if the user for example decides for changes during the process). Since the sales process has only two important components: the tree amount and the contact form, we wanted to have it on one page.

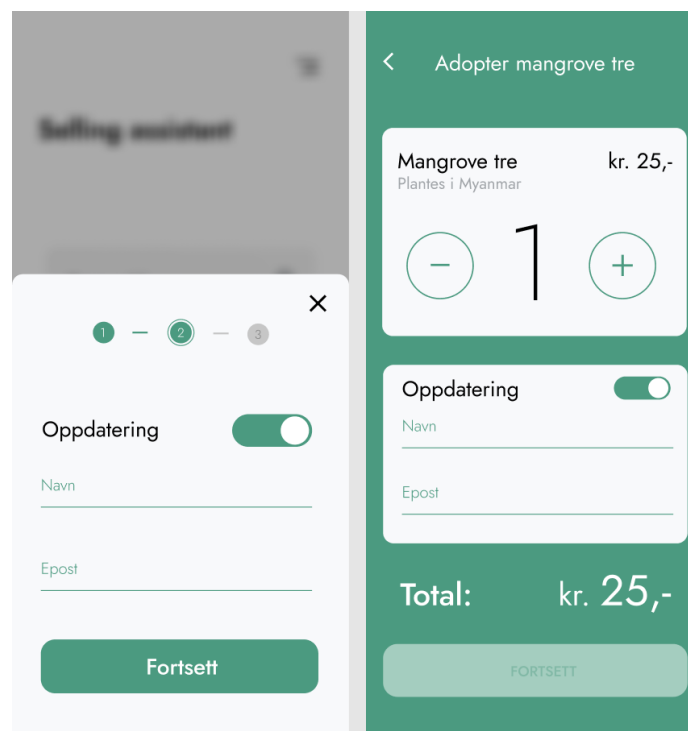


Figure 19: Left: old sales process. Right: third prototype sales process

Until this moment we were considering Vipps QR code as a main payment option. However during the programming process we found the size of the URL sent to us from Vipps was too long to generate a scannable QR code. Also since we had to add the phone number during the sales process, we agreed it would be better to drop the QR option and focus on the payment with the phone number instead.

4.6.2 User testing

First round of user-testing

The testing went very well and the feedback was uplifting. The three test-subjects liked the solution, found it simple and pleasant, and one reported: *“I wish I could use this instead of selling my daughters tickets on Facebook, so neat!”*. The test-subjects generally appeared to like the design, found the app easy to use and minimalistic.

In this screencap of the app (Figure 20), testers commented that it was unclear that the number representing the amount of trees to sell was directly editable.

Along with that, we noticed a lot of confusion regarding the “Ønsker oppdatering?” form, where users could not tell if any of the options were optional or not. This new layout did prove however that the overall view of the sales process as one was a good choice.

Second round of user-testing

Based on previous results we only made changes to the sales process. We now added an underline on the input for how many trees are to be sold for an input visualisation, we disabled the update option on entering the page so that the only thing our sellers need is to enter the phone number. When the customer wants an update, the inputs are the same as the previous version, but with an asterisk at the end of “E-post” to indicate that it’s required information.

If our seller is on multiple teams we need a way for them to enter which team the money goes to. The easiest and cleanest solution would be to have a dropdown selection. Now the seller simply selects which team it goes to and then they proceed.

The screenshot shows the 'Adopter mangrover' app interface. At the top, there is a back arrow and the title 'Adopter mangrover'. Below this, there is a card with 'Mangrove' on the left and 'kr. 25,-' on the right. In the center of this card is a quantity selector with a minus sign in a circle on the left, the number '1' in the middle, and a plus sign in a circle on the right. Below the quantity selector is a horizontal line. The next card contains the text 'Ønsker oppdatering?' followed by a toggle switch that is currently turned off. Below this is the text 'Oppdatering sendes ikke'. The next card contains the text 'Telefon nr*' followed by an input field with a horizontal line. The next card contains the text 'Hvilket lag skal pengene til?' followed by a dropdown menu showing 'Aalesunde Fotball klubb 12' with a downward arrow. At the bottom, there is a 'Totalt:' label on the left and 'kr. 25,-' on the right. Below this is a large grey button with the text 'FORTSETT'.

Figure 20: Changed inputs

Three subjects were tested. The findings this time was that the input number is too big. Whenever subjects were asked to sell 150 mangroves, the existing input wasn't removed when typing the new value, creating 1501 mangroves to sell. This issue didn't occur during the previous round since the test subjects removed the existing value themselves. Other findings indicated that our payment choice modal wasn't intuitive enough, even though this finding is on one test-subject it is something to consider changing.

4.7 Dashboard Prototype

Dashboard was the last thing to design since we had to make sure what kind of data we were able to collect for further presentation. Based on the data sellers get during the sale we decided to present an analytical approach for both admins and moderators. We used Chart.js to represent the data in a donut chart and a bar chart. Donut chart includes total earning per team in the organization while the bar chart shows total daily organization income. As Taras stated in his article *“using visualization to compare one or many values sets is so much easier than looking at numbers in the grid”* (Bakusevych, 2018). Every component on the dashboard is wrapped in a card for better readability (Figure 21).

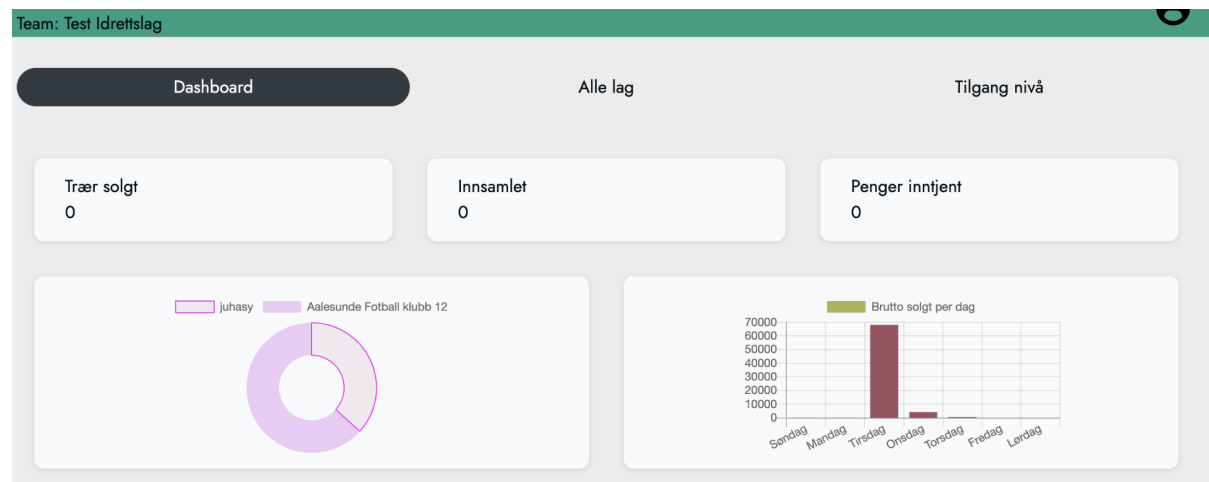


Figure 21: Dashboard prototype

For bigger groups of data like lists of all teams and their members we created tables. Table “Alle lag” (Figure 22) shows all the teams within the organization (in admin view)

or the teams a coach is assigned to (moderator view). Based on the organization roles (attachment B) org_admin/moderator can add, edit, remove teams from organization but also manage the members of particular teams. We decided to focus on the most important data, and show only names and phone numbers of the team members. To make the view more simple, the user sees the teams' names inside cards that can be spread by clicking on the toggle button. All the team members are shown when the card is spread.

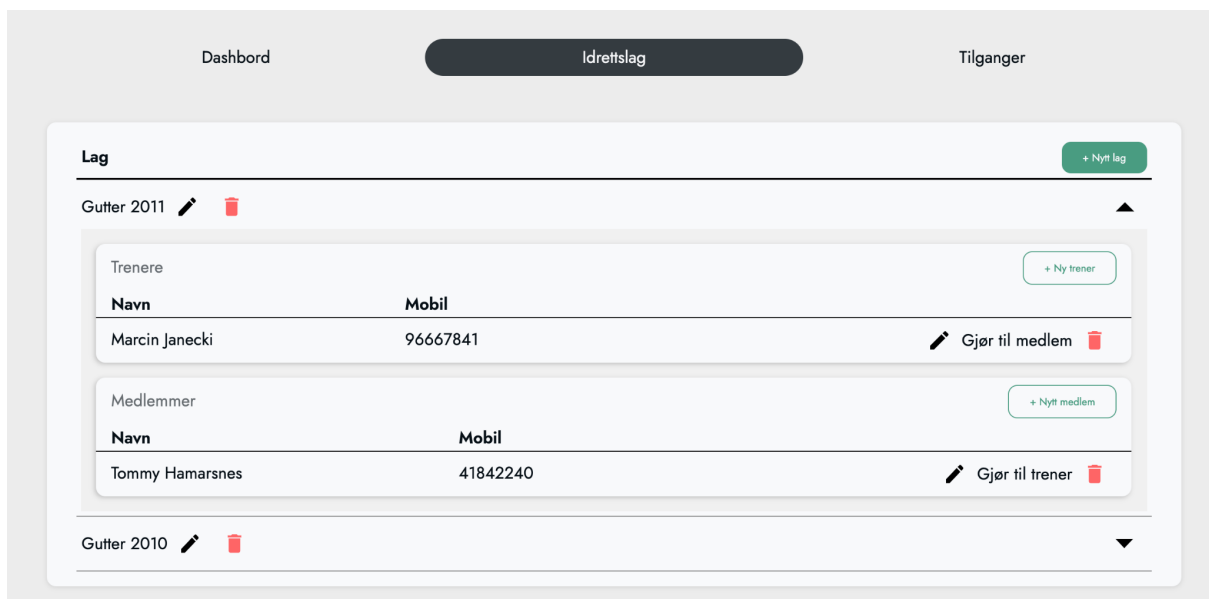


Figure 22: Dashboard table first design

We didn't find time to test the dashboard outside of the group. Therefore the final look is mostly based on the principles we mentioned in 2. Literature review and our intuition.

5. Solution

In this chapter we represent our solution with the focus on layout and functionality but also how the app architecture had an impact on it.

As mentioned in 1.4 *Expectations from the Client*, the currently delivered state of the app is by no means a complete and finished product, but rather an MVP meant to represent what a real solution could look like.

While the app's code is written in English, the language displayed in the app is Norwegian as it's expected to be available on the Norwegian market only.

5.1 Architecture

The architecture of the app is based on common patterns of hierarchy within sports organizations.

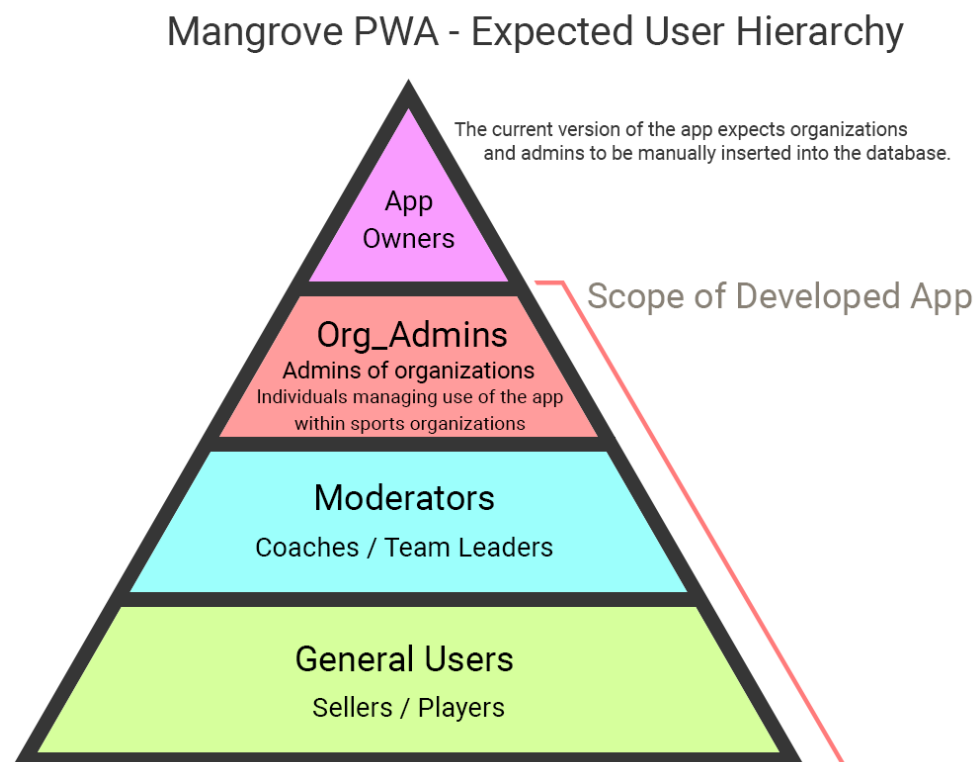


Figure 23: The hierarchy of the accounts within the app

Note: The scope of the developed app represents the parts we have developed over the course of the project. Future development would include app owners as well.

We chose to create roles based on this hierarchy to define user permissions and the primary tasks each “role” had to fulfill in the system, in a way that accurately reflects how sports organizations are structured.

Each level of this hierarchy has the primary task of maintaining and controlling the role underneath them in the hierarchy. App owners are responsible for creating and managing org_admins, who are responsible for controlling the moderators and coaches who then again are responsible for controlling the teams and sellers themselves (Figure 23).

5.2 Layout and functionality

Figure 24 represents the layout and options available to the users in the final state of the app compared to the expected layout shown in attachment H.

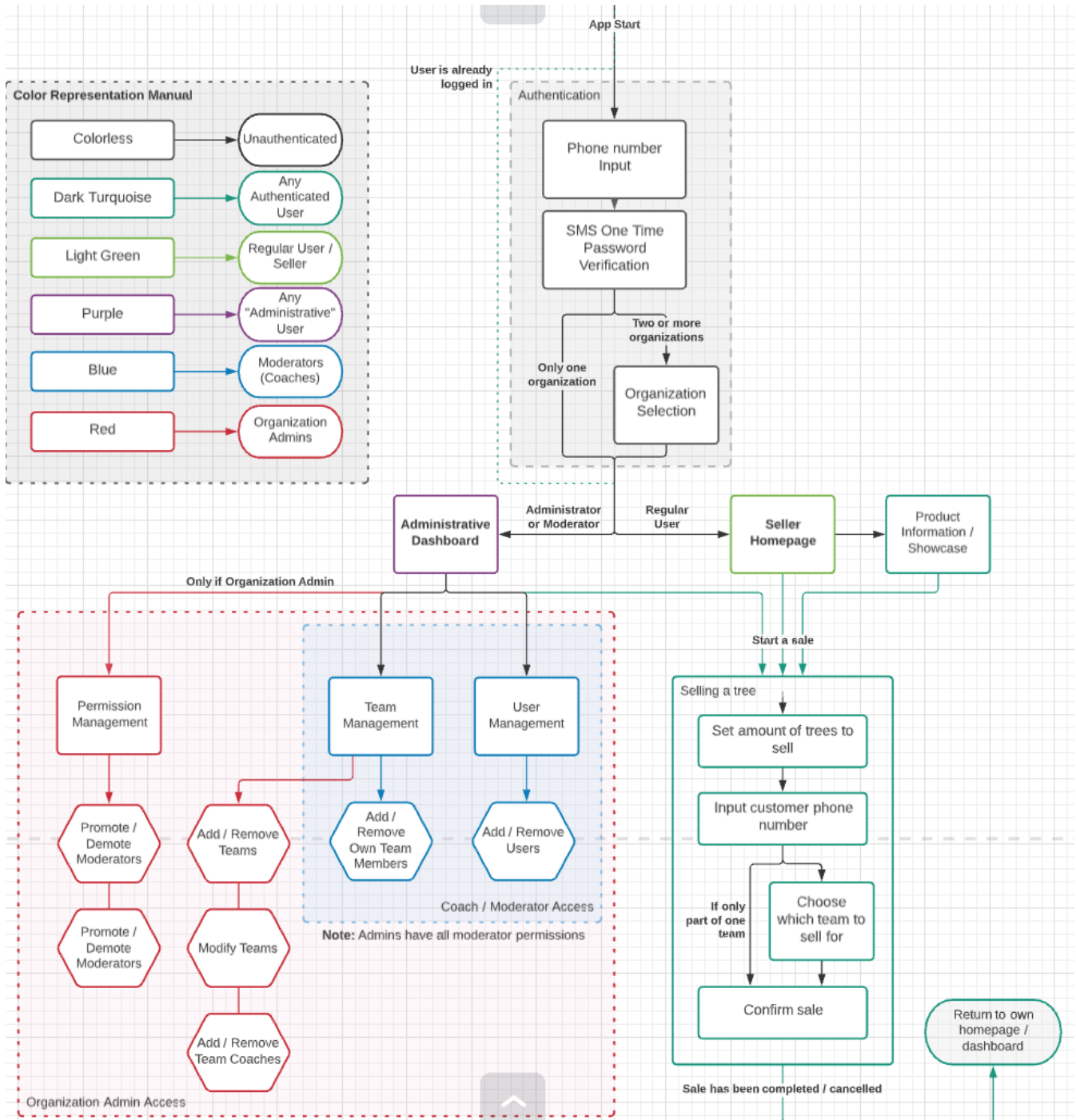


Figure 24: App flowchart and layout based on the currently developed version of the app

Sales assistant interface (Figure 25) was meant to be displayed mostly on mobile devices, that's why the whole page is in a narrower container also on the desktop devices. We wanted to achieve a native app look here, with as little scrolling as possible to have everything easily accessible.

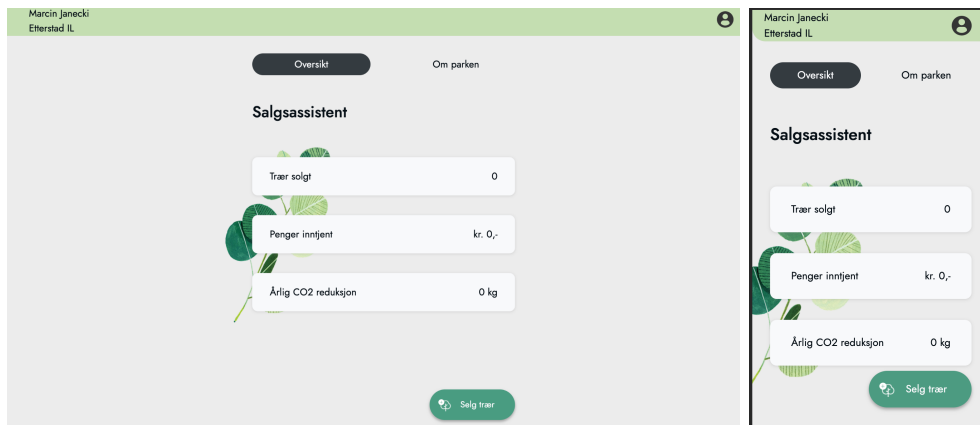


Figure 25: Left: Sales assistant on the desktop. Right: Sales assistant on mobile.

After the last usability testing we signified the input field for the amount of the trees. Following the Material Design rules in the contact form, the user gets proper feedback if one of the required fields is not filled correctly. The form changes color to red and the error message appears below the input field (Figure 26).

Figure 26: Contact form with input validation. From top: selected, filled correctly, error

Payment method button was changed into the default Vipps button by vipps (Figure 27) to show more contrast on the white background of the modal (Vipps, n.d.).

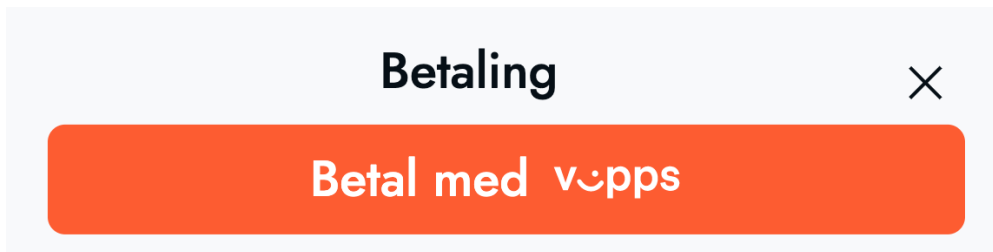


Figure 27: Vipps button in the payment methods modal

Administrative interface of the app is represented in a form of a dashboard. As mentioned before it's only org_admins and moderators that have access to this part of the app (attachment B). Unlike the sales assistant, the administrative interface is fully responsive on the desktop devices (Figure 28).

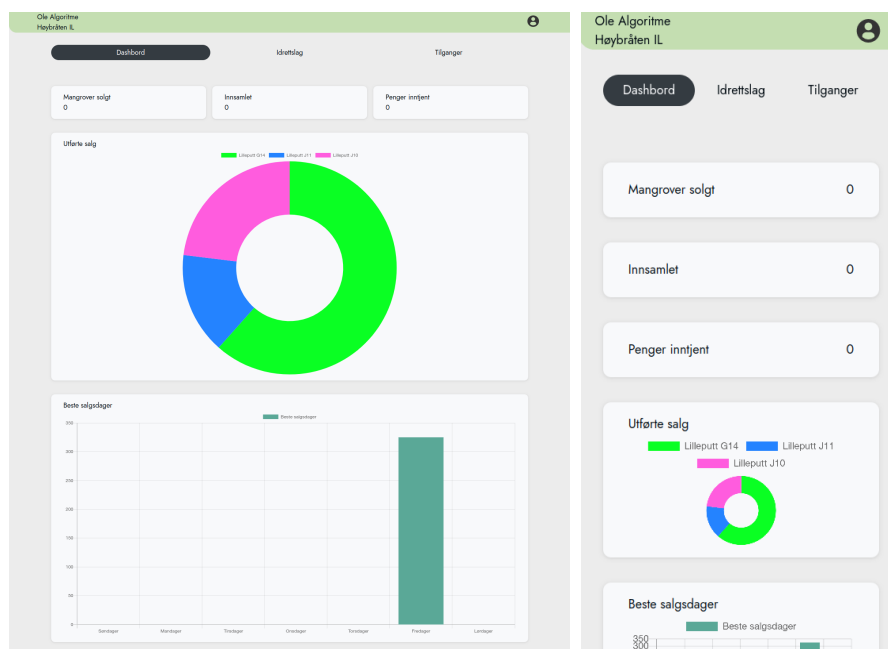


Figure 28: Left: Dashboard desktop view. Right: Dashboard mobile view.

When adding a new user to the team table, we first verify if the user already exists in the database. If so, the result shows the name of the existing user and this user can be added to the team by clicking the button “Legg til laget” (Figure 29).

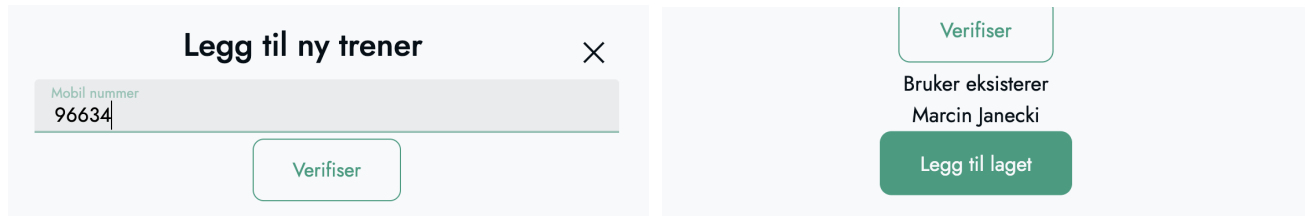


Figure 29: Left: User phone verification. Right: Add existing user to the team

If the user doesn't exist the form shows up where name and phone fields are required. The organization role of the newly added user is set to “user” by default. Then org_admin can change it in the access table. Org_admin can also remove the user from the organization, but not from the database as we decided that this task belongs to the App_owners only. By clicking on the profile icon a menu card will pop-up (Figure 30) where users can navigate between the administrative part and the sales assistant part, but also log out from the app.

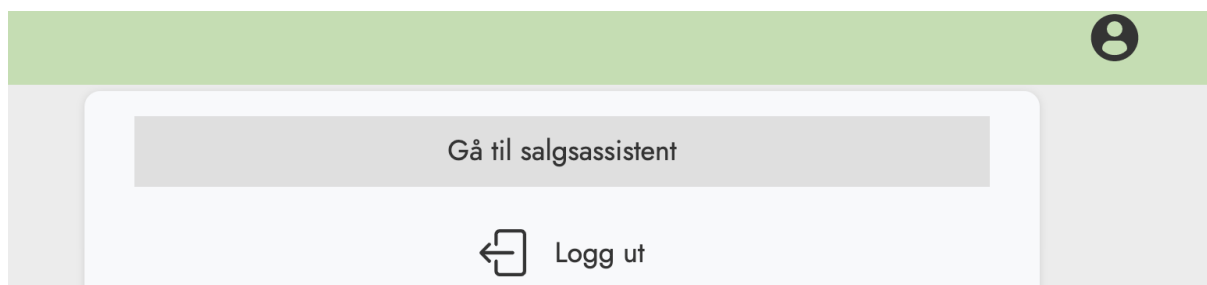


Figure 30: Menu card and profile icon.

5.3 Database

Designing the database was challenging, as we were not familiar with the structure of the sports organizations. We interviewed a member of one sport organization and the information we received helped a lot in shaping the roles for the solution..

In short we have 3 access levels for the sports organizations, highest to lowest;

- **Org_admin:** Organization leaders
- **Moderator:** Coaches / Team leaders
- **User:** Sellers / team-members / any registered account

Anyone with access level higher than **User** has access to the Sale process.

We also have **App_owner** which is reserved for Waterdrop to add/approve organizations to the platform later on. For more details about the access levels and what they are allowed to do, see attachment *B*.

Other data is also stored such as OTP codes for login.

For the customer it's only saved on phone number for transaction tracking in case of errors, name and email for update on their purchase if needed. For the ER structure of the database, see attachment *A*.

5.4 Backend API Services

To have a functional application we use our own API services that performs identification of users (Authentication API) and payment processing (Payment API). Figure 31 represents the separate backend services.

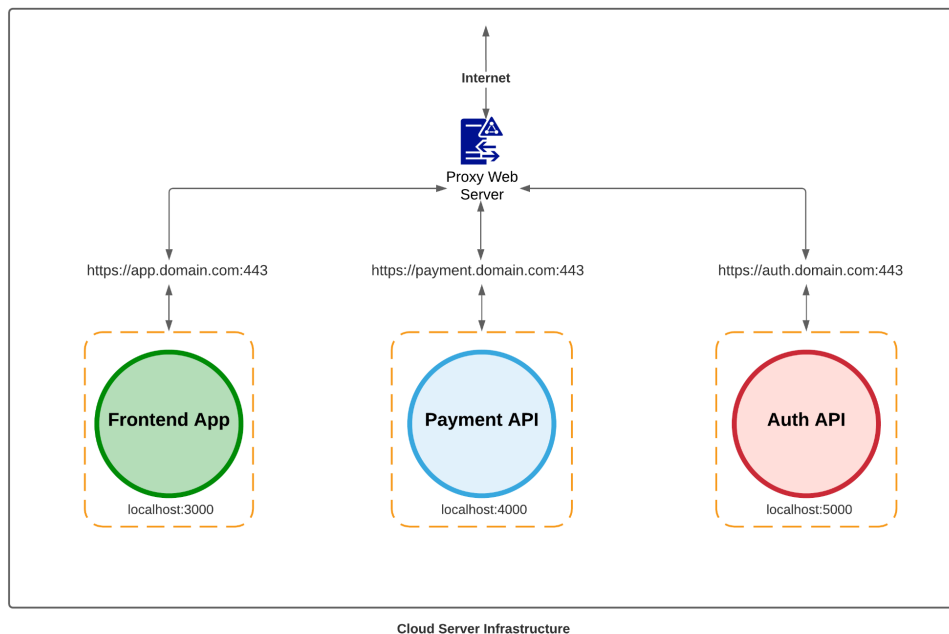


Figure 31: The three components of our application in a server environment

5.4.1 Authentication API

JSON Web Token (JWT) is a “compact, URL-safe means of representing claims between two parties” (IETF, 2015) . JWT is our chosen form of Authorization token used by our Authentication API (Auth API). Our group did not want to rely on third party authentication solutions, as this comes with the added risk of relying on a third party code base being maintained for the unforeseeable future. The Auth API is written from scratch. To identify users from the front end application we use SMS One-Time-Passwords (*SMS OTP*) that returns JWT tokens. The reason for choosing JWT is that “JWT token authentication is considered the *de-facto standard*” (Kukic, 2016). Summarized: Auth API generated JWT tokens are sent in response to a successful login request initiated by the end user.

The authorization process consists of returning a cryptographically signed JWT token to the user. Inside the signed JWT token we store a set of claims: the user id and the given

user role, along with organizational information - which is fetched from the database. Once the JWT token is received it is set in the HTTP authorization header and used for each consequent request made by the user. The front end application decodes and verifies this JWT token's authenticity before rendering the appropriate data associated with the user and given user role (Figures 32 and 33).

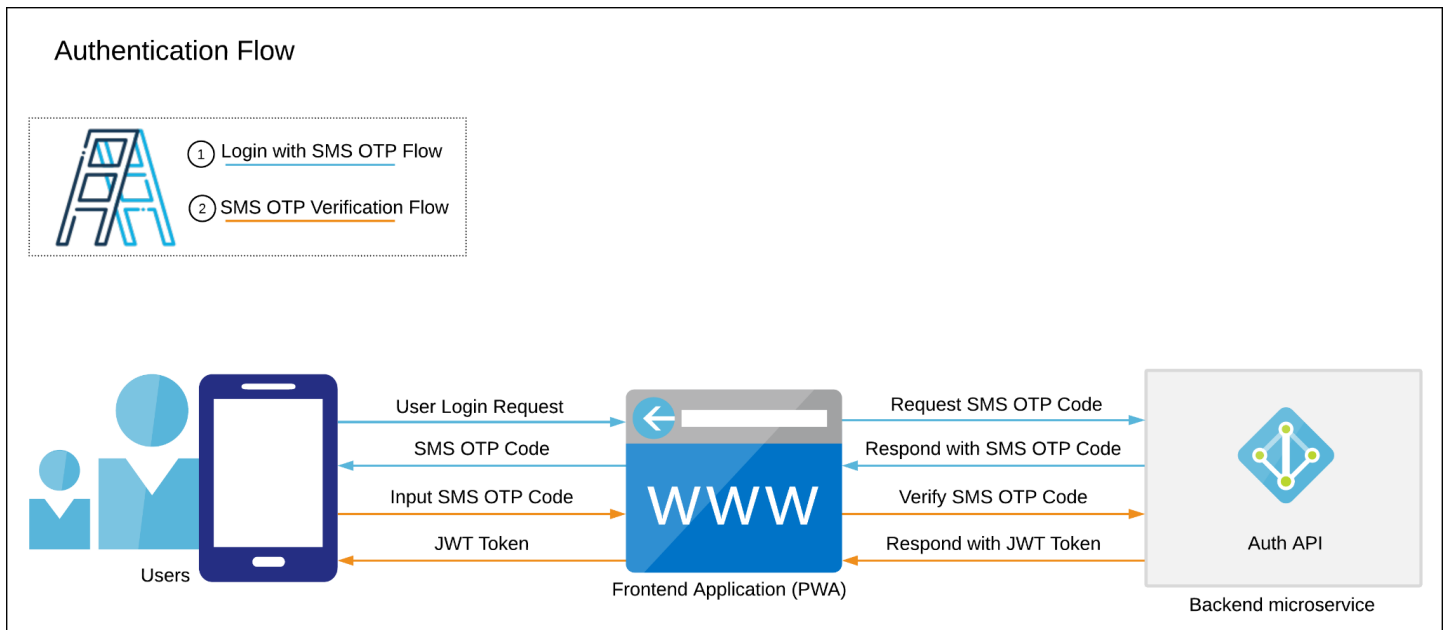


Figure 32: A visual representation of the Auth API handles authentication through SMS OTP.


```

jwt.ts TS
32 import jwt, {Algorithm} from 'jsonwebtoken';
31 import CONFIG from '../../config';
30 import {OrganizationRole, User} from './types';
29
28 class JWT {
27
26   private static algorithm: Algorithm = 'RS512';
25   private static privateKey: string = CONFIG.JWT_PRIVATE_KEY;
24
23   public static getJWTSignature = (user: User, orgRole: OrganizationRole, issueDate: Date, expireDate: Date): string | null => {
22
21     let tokenString;
20
19     try {
18       const claims = {
17         "sub": user.id.toString(),
16         "name": user.name,
15         "phone": user.phone_number,
14         "country_code": user.country_code,
13         "organization_id": orgRole.organization.id.toString(),
12         "organization_name": orgRole.organization.name,
11         "organization_teams": orgRole.organization.teams,
10         "iat": Math.round(issueDate.getTime() / 1000),
9         "exp": Math.round(expireDate.getTime() / 1000),
8         "https://hasura.io/jwt/claims": {
7           "x-hasura-allowed-roles": ["public", orgRole.given_role],
6           "x-hasura-default-role": "public",
5           "x-hasura-role": orgRole.given_role,
4           "x-hasura-user-id": user.id.toString(),
3           "x-hasura-org-id": orgRole.organization.id.toString()
2         }
1       }
33
1       tokenString = jwt.sign(claims, JWT.privateKey, {algorithm: JWT.algorithm});
2
3     } catch (err) {
4       return null;
5     }
6
NORMAL | auth-api | api/v1/auth/jwt.ts
1 | 1 | nvim

```

Figure 33: Code illustrates how we generate JWT token based on user role in an organization

SMS OTP Authentication Process

End users initiate the authentication process inside the front end application by entering the user's phone number. Button click triggers a HTTP POST request to the `/api/v{versionnumber}/auth/sms` endpoint with the user's phone number (Figure 34). The request is received by the Auth API, which in turn generates a 6 digit code that is inserted into our custom SMS text message. This message is passed on to our SMS provider (Twilio), which in turn sends the SMS to the end user (Figure 35).

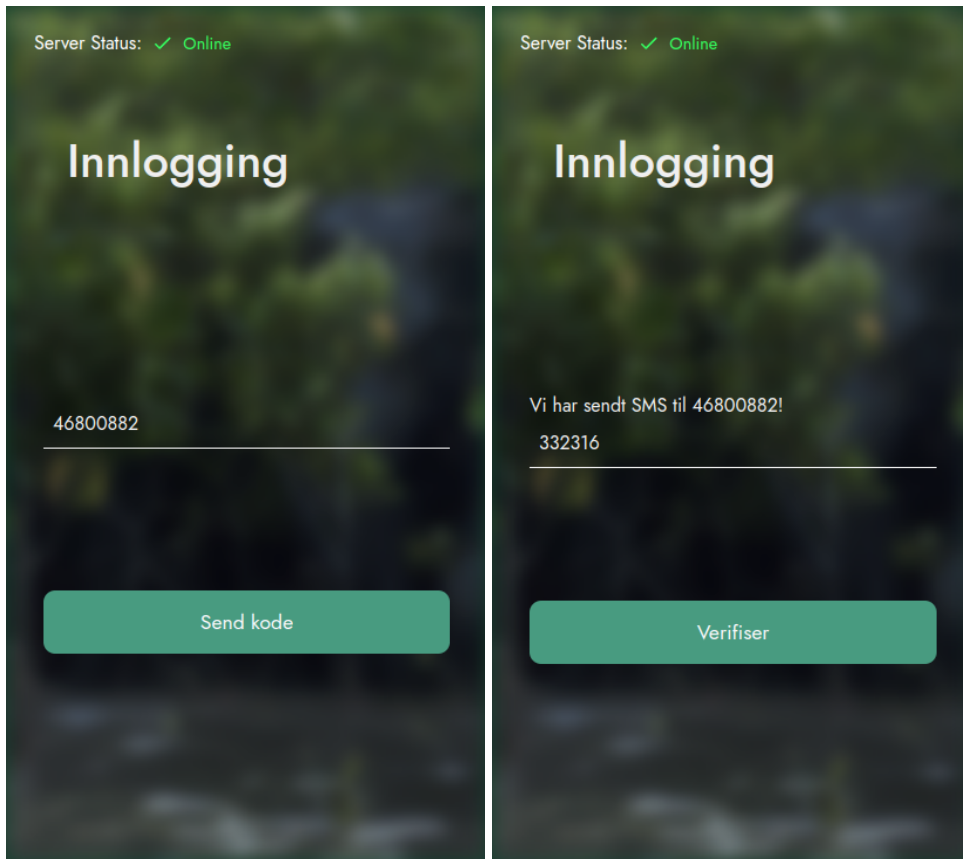


Figure 34: SMS OTP login procedure.

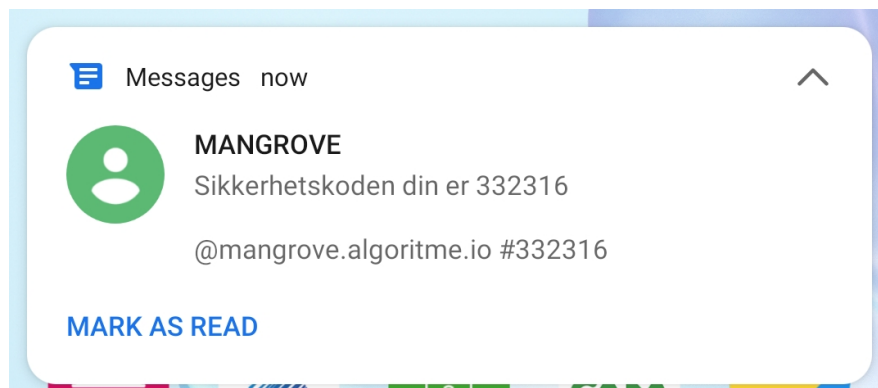


Figure 35: Received SMS OTP.

On successful receipt of an SMS OTP code, the user proceeds to enter this OTP code through the front end application followed by a button click. The button click triggers a new HTTP POST request with the entered OTP code to the `/api/v{versionnumber}/auth/sms/otp` endpoint of the Auth API. Once the code is received by the Auth API it tries to match the generated code with the received code. If the received code matches the generated one, and if it was entered within 5 minutes

from being generated, the Auth API verifies the user’s identity and automatically generates the associated JWT token. The generated JWT token is returned to the front end application and saved as a cookie within the end user’s browser.

5.4.2 Payment API

The functional prototype is based on using the testing environment of Vipps API provided to us by Bouvet.

The back end receives a POST Request from the Front end with the required data to the endpoint `"/api/v{versionnumber}/payment"`. The back end then handles the data and sends the necessary data to Vipps. Vipps then handles the data from us and sends back the payment url. This payment url is then sent back to the front end for the seller to initiate or cancel the payment request they started. After the seller sends in the request, the rest of the transaction is in the hands of the customer. The customer then chooses whether or not to complete the purchase (Figure 36).

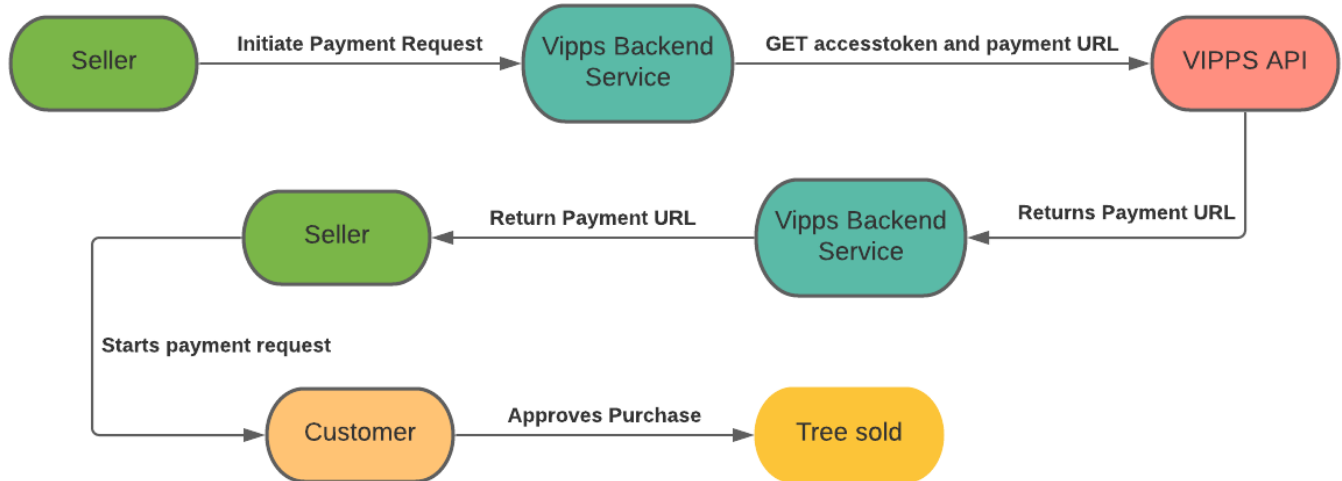


Figure 36: Vipps flow

During this process there are two parts that have a time limit to be initiated by the seller or the customer before the process is set to be cancelled. One is when the seller receives

the URL provided by our back end after receiving it from Vipps, and the other is after the customer has opened the Vipps app after receiving the payment request.

After the seller requests the payment the back end is in charge of doing a 2 seconds interval where it checks the current status of the purchase. Only when the status is set to RESERVE can our back end capture the reserved money to be transacted.

5.5 Unit Tests

In an effort to maintain a certain level of quality in our codebase as well as ensure our code is performing the way we intended, we wrote unit tests that simulate the app's runtime and functions and inputs a regular user would perform, such as clicks and text inputs.

As writing tests was not a high priority during development, as of the end of the project it covers about 45% of the app's front end functions, mainly covering authentication and login, as well as the sales process (Figure 37).

The same type of unit testing is conducted in the backend solution as well, although there it also tests for the presence of environment variables and that functions in the backend are working as intended (Figure 38).

For our version control on github, these tests are automatically run with each pull request, ensuring that all primary functions still work with each iteration of the app.

```
Test Suites: 5 passed, 5 total
Tests:      8 passed, 8 total
Snapshots:  0 total
Time:       12.319 s
Ran all test suites.
```

Figure 37: Frontend passing test results

```
Preflight env keys and value
✓ subscription-key
✓ client-id
✓ client-secret
✓ test-number
✓ msn
✓ pincode
✓ graphql-endpoint
✓ hasura-admin-secret

8 passing (7ms)

POST /payment
Server is listening @ localhost:8668
```

Figure 38: Preflight checking if environment keys exist and have values.

5.6 Further development

Regarding the client's expectations, we accomplished the main goal set for us which was the sales assistant interface as an MVP.

The next step would be to add an interface where the owners of the app can accept incoming requests from sports organizations and where unregistered organizations can apply to use the app for their teams to sell trees. This will put the app in a state where it can be used without any direct interaction with the database through the hasura console, making it an MVP of the entire solution, as opposed to just an MVP for parts of the system.

Along with this, the team also has a list of future features to potentially implement to improve the quality of the app, as well as provide the users of the app with more options to conduct their sales. Most of these potential features come from suggestions we discussed among ourselves as well as requested features we discovered from our subjects during user-tests.

Among these potential future features, these are some that are particularly interesting:

- A way for *any visitor* of the site to order trees freely without having to log in or be part of any organizations or teams
- Adding more payment options for the customers
- A way for customers to visit the site and “look up” their purchased mangrove trees

Our client also had ideas for future versions, some of which included functionality such as:

- A way for the app owners to send invitations to various organizations to use the app
- Ways for the app owners to see their app usage statistics and how active sales are

6. Discussion

In this part of the document, we will reflect over the decisions we've made, the tools we've used and discuss how those choices affected the overall state of our bachelor.

6.1 Assessment of the solution

While our solution accomplishes the goal of allowing users to sell mangroves and raise funds for organizations, our solution covers only the bare minimum of what is necessary to accomplish these goals.

6.1.1 Design

During the project it became more significant to us that Material Design rules we chose to follow were covered with test-subjects expectations. On some parts, like the floating action button without a label, we decided to trust our intuition, yet the survey (attachment F) and then meeting with UX designer proved that not everyone would understand it the same way we do. Therefore adding the label made the button more understandable. The forms weren't initially following any rules, they were just made with intuition. Although no particular complaints appeared during the usability testing, applying both Material Design and UD recommendations helped distinguish the text fields from the rest of the view and we added more reasonable placement for the errors.

6.1.2 Technical solution

One of the largest questions we faced from the start has been, "How do we develop a 'sales-app' that is not in the hands of the customer, but the seller?".

Our solution to this question has become a POS system that closely resembles that of generic door-to-door sales approaches, where sellers ring the doorbell and ask the customers if they would like to purchase their product.

Our case was provided to us with the context that these sales would be carried out by sports teams and children who belong to those, who typically do run door-to-door trying to sell products, and we believe we made the right call in not deviating too much from that approach towards the customers.

6.1.3 Technical implementation

We spent a considerable amount of time on implementing the Auth API as a backend service. The reasoning behind this decision was to avoid relying on third party providers and instead prioritizing the privacy of our users. On top of this, we wanted the sender ID of our SMS OTP messages to match our application name to establish more user trust. By implementing our own solution that avoids authentication by email, we diminish sharing any personal information with third parties, and more reliably verify the identities of our users.

6.1.4 Benefit for the client

The biggest benefit our project has brought to our client is the overview over what is needed as a whole to develop a solution for WIF. It created a largely beneficial scope of the required infrastructure, people and systems for the concept to become a reality. It also provides examples for how the solution can be designed as an app, what data is required from the system's users, and information about the hierarchy and structure of the app's expected user base.

6.2 Assessment of process and method

Although the bachelor semester has progressed in an unusual way as a consequence of the pandemic, our chosen methodologies and tools have for the most part been common practice, except online as opposed to in-person.

6.2.1 Our use of Scrum and Kanban

Jira, the solution we chose as our platform for backlog maintenance and task boards ended up being a decent solution to keep track of our backlog and project requirements. Although it ended up very common among all team members to forget to update the tasks they had started on or completed. In light of the pandemic restrictions, each member of the team worked from home, which resulted in a lower overall quality and amount of communication and made it easier to forget to update the task board.

In the end, we believe we relied less on following what was listed on the backlog and task board when deciding our next steps in the projects, and rather relied on directly

contacting other team members over our chosen communication platform to inform ourselves on what was done and what was not.

Despite this, we periodically reminded ourselves to return to the backlog and task boards during our sprints to get a documented record of our progress.

In the future we would ideally be more disciplined about conducting daily meetings, backlog and task board updates.

6.2.2 Project plan

In our project plan, we managed to determine early on that the extent of which we could develop the app was largely unknown, as we were unfamiliar with the environment we were developing for, and how to approach the problem. Instead, we set up expectations for how to proceed with developing our solution more practically, and we have generally stuck to that plan the entire way through. Some exceptions to the plan were made, such as when we dedicated certain weeks, or periods of the project for information gathering and documentation, as opposed to progressing on app development.

This is reflected in our roadmap (figure 4) on Jira, in which the breaks between the development sprints can be classified as either weeks dedicated to gather information or progress on our documentation.

6.2.3 Risk plan

In our risk plan, we defined a set of potential obstacles we could encounter during the project, and estimated their likelihood, impact, and method to mitigate or avoid them. Generally, we agree that our estimations and predictions were accurate to a certain extent, however we also believe we greatly underestimated the impact the ongoing pandemic had on the group's performance.

With every form of communication being online and a lack of in-person meetings, it became visibly harder for the group to work as a team. Aside from occasional technical difficulties, having to work separately from home showed the differences in what times of the day some members were more comfortable working in, and we figured out as the

project went along, that the best way for us to progress was to let each member work on their tasks when it best suited them.

Working this way came with the drawback that our work became a bit more separate than we would have preferred, and each individual member struggled more with keeping up with the total state of the project.

6.3 The research basis

One of the most challenging parts for us was finding relevant research topics we could base our application on, and we believe a large part of that was due to our narrow view when it came to searching for topics.

Choosing PWA as the main solution seemed to fit everyone in the group, but also the performance of the final version of the app proved that native features and code implementation was not necessary.

We found that our research on the design part was very helpful during development of the app. Many principles from the Norman Group regarding usability helped us to build a satisfying product that was appreciated by the test-subjects. Following POS rules contributed in shaping a simple selling app where the user is not overwhelmed with information. Both Material Design and UD provided a necessary theory regarding the design of the components. We got positive feedback from the user testing which also proved that the choices we made here were relevant.

6.4 Assessment of tools and technologies

For tools we used to help us progress with the project, we had fairly good experiences with the choices we made.

Discord, our main platform for communication helped us greatly in organizing our discussions into separate threads, and let us easily read back on discussion logs and search for information posted earlier in the project.

Jira, the platform we used for our Scrum backlogs and Kanban task boards offered all of the tools we needed to conduct proper agile sprints, although we found the service's

lists, roadmaps and task boards to look good, some of us also found the service was performing fairly slowly, and would consider looking for better performing alternatives in the future.

Google Docs is the platform we used for writing our report, and the more we wrote the slower it became. This added to the frustration for all of us especially at the end of the project phase when the five of us and external supervisors are visiting the document at the same time. The comment history function also had issues, as when we were trying to close the history it would not be closed and the whole document performed slowly.

6.5 Assessment of results in relation to the group's goals

When it comes to meeting our goals and expectations for the project, we understand that we set the bar for an ideal project higher than what we could achieve given the amount of time we had.

We underestimated how much time we would need to spend gathering information we needed to develop certain parts of the app, such as our expected user hierarchy and how the sports organizations we were building the app for were structured.

In the end, we felt we had to cut back on our scope of development and did not manage to accommodate “every” relevant user we planned to build the app for. Despite having to cut back on the scope, the core features are working and the remaining requirements can easily be added in future implementations.

We primarily focused on the “sellers” of the app, and added the required functionality for their administrators, but did not develop any interface for the owners of the app to confirm applications to use the app or any way for organizations to request access to it.

We did however feel that we learned a lot about the real requirements of app development, and how much time the development team needs to spend learning about who they are developing for, and how much time a team must spend planning out their application before real development begins.

6.6 Challenges

Throughout the course of the project, we experienced several challenges related to the development of the app, as well as general challenges tied to the project.

An overwhelming majority of these obstacles and challenges are tied to the ongoing Covid-19 pandemic, which has had an impact on almost all aspects of our project, from the basis of how our app should be used, to our own internal teamwork.

6.6.1 Everything is Online

Being forced to communicate online and mostly from home due to pandemic restrictions has impacted the team's motivation, availability and ability to communicate greatly. Despite chat-logs allowing us to keep a permanent record of what is being discussed, it didn't outweigh the quality in-person discussions provided in terms of communication.

We felt that technical difficulties were harder to resolve and the quality of teamwork was reduced when not given the ability to directly cooperate in the same environment.

6.6.1.1 Google Design Sprint

We had to hold the Google Design Sprint remotely, which made us aware that properly conducting it the way it was intended would become very challenging.

Arranging meetings or having the opportunity to communicate with the experts (GV, n.d.) became difficult within the short time constraints the GDS presents, and we started to understand that these challenges would persist throughout the week.

Early in January we were still missing a lot of critical information in order to understand the problem we needed to solve. After the design sprint, we had a meeting with WIF representative, Andreas Luksepp, in which we learned more about the requirements of the app.

In the end, the GDS we conducted was only marginally useful to us in this project, and mostly contributed to giving us ideas for what a general "door-to-door" sales solution could look like, with only minimal relation to what the actual users of our application needed.

6.6.2 Learning about the unknowns

During development of our app, we spent a lot of time learning about who we were developing the app for. It was particularly important for us to learn about how sports organizations were structured in order to build our own role based permission system.

Due to restrictions we had to rely on testimony we received from our client as well as our user test participants.

We found this obstacle interesting as it provided us with a lot of insight into how much research is needed to develop real life solutions and is likely to be valuable to us as future developers.

6.6.3 Developing with security as a focus

Our application deals with monetary transactions, which is why we put security in a large spotlight when developing our application. This was challenging as most of the team had very little experience in developing solutions where this level of security was necessary. We found ourselves spending a lot of development time learning about ways to secure calls to our own back end, as well as securing our user inputs and permissions in just about every layer of our system.

7. Conclusion

The project and its results shows that we have met the main requirements set by all of the parties involved. The original scope was giving organizations the ability to sell mangrove trees either door-to-door or remote with the application we have developed.

We have also managed to develop features outside of the MVP that were not part of the original scope. We added the possibility for the organization, WIF and coaches to manage the sellers of the app and keep track of income and amount of trees sold with an admin dashboard.

We have gained insight in user behaviour through user-testing and surveys, as well as insight about how the industry works. We have been cooperating with our customers throughout the project with weekly meetings. We learned about how app development can take place in reality, especially the parts relating to what is unknown to the development team, and how much infrastructure behind the app can influence and change it's outcome.

References and literature list

- Bakusevych, Taras. 2018. "10 rules for better dashboard design". UXPlanet. Accessed April 6. 2021.
<https://uxplanet.org/10-rules-for-better-dashboard-design-ef68189d734c>.
- Biørn-Hansen, Andreas, Tim A. Majchrzak, and Tor-Morten Grønli. 2017. "Progressive Web Apps: The Possible Web-Native Unifier for Mobile Development:" In *Proceedings of the 13th International Conference on Web Information Systems and Technologies*, 344–51. Porto, Portugal: SCITEPRESS - Science and Technology Publication. Accessed February 9. 2021.
<https://doi.org/10.5220/0006353703440351>.
- BuiltWith. u.d. "React Usage Statistics". Accessed May 4. 2021.
<https://trends.builtwith.com/javascript/React>
- Calzavara, Stefano, Alvisè Rabitti, and Michele Bugliesi. 2021. "Dr Cookie and Mr Token - Web Session Implementations and How to Live with Them". Accessed March 24. 2021 <https://core.ac.uk/download/pdf/153147345.pdf>.
- Cardieri, Giulia de Andrade, and Luciana Martinez Zaina. 2018. "Analyzing User Experience in Mobile Web, Native and Progressive Web Applications: A User and HCI Specialist Perspectives." In *Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems*, 1–11. IHC 2018. Belém, Brazil: Association for Computing Machinery. Accessed February 9. 2021.
<https://doi.org/10.1145/3274192.3274201>.
- Chang, Yen-ning, Youn-kyung Lim, and Erik Stolterman. 2008. "Personas: from theory to practices". Accessed January 20. 2021.
<https://dl.acm.org/doi/10.1145/1463160.1463214>
- Cherry, Kendra. 2020. "How Does the Color Green Impact Mood and Behavior?". Article. Verywell Mind. Accessed February 17. 2021.
<https://www.verywellmind.com/color-psychology-green-2795817>.
- Clemente, Violeta, Rui Vieira, and Katja Tschimmel. 2016. "A Learning Toolkit to Promote Creative and Critical Thinking in Product Design and Development through Design Thinking." In *2016 2nd International Conference of the Portuguese Society for Engineering Education (CISPEE)*, 1–6. Accessed February 15. 2021.
<https://doi.org/10.1109/CISPEE.2016.7777732>.

- Datatilsynet. n.d. "Personvernprinsippene." Datatilsynet. Accessed March 24. 2021.
<https://www.datatilsynet.no/rettigheter-og-plikter/personvernprinsippene/>.
- Deloitte. 2019. "Nordic Mobile Payment Report 2019 – Chasing Cashless." Deloitte. Accessed January 22. 2021.
https://www2.deloitte.com/content/dam/Deloitte/dk/Documents/financial-services/Downloads/Chasing_Cashless-The_rise_of_Mobile_Wallets_in_the_Nordics.pdf.
- European Commission. "Do We Always Have to Delete Personal Data If a Person Asks?". European Commission. Accessed April 13. 2021.
https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/dealing-citizens/do-we-always-have-delete-personal-data-if-person-asks_en.
- . "How Should Requests from Individuals Exercising Their Data Protection Rights Be Dealt With?" Text. European Commission . Accessed April 13. 2021.
https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/dealing-citizens/how-should-requests-individuals-exercising-their-data-protection-rights-be-dealt_en.
- . "What personal data and information can an individual access on request". European Commission. Accessed April 14. 2021.
https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/dealing-citizens/what-personal-data-and-information-can-individual-access-request_en
- Farrell, Susan. 2016. "Observer Guidelines for Usability Research". Article. Nielsen Norman Group. Accessed March 10. 2021.
<https://www.nngroup.com/articles/observer-guidelines/>.
- GraphQL. n.d. "GraphQL | A query language for your API". Accessed May 4. 2021.
<https://graphql.org/>.
- GV. n.d. "The Design Sprint". GV. Accessed January 18. 2021.
<https://www.gv.com/sprint/>
- Harley, Aurora. 2015. "Personas Make Users Memorable for Product Team Members". Nielsen Norman Group. Accessed February 16. 2021.
<https://www.nngroup.com/articles/persona/>
- IDEO U. n.d. "Design Thinking." IDEO U. <https://www.ideo.com/pages/design-thinking>.
- . "What is design thinking?" IDEO U. n.d.

- <https://www.ideo.com/blogs/inspiration/what-is-design-thinking>,
IETF. 2015. "RFC 7519 - JSON Web Token". tools.ietf.org. Accessed April 28. 2021.
<https://tools.ietf.org/html/rfc7519>
- Joyce, Alita, and Jakob Nielsen. 2019. "Teenager's UX: Designing for Teens." Article.
Nielsen Norman Group. Accessed March 17. 2021.
<https://www.nngroup.com/articles/usability-of-websites-for-teenagers/>.
- Knapp, Jake, John Zeratsky, and Braden Kowitz. 2016. "Sprint: how to solve big problems
and test new ideas in just five days". New York: Simon & Schuster.
- Kukic, Adnan. 2016. "Cookies vs. Tokens: The Definitive Guide." Article. dzone. Accessed
March 10. 2021.
<https://dzone.com/articles/cookies-vs-tokens-the-definitive-guide>.
- Lenard, Dennis. 2018. "User Experience Barriers In POS Systems." Article. Usability
Geek. Accessed February 28. 2021.
<https://usabilitygeek.com/user-experience-barriers-pos-systems/>.
- Material Design. u.d. "Material Design". Material Design. Accessed March 24. 2021.
<https://material.io/design/introduction>.
- . "Material Design - Bottom Navigation." Material Design. Accessed March 24.
2021. <https://material.io/components/bottom-navigation#usage>.
- Nielsen, Jakob. 1994. "Usability Engineering". Accessed March 12. 2021.
<https://dl.acm.org/doi/book/10.5555/2821575>
- Nielsen, Jakob. 2012. "Usability 101: Introduction to Usability". Article.
Nielsen Norman Group. Accessed March 9. 2021.
<https://www.nngroup.com/articles/usability-101-introduction-to-usability/>
- Nielsen, Jakob. 2000. "Why You Only Need to Test with 5 Users". Nielsen Norman Group.
Accessed April 22. 2021.
<https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>.
- Pattakos, Aris. 2021. "Angular vs React vs Vue 2021". Accessed January 25. 2021
<https://athemes.com/guides/angular-vs-react-vs-vue/>
- ReactJS. u.d. "React - A JavaScript Library for Building User Interfaces".
Accessed May 4. 2021. <https://reactjs.org/>
- Rubin, Kenneth S. 2012. "Essential Scrum: A Practical Guide to the Most Popular Agile
Process". Upper Saddle River, NJ: Addison-Wesley.

Sandu, Bogdan. 2016. "Create a Better App by Finding the Right UI Color Scheme". Blog. Vandelay Design. Accessed March 15, 2021.

<https://www.vandelaydesign.com/better-app-ui-color-schemes/>.

Sev, Chris. u.d. "React Popularity and When Not to Use React". Article. Scotch. Accessed May 4, 2021.

<http://scotch.io/starters/react/react-popularity-and-when-not-to-use-react>.

The Centre for Excellence in Universal Design. u.d. Universal Design. Accessed January 25, 2021.

<http://universaldesign.ie/>

Tomlin, Rob. 2020. "How to Use React-Router to Create an SPA". Article. Medium. Accessed January 12, 2021.

<https://javascript.plainenglish.io/react-router-and-spas-made-easy-19fd1db0d6fc>

TypeScriptlang. Accessed February 1, 2021.

<https://www.typescriptlang.org/>.

Vipps. u.d. "Vipps i tall". Vipps. Accessed March 24, 2021.

<https://www.vipps.no/om-oss/vipps-i-tall/>

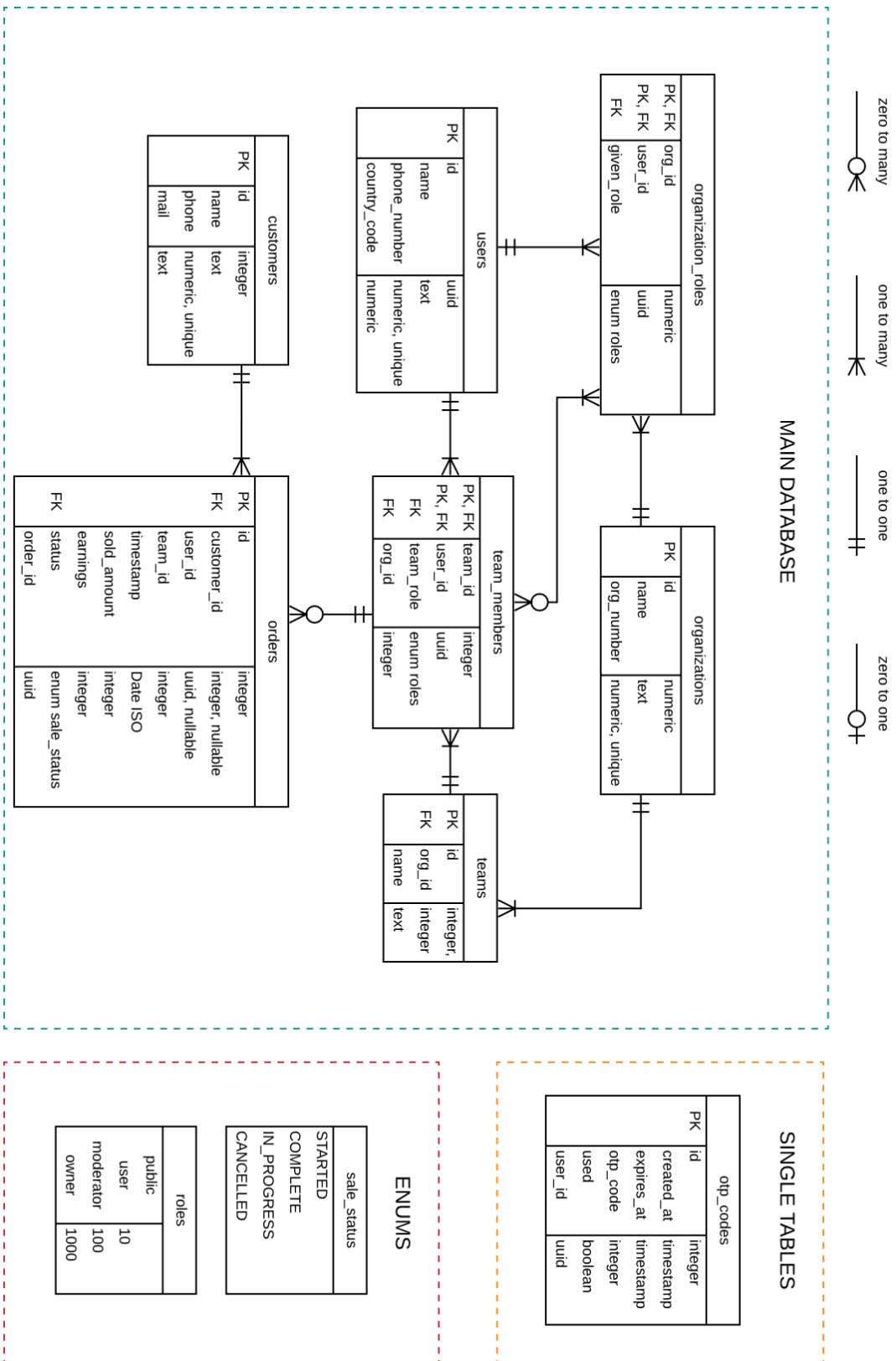
———. "Get started. Vipps buttons". Vipps. Accessed April 16, 2021

<https://www.vipps.no/developers-documentation/getting-started/design-guidelines>

Attachments

Attachment A: Database UML in Crow's foot notation	71
Attachment B: Mangrove App - Roles and permissions	72
B.1 - Solution roles and permissions overview	72
B.2 - Roles and User types	72
B.3 - Defining the baseline role uses in the app	73
B.2.1 - Users, Moderators and Org_Admins within Organizations	73
B.2.2 - Users and Moderators within Teams	74
B.3 - The User Role	75
B.3.1 - User permissions in the app	75
B.3.2 - User CRUD permissions	75
B.4 - The Moderator Role	76
B.4.1 - Moderator permissions in the app	76
B.4.2 - Moderator CRUD permissions	76
B.5 - The Org_Admin Role	77
B.5.1 - Org_Admin permissions in the app	78
B.5.2 - Org_Admin CRUD permissions	78
B.6 - The App_Owner Role	79
B.6.1 - App_Owner permissions in the app	79
B.6.2 - App_owner CRUD permissions	79
B.7 - GDPR Considerations	80
B. 8 - Sources / References:	82
Attachment C: Information letter for test participants	83
Attachment D: UI Kit	84
Attachment E: Material Design rules	85
Attachment F: Survey Sections	97
Attachment G: Group Contract	98
Attachment H: Initial App Flowchart	101

Attachment A: Database UML in Crow's foot notation



Attachment B: Mangrove App - Roles and permissions

B.1 - Solution roles and permissions overview

This document describes the various roles that exist within the environment of the Mangrove application, what they represent, and what permissions they have and need in terms of access to CRUD (Create, Read, Update, Delete) operations on the solution's database.

The permissions inside of the tables in Hasura (the service we use to handle our tables and data), are reflective of the permissions written up in this document.

B.2 - Roles and User types

Our application has a few sets of roles and user types that control permissions and affect what you see in the app.

Table 1: Each role and their respective user representation

Role	Representation
User	Basic users, in our case, the members of sports teams and their parents who will be conducting the majority of sales. Will only be able to see their own sales progress and conduct sales for their team.
Moderator	Users with responsibilities reflecting their Coach role in the organization, Moderators can create users for their "team(s)" and distribute access to the app to the aforementioned Users.
Org_admin	The sports organization administrators, those who are responsible for establishing the organization within the app to begin with. They can control and view essentially everything relating to the organization they are part of, such as creating teams and users, and managing the Moderators within their organization
App_owner	The owners of the application. They have access to view the status of every organization connected to the app, as well as managing the creation of new organizations that want to start selling their products.
Admin	A predefined role in the hasura system with all access rights to every

	operation in the database. Reserved only for development purposes, the application's backend and individuals with permission to make such edits.
--	--

Going forwards in the document, we will be referring to these roles when listing and defining their specific permissions and defining what they can or cannot see / do.

B.3 - Defining the baseline role uses in the app

We separate permissions based on the hierarchy of real sports organizations. Where the Users AKA “players” and team coaches are on the bottom level of the hierarchy, while administrators and leaders are above.

This separates common sports organizations into the “organization level” and the “team level”.

It's important to remember that any account within the app, CAN be a part of multiple organizations and teams. Such as in cases where for example; A father who is a leader within a sports organization has a child that may play for a team in a different organization, they can use the same phone number to log into other organization accounts.

Everything that happens within the app from Org_admin role and below is **isolated within each organization**. Should the account be part of multiple organizations and want to view info from said other organizations, they are required to log out and back into a different organization account.

Accounts associated with only **one** organization will not need to select and instead be directly logged in.

B.2.1 - Users, Moderators and Org_Admins within Organizations

Within the app, on the organization level we primarily use the three roles: **user**, **moderator**, and **org_admin** to define what the app displays and allows.

This “given_role” for the organization, basically defines

- What pages are they allowed to access?
- What operations are they allowed to do from the front-end.

Almost every action and effect for **all three** roles is isolated on an organization level, so that any operations that could potentially occur in one organization, ever affect data relating to a different one. The **sole exception to this** is where an account's phone number or name changes.

Changing an account's name or phone number will be reflected inside all organizations.

B.2.2 - Users and Moderators within Teams

On a team level, we use the two roles: **user** and **moderator**, and this is used to represent the "players" and "coaches" within each "team" within organizations.

As *org_admins* likely will not be that involved with each individual player, it was reasonable to delegate the creation and distribution of most user accounts to the coaches that are closely tied to sellers.

This separation allows us also to control **which teams each moderator can edit** and which ones they cannot.

B.3 - The User Role

The users in the app can be anybody from players of a team, parents of the player, coaches, family of the player and/or someone higher up in the organization hierarchy.

Any account with the **user** role, is granted access to logging into the app.

This will be the assigned role to any user who does not belong to a higher administrative level within that organization.

The only requirements for this role to work properly is a **name** and a **phone number**. Which will be unique to the account.

Some **user** level accounts would likely belong to children of a very young age, in such cases; their parents' devices / numbers will likely be used when they travel around selling products.

B.3.1 - User permissions in the app

For the **user** role, they will only be able to login and view details relevant for that one specific user. Users are only able to see the statistics of *their own sales*, as well as the ability to conduct a sale of Mangrove trees to paying customers using Vipps as the payment method.

They will NOT be able to access any **Protected Routes** (paths within the web-app that are only accessible by moderators and above), or any CRUD operations besides Reading their own statistics and sales performance.

B.3.2 - User CRUD permissions

The User role has **only READ access** to the orders table in the backend only if their X-Hasura-User-Id equals their existing UUID key on their user.

In order to retrieve user sales and the statistics of their sales to display on the home-page upon logging in.

When conducting a sale, **Create** and **Update** calls are made to the Orders table in the database, however this is not a permission granted to users, but handled by the vipps-backend through a REST API. (A server that can only perform certain tasks when their endpoints are called)

The **user** role is only allowed access to the properties on the **orders** table:
Team_id, timestamp, sold_amount, earnings, status, order_id and user_id

B.4 - The Moderator Role

Moderators are accounts assigned to represent coaches or individuals responsible for the creation and management of most of the bottom-level user accounts.

Moderators have access to see and edit information about **users within the teams they moderate**, and review the sales statistics of the teams they are associated with.

Accounts must be set to the **moderator** role by a residing **org_admin** within the organization after at least one **team** has been created.

B.4.1 - Moderator permissions in the app

The moment an account has been elevated to be a **moderator** for ANY team in the organization, they are considered to have the **moderator** role within the organization. Moderators have the access to view their own personal statistics similarly to **users**, as well as conduct sales, but also have the ability to **add, edit and remove users** that are part of **their team** from their organization and team.

This does **NOT** remove all data on the user, as users could potentially be part of multiple organizations, but it does remove their connection to the organization they were removed from.

Moderators will upon logging in be redirected to their own “dashboard” instead of the statistics page users see. This will allow them to view the **statistics** of the teams they manage within that organization

B.4.2 - Moderator CRUD permissions

Moderators have access to create, modify, and delete **users only within the teams they manage** and as an effect of that, creating accounts that can access the app.

Any users created by a moderator will be automatically assigned to the **organization the moderator was logged in to upon account creation**.

If the phone number is already registered in the database in another organization, the act of registering a new user will simply add the user to the moderator's current organization in the **organization_roles** table.

The preconditions for adding users is that the moderator is logged in to an organization (**X-Hasura-Org-Id**), that they are adding it to a **team they manage** and that their role has the access to the **moderator (X-Hasura-Role)**

Moderators will be able to read data on **all users within their team (team_members)**, and view the performance and statistics of each individual member of the team.

Moderators also have **read** access to the teams tables, in order to list out multiple teams given that they are associated with more than one team at a time.

Moderators can only **update** and **delete** data about **the users they manage**. Meaning that the users they manage has to meet the following criteria for update and deletion:

- The user is not a **moderator or higher** within the same organization
- The user is part of the team where the moderator has the team_role '**moderator**'

The Moderator does NOT have access to **delete** an account from the database. Only the connection between the user and that specific **team**. (NOT from the Organization)

B.5 - The Org_Admin Role

Org_admin, short for Organization administrator, is the highest level of management **within an organization** which is meant to represent individuals who are in full control of the app's use for their organization.

Their main purpose will be to **create and manage teams and moderators for teams**, as well as **removing account connections to their organization**.

B.5.1 - Org_Admin permissions in the app

Unlike **users** and **moderators**, org_admins will be established along with their organizations when the **app_owners** have created both.

There can be multiple org_admins within each organization, and their accounts CAN be used as other lower hierarchy roles in other organizations should they have the need to be part of multiple.

Org_admins will basically have every permission there is within their organization, being able to view **total sales and data for the entire organization as a whole**, and upon logging in be shown a dashboard where they will be able to **manage teams and users**.

Within the app, org_admins can:

- **Add** and **edit** and **remove** accounts and teams from the organization
- **Manage moderators** within each **team** and within the **organization** in the cases where a moderator is no longer a coach for certain teams or unaffiliated with the organization.

B.5.2 - Org_Admin CRUD permissions

Create: Org_Admin needs to, as mentioned above, add users, moderators and teams, this will naturally include the inserting of a users organization_role and team_members role to a team.

In order for the Org_Admin to add a user to a team as a Moderator there needs to be a Team created beforehand.

Read: An Org_Admin needs to have the basic read access to all tables except for **customers** as they do not have an involvement with the customers data directly.

Org_Admins will, in the same way as creating, be able to **update the same data they have created for updated or falsely inserted information**. They can however not edit the unique values such as ID's.

For **deletion** of data an Org_admin is able to remove the role set for a User in an organization and their roles in the team_members table if they are removed from the organization or a particular team. In this case could be a forced removal of a Moderator.

B.6 - The App_Owner Role

App_owners are strictly reserved for the company and individuals within that company that owns and maintains the app we develop. They will be the ones in charge of how the app is distributed, what organizations are given access to use the platform, and also control deletion of potentially identifying information.

B.6.1 - App_Owner permissions in the app

The **app_owner** will have access to manage all of the organization's statistics, teams, users and customer information, however, there will be no implementations in the app for app_owners to create teams and users within organizations.

B.6.2 - App_owner CRUD permissions

As the owners of the application, they will be given full rights to all **CRUD** operations relating to the database.

The permission granted here is based on a level of trust that the owners of the app do not misuse their permissions.

Note that the application does not create the opportunity for App_owners to create regular users or teams, but rather, **Creating Organizations** and **org_admins** for those respective organizations.

However, it will give **app_owners** the ability to view all user accounts, their organizations, and teams, and completely remove them from the system.

Note that removing a user account, does **NOT** remove any transactions they have made from the **orders** table in the database, as there is no identifying information about the selling users in that table, on top of the fact that transaction information should be kept, even if the user conducting the sale requests their data deleted.

B.7 - GDPR Considerations

In accordance with the General Data Protection Regulation (GDPR) within the EEA (European Economic Area) of which Norway is a part of. We have to adhere to certain requirements regarding the storage of information that could potentially be considered to be “identifying information” within our data.

To comply with this, we are required to give anyone who has their data stored in our database the option to request deletion of anything that could potentially be used to identify them. (*“Do We Always Have to Delete Personal Data If a Person Asks?”*, European Commission)

We have established permissions within the app, such that anything that could identify can be deleted.

While **users** and **customers** (a group that is not defined in our permissions, as they are not directly interacting with any of our systems) cannot directly delete information themselves, should they want their data deleted from the database, they will have to request such deletion from accounts higher up in the hierarchy.

Moderators can remove any association between the **user** and the team, but **not** remove an account’s association with the **organization**, as users could potentially also be part of teams the moderator does not manage.

Org_admins can remove any association between the **user** and the **organization**. Meaning they can freely remove their connection to any teams within the organization and sever that connection entirely. However, org_admins can **not** delete user accounts entirely, as accounts could potentially be affiliated with other organizations not within the org_admin’s control.

Because of these cases for moderators and org_admins, we can only delegate the complete deletion of an account in the database and the customers within, to the **owners of the application**.

App_owners, who in our case will be our company’s client, Waterdrop, will have every permission and option necessary to completely delete user information.

In order for any user account and any customer, should they want to have any identifying information deleted, they will have to request so towards the owners of the app. (*«How Should Requests from Individuals Exercising Their Data Protection Rights Be Dealt With?»*, European Commission)

How users and customers approach requesting their stored information, and/or deletion, must be determined by the owners of the app, as well as informed about to any users of the app, by the developers (us).

B. 8 - Sources / References:

European Commission. «Do We Always Have to Delete Personal Data If a Person Asks?» Text. European Commission. Opened 13. april 2021.

https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-or-organisations/dealing-citizens/do-we-always-have-delete-personal-data-if-person-asks_en.

———. «How Should Requests from Individuals Exercising Their Data Protection Rights Be Dealt With?» Text. European Commission . Opened 13. april 2021.

https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-or-organisations/dealing-citizens/how-should-requests-individuals-exercising-their-data-protection-rights-be-dealt_en.

Attachment C: Information letter for test participants



Informasjonsbrev - Bachelorprosjekt v/ Bouvet ASA

Hei,

Vi er en bachelorgruppe fra Høyskolen Kristiania som studerer Informasjonsteknologi - Frontend og mobil-utvikling. Vi gjør denne bacheloroppgaven i regi av Bouvet for Thor Heyerdahl Climate Park. Prosjektet går ut på å utvikle en applikasjon for selgere, med hensikt å assistere med salg av produkter for både unge og voksne selgere på døren.

Bacheloren går ut på å utvikle en salgsplattform for å selge mangrover som plantes i Myanmar i Asia som da kan brukes av organisasjoner eller idrettslag som gjør en innsamling via dør til dør salg.

Vi trenger deltagere til å teste prototypen som vi har laget. Dette er en app som du som selger går rundt med for å gjøre et salg når du møter kunder. Testingen vil bli gjort digitalt via plattformen som vi deler med deg, og vi vil også spørre eventuelle spørsmål i etterkant om hvordan det gikk. Webkamera og deling av skjerm blir nødvendig og **ingenting tas opp**.

Data vi samler inn under brukertestene anonymiseres og det er ingenting du som deltaker trenger å forholde deg til etter testene er over.

Det er ikke obligatorisk å delta i dette studiet, og hvis du velger å delta, kan du trekke deg når som helst uten konsekvenser. Alle personlige detaljer vil bli holdt anonymt.

Har du bekymringer eller klager angående de etiske elementene i dette prosjektet, ta kontakt med hamtom18@student.kristiania.no eller Tommy Hamarsnes på telefon +47 41 84 22 40 med tittel "Bachelorprosjekt Bouvet".

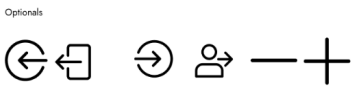
Vår kontaktperson:

Mail: hamtom18@student.kristiania.no

Tlf: +47 418 42 240

Attachment D: UI Kit

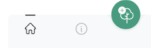
Master Icons Sheet



App Bars: Bottom

Colors Used

- Primary
- Black
- White
- Grey

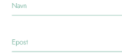


Active Home Tab



Active Info Tab

Text Fields / Text Fields Outlined



Name



Epon

Typography Sheet

Login Page

Login/Jost/Light/3rem rem (48 px)

Button Text/Jost/Medium/1.5 rem (24 px)

Form/Jost/Light/1.12 rem (18 px)

Home Page

Title Digit/Jost/Light/2.25 rem (36 px)

this is the numbers for number of sold trees for example

H5/Jost/SemiBold/2 rem (32 px)

Title/Jost/Regular/1.25 rem (20 px)

Nav Drawer

Nav Drawer Item/Jost/Regular/1.12 rem (18 px)

Selling process

Total/Jost/SemiBold/ 2.25 rem (36px)

Total Number/Jost/SemiBold/ 2.25 rem (36px)

Subtitle/Jost/SemiBold/ 1.9 rem (30px)

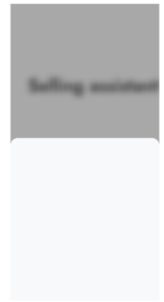
Backdrop

Colors Used

- White
- Black



Active back layer



Active front layer

Color Sheet

Primary

#489B80

Black

#060F16

White

#F8F9FB

Background

#ECECEC

Vipps

#FD5B24

Grey Inactive

#B2B2B2

Buttons

Large

Optional small

Optional

Medium

Optional medium

Inactive

Small

Optional small



Attachment E: Material Design rules

E.1 - Layout

Dimensions

Dimensions refer to the width and height of component elements.

Some components, such as an app bar or list, only outline the height of an element. The heights of these elements should align **to the 8dp grid**. Their widths are not specified because it's responsive to a viewport width.

1. Status bar height: 24dp
2. App bar height: 56dp
3. List item height: 88dp

Containers (make sure every icon and image have containers)

A container is a shape used to represent an enclosed area. Containers can hold UI elements such as images, icons, or surfaces. Containers can be rigid and restrict the size or cropping of elements within them. Alternatively, they can be flexible and grow to support the size of the content they hold.

Don't use a container on fluid components if it's too narrow to display elements and padding at smaller widths.

Use caution when spanning a fluid component across several columns in a wide screen. Certain components, like buttons, might be overly emphasized on larger screen widths.

To maintain consistency in your layout, use a consistent aspect ratio on elements like images, surfaces, and screen size.

To display images responsively, define how an image will be cropped at different breakpoint ranges. At different breakpoint ranges cropping can:

- Maintain one fixed ratio
- Adapt to different ratios
- Fix image heights

Touch targets apply to any device that receives both touch and non-touch input. To balance information density and usability, touch targets should be at least 48 x 48 dp with at least 8dp of space between targets.

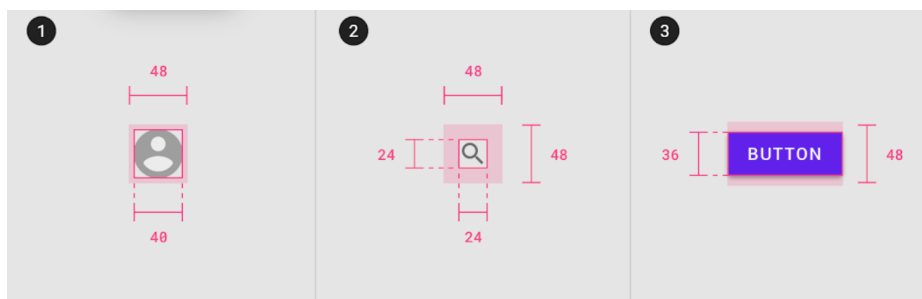
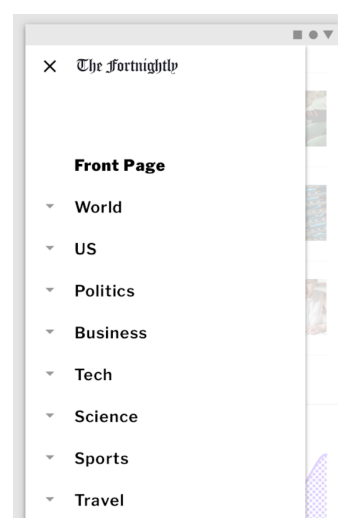


Illustration: Containers specs

E.2 - Color

Modal sheets

Use contrasting colors on surfaces that appear on-screen temporarily, such as navigation drawers and bottom sheets. Usually these surfaces are white, but you can use your app's primary or secondary color.



This app uses its primary color (white) for its modal navigation drawer, creating the maximum contrast between the dark typography and the navigation. A white scrim is used to make content behind it less noticeable, as the navigation drawer is the same color as the background.

Color usage

To bring attention to important events, use stronger color contrasts between elements.

By limiting the use of color in your app, the areas that do receive color gain more attention, such as text, images, and individual elements like buttons.

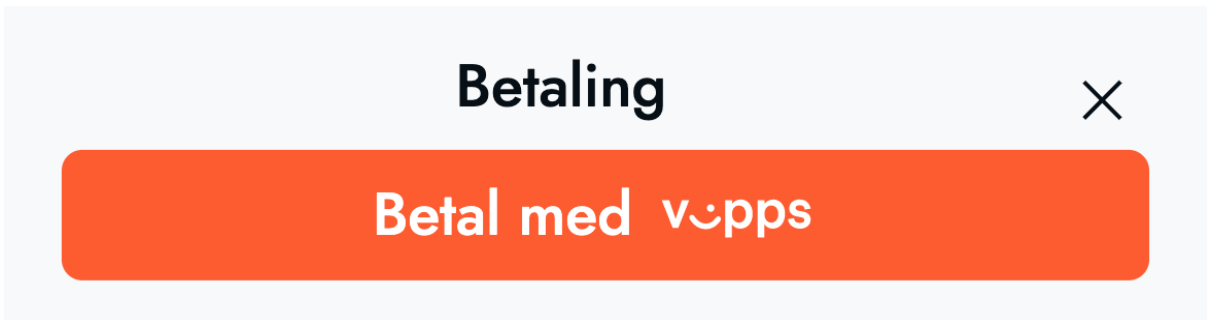
Because the content of this product is multicolored, a black floating action button contrasts greatly with the bright colors, making it more visible. To bring attention to important events, use stronger color contrasts between elements.

Progress indicators

Progress indicators are a subtle but powerful place to incorporate brand color, as they tie the function of the app to the brand.

Color can provide information about the state of an app, its components, and elements. This includes:

- **Current state** of an element or component, such as whether a button is



enabled or disabled

- **Changes in state** to an app, component, or element

Color should be noticeable when indicating state changes, as subtle differences in color may be missed. It's best to indicate a change of state in more than one way, such as by displaying an icon or moving the location of an element.

Using text opacity

Instead of using grey text and icons on top of colored backgrounds, create better contrast by displaying white or black text with reduced opacity. For example, black text displayed at 75% opacity on a green background gives the text an appearance of black, with a hint of green.

Dark text on light backgrounds

Dark text on light backgrounds (shown here as #000000 on #FFFFFF) applies the following opacity levels:

- High-emphasis text has an opacity of 87%
- Medium-emphasis text and hint text have opacities of 60%
- Disabled text has an opacity of 38%

Helper text gives context about a field's input, such as how the input will be used. It can adopt brand colors, but should be legible as determined by WCAG standards. For example, helper text on light backgrounds could apply the following opacity levels and default hexes:

- High emphasis helper: This text uses a hex value #000000 at 87% opacity
- Default color helper text: This text uses a hex value of #000000 at 60% opacity
- Default error helper text: This text uses a hex value of #B00020 at 100% opacity

E.3 - Components

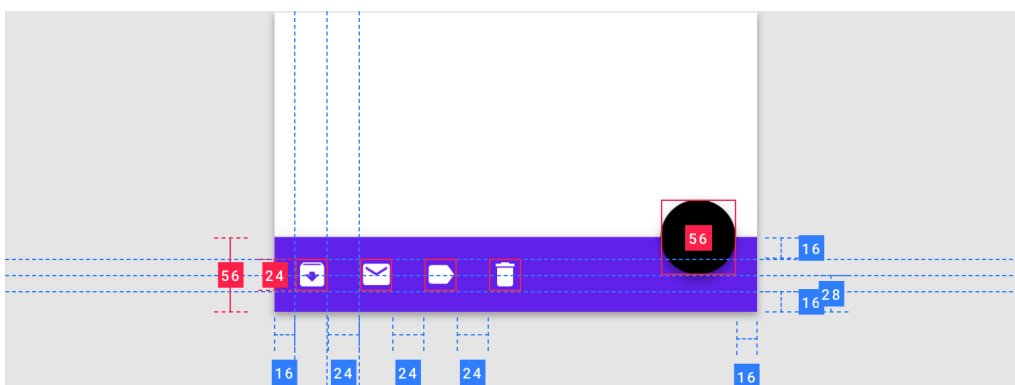
App bars: bottom

Bottom app bars provide access to a bottom navigation drawer and up to four actions, including the floating action button.

1. **Inset:** The FAB is at the same elevation as the bottom app bar, and the bar shape transforms to let the FAB dock in a notch carved into the bottom app bar.

FAB End

Minimum 2, maximum of 4 actions aligned to opposite edge of the FAB



Bottom navigation

Bottom navigation can be temporarily covered by dialogs, bottom sheets, navigation drawers, the on-screen keyboard, or other elements needed to complete a flow. They should not be permanently obstructed on any screen.

On mobile (in landscape mode) or tablet, bottom navigation destinations can retain the same spacing used in portrait mode, rather than being equally distributed across the bottom app bar.

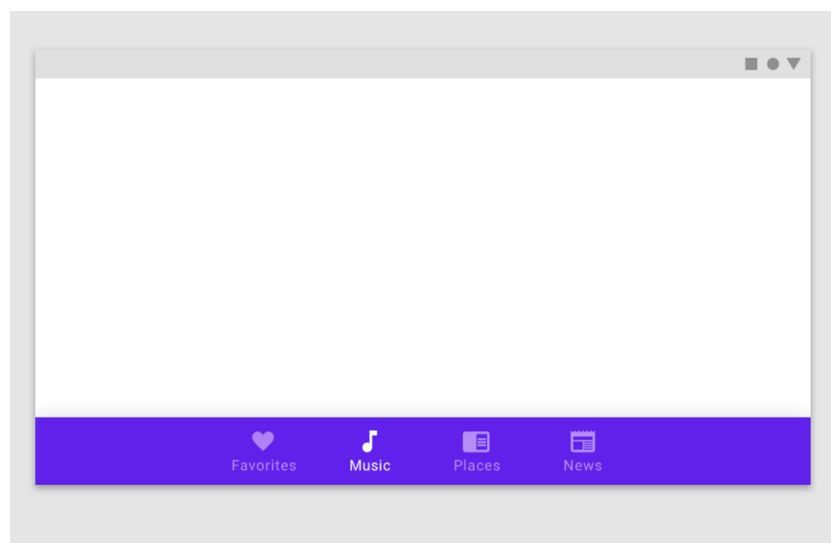


Illustration: Bottom navigation

Landscape

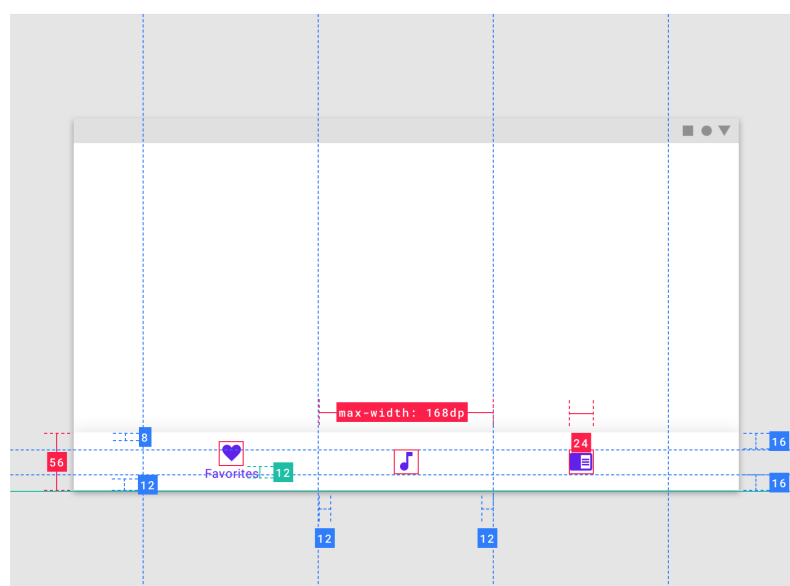
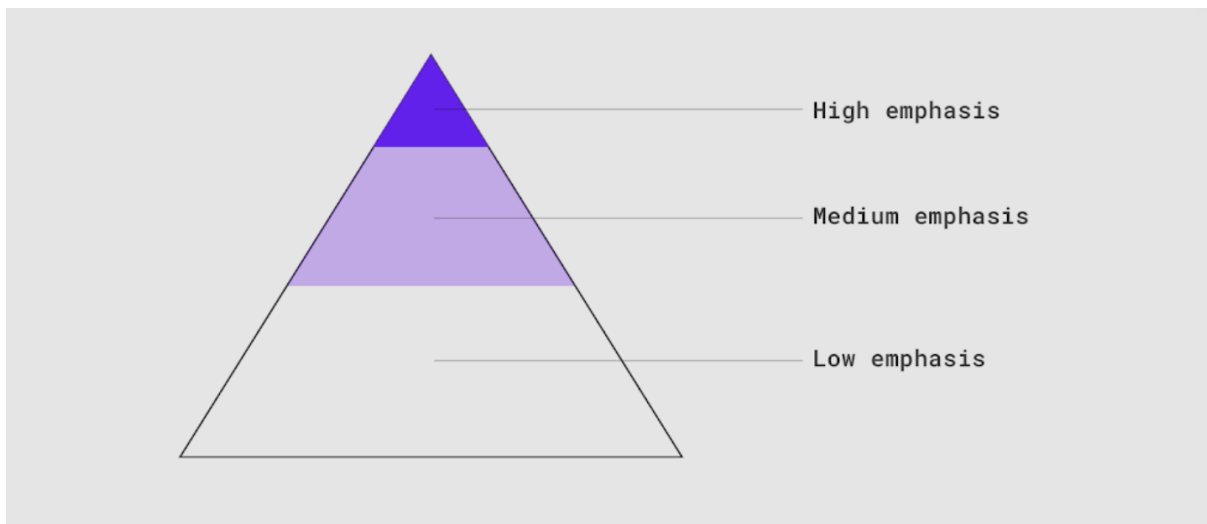


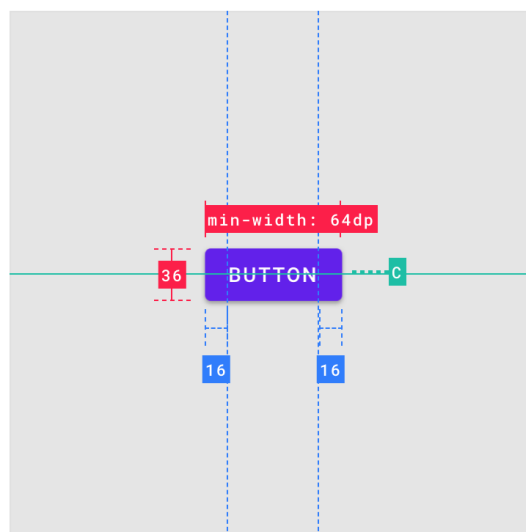
Illustration: Bottom navigation specs

Buttons

An app can show more than one button in a layout at a time, so a high-emphasis button can be accompanied by medium- and low-emphasis buttons that perform less important actions. When using multiple buttons, ensure the available state of one button doesn't look like the disabled state of another.

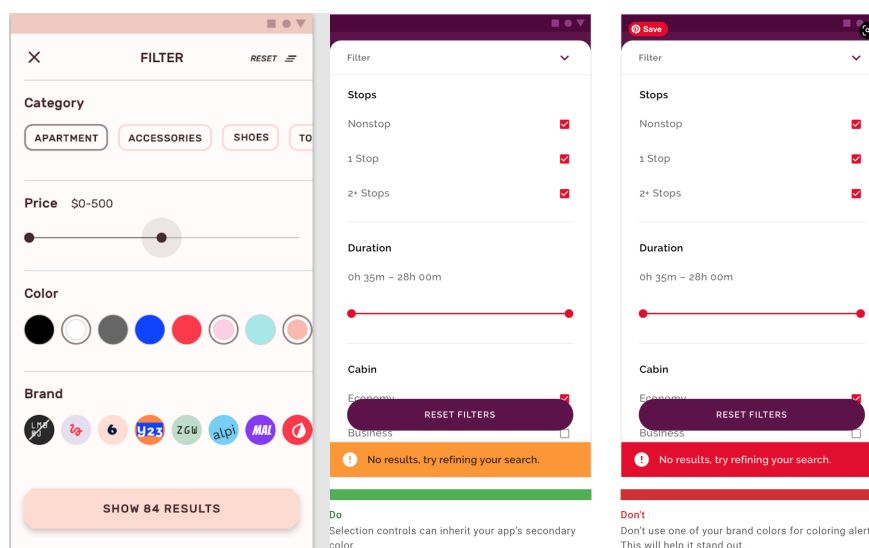


Contained button



- The baseline color for contained, text and outlined buttons is your **primary color**.
- The baseline color for floating action buttons and extended floating action buttons is your **secondary color**.
- The baseline color for selection controls is your **secondary color**.

Buttons and selection controls can be emphasized with your primary or secondary colors.



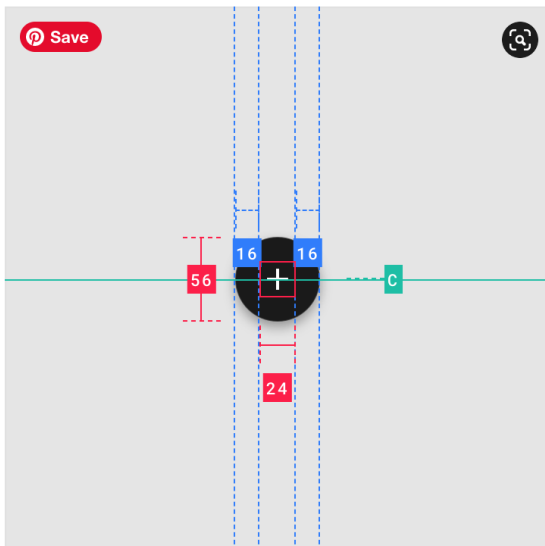
Floating action button (FAB)

A floating action button (FAB) performs the primary, or most common, action on a screen. It appears in front of all screen content, typically as a circular shape with an icon in its center. FABs come in three types: regular, mini, and extended.

The FAB is typically displayed in a circular container. An app's color scheme determines its color fill, which should contrast with the area behind the FAB. A mini FAB should be used on smaller screens. When a screen width is 460dp or less, the container of a default FAB (56dp) should transform into the mini size (40dp).

Use color to create contrast between the FAB and surrounding elements, such as the app bar. Your secondary color is the baseline color for use on the FAB. If your canvas uses many colors, your FAB can use monochromatic coloring instead, to stand out from the content.

Regular



Mini FAB

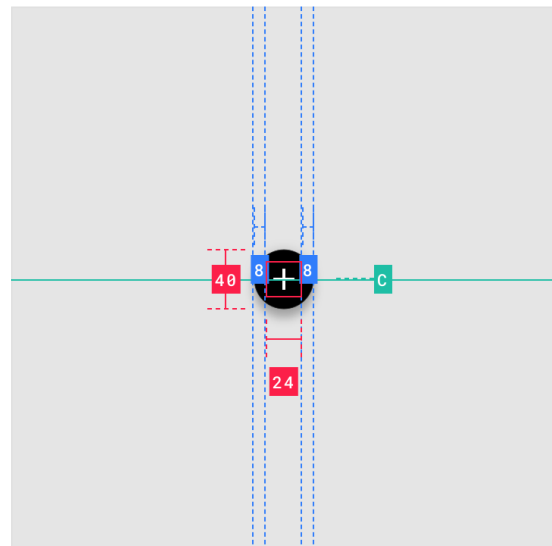


Illustration: FAB types

Bottom sheets (modal bottom turns into modal centered on desktop and tablet most likely)

Modal bottom sheets present a set of choices while blocking interaction with the rest of the screen. They are an alternative to inline menus and simple dialogs on mobile, providing additional room for content, iconography, and actions.

Modal bottom sheets are used in mobile apps only.

Modal bottom sheets have a **default elevation of 16dp**. This elevation allows them to appear over most UI elements and allows them to be pulled up in front of the entire UI to display more options.

A modal bottom sheet causes all content and UI elements behind it to display a scrim, which indicates that they will not respond to user interaction. Tapping the scrim dismisses both the modal bottom sheet and scrim from view.

Modal bottom sheets are most effective on small screens.

On larger screens, use menus or dialogs to create clear visual connections to the triggering UI element.

Text fields (Form)

Text fields come in two types:

- Filled text fields
- Outlined text fields

Both types of text fields use a container to provide a clear affordance for interaction, making the fields discoverable in layouts.

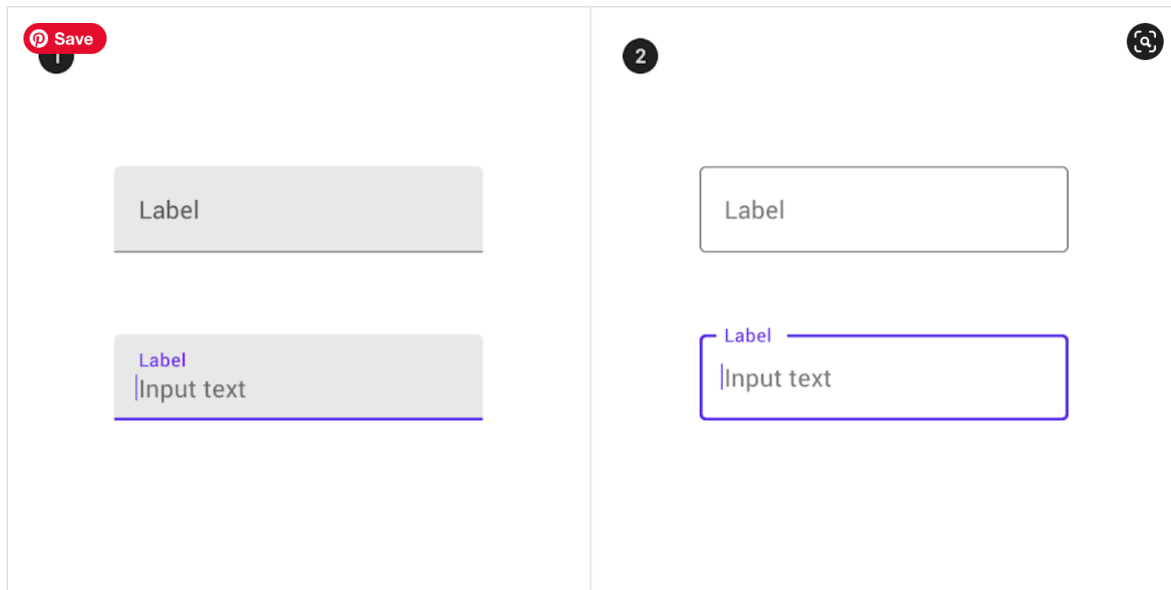


Illustration: Text fields types

To indicate that a field is required, display an asterisk (*) next to the label text and mention near the form that asterisks indicate required fields.

Assistive elements provide additional detail about text entered into text fields.

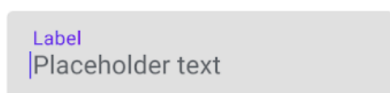
Helper text conveys additional guidance about the input field, such as how it will be used. It should only take up a single line, being persistently visible or visible only on focus.

When text input isn't accepted, an error message can display instructions on how to fix it. Error messages are displayed below the input line, replacing helper text until fixed.

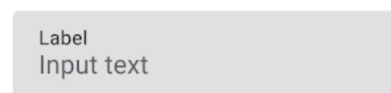
For text fields that validate their content (such as passwords), replace helper text with error text when applicable.

Filled text fields will display the following states in our app: activated, focused, error.

Focused



Activated



Error

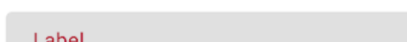


Illustration: Text fields states

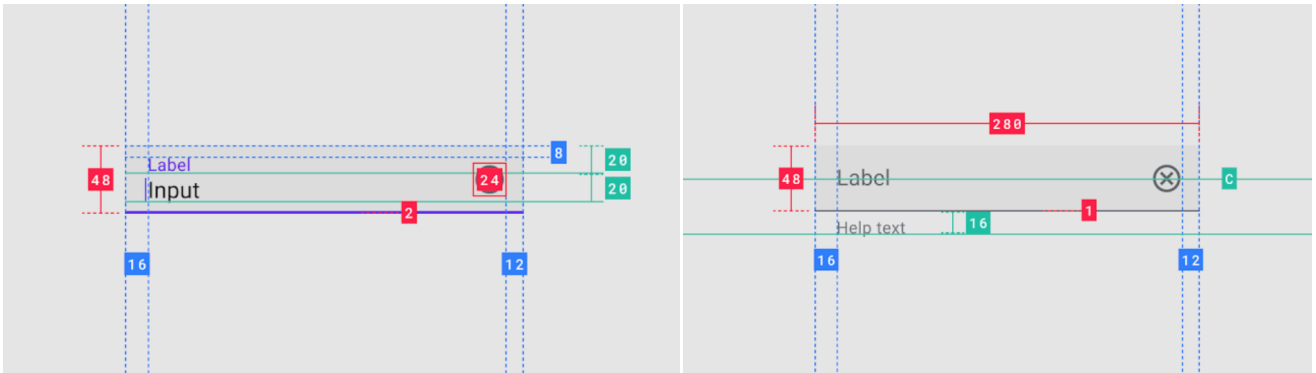


Illustration: Specs on text fields

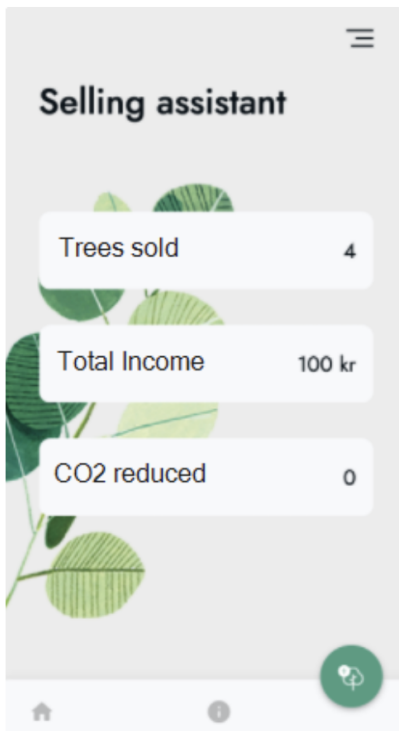
Sources / References:

Material Design <https://material.io/>

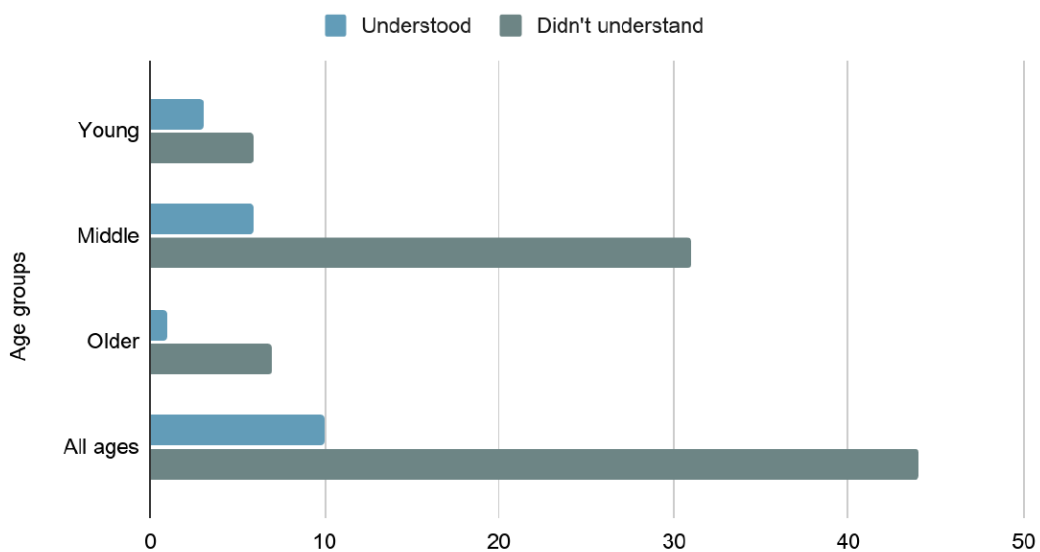
Attachment F: Survey Sections

Fab button question and results

This is an application meant to be used by a seller. Put yourself in the shoes of someone who is going to "sell trees to others" with the app. What do you expect the Floating Action Button in the bottom right to do?



Respondents understanding purpose of presented action button



Attachment G: Group Contract

Group Contract

Bachelor Group at Bouvet ASA - Bachelor Semester 2021

Ingrid-Alice Bløtekjær, Krister Pongos Emanuelsen, Marcin Janecki, Ole Algoritme, Tommy Hamarsnes

1. Goals / Visions / Ambition Levels

Our goal is to execute and complete our bachelor project at Bouvet ASA, and deliver a well performing and fledged out Bachelor project within the spring semester of 2021.

We aim for the best possible result we can manage as a group.

The bachelor is to be performed with a large focus on the development and documentation process, while always making informed decisions based on established research and literature, ensuring we not only deliver a product useful to our client but also a high-quality project fit for our bachelor.

Our primary focus is that every member of the bachelor group feels they get a chance to display their own talents and interests when it comes to project contributions, and that everyone is comfortable with their own participation and the group's dynamics.

We envision a lot of continuous interaction with our client to ensure the project stays closely connected to the client and so they will find that our contributions during this semester ends up being useful to them even after our time working with them ends.

2. Roles - Scrum

Throughout this semester, the team has decided on using scrum as an agile approach towards continuous delivery of our product, with basis on how we are previously familiar with the process and feel it will help us properly maintain progress in our project.

Scrum Master - Krister Pongos Emanuelsen

Project Manager - Ingrid-Alice Bløtekjær

Scrum Team - Marcin Janecki, Ole Algoritme, Tommy Hamarsnes

3. Procedures

Communication within the group: All group members ensure that they have exchanged mobile phone numbers, e-mail addresses, and have each other added on at least one common social media platform to ensure proper communication can be upheld. This ensures that we always have at least more than one way of contacting each member should someone be unexpectedly unpresent.

Planned Absence: All absence that does not arise as cause of emergencies or unexpected reasons must be given notice in public channels as soon as possible. Should these team members have important tasks to finish this information should be relayed to other members.

Unplanned Absence: In case of emergencies and other unexpected occurrences we cannot expect members to be able to give notice, but the chance members should at such times try to alert the rest of the team of sudden occurrences that prevent them from participating as soon as possible, with no strict expectations to declare their reasons for absence in the present moment.

All decisions that affect the product or process in some way should be carried out through a group consensus. In cases of disagreement both sides should argue for their reasoning and the majority's approach takes hold after discussion.

Everyone needs contributions: The group expects a certain level of contribution from every member, and active participation when available for group work and meetings.

Meetings are held *daily* given it is a day reserved for work on the bachelor project. The group reaches a consensus on when the meetings are to be held and any absent members should give notice.

Plagiarism: Plagiarism is unacceptable, team members are encouraged to avoid copy-pasting content at any given time, not counting citations and sources.

Sourcing and referencing: Group members are encouraged and should encourage each other to be mindful of their sources and make sure to note down and document any references, articles and research papers they may use throughout the entire duration of the project.

Consequences upon breach of contract: In the case where group members do not contribute towards the goals described under article 1, abandons their responsibilities assigned to them under article 2, or breaches the procedures described in article 3, these individuals will not be excluded from the group, but will likely face backlash in the documentation and likely be excluded from receiving the same grade and results as the rest of the contributing members.

Upon suspicion of the point above, the individual in question will be directly approached by other team members regarding which parts of the contract have been breached and be given a warning. These warnings should include deadlines for the individual to improve their condition and re-establish working procedures.

Should the deadline be exceeded without improvement, the case should be discussed internally within the group on how to handle the situation from there, this discussion should be recorded, logged and kept for the documentation delivered at the end of the project, and should also be discussed with our internal counselor for the bachelor at Kristiania University College.

Contract Revisions: Should the contract be revised, the following procedure stands: The revision should be discussed in a project meeting at it's own separate time disclosed in the meeting invitation. The revised version of the contract is distributed to group members to consider any objections, and the final validation of the revised contract should be one-sided.

4. Interpersonal Questions

We are to respect other team member's inputs but also voice our own opinions should we have any to share.

Silence during conversation and discussion equates to agreement.

Problems internally within the group are to be discussed and solved by the group with a consensus, given it can be done. Praise and constructive criticism is accepted at any time such as meetings or anytime else within the project. Regularly a feedback round is done where each member shortly states their own work, teamwork and social experience regarding the group.

Signatures:

Place: Oslo	Date: 14.01.2021	Signature: 
Place: Oslo	Date: 14.01.2021	Signature: 
Place: Oslo	Date: 14.01.2021	Signature: 
Place: Oslo	Date: 14.01.2020	Signature: 
Place: Oslo	Date: 14.01.2020	Signature: 

Attachment H: Initial App Flowchart

