Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Mona Mousa
Christel Ossletten

# Developing a Booking System for Voll gård

Bachelor's thesis in Dataingeniør
Supervisor: Elise Klæbo Vonstad
May 2023

NTNU
Norwegian University of
Science and Technology

Mona Mousa
Christel Ossletten

# Developing a Booking System for Voll gård

Bachelor's thesis in Dataingeniør
Supervisor: Elise Klæbo Vonstad
May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

# Sammendrag

Hensikten med gjennomføringen av dette prosjektet var å utvikle et web-basert bookingsystem for Voll gård, med det formål å tilby besøkende en brukervennlig, sikker og tilgjengengelig plattform med effektiv bookinghåndtering. En effektivisering av dagens system ville også komme oppdragsgiver til gode, da tidsforbruk, og derav pengebruk, redusreres betraktelig. I tillegg gir systemet en full oversikt over alle bookinger.

Webapplikasjonen ble designet i henhold til ønsker og krav ytret av oppdragsgiver i starten av semesteret. Disse kravene var knyttet til de mest essensielle funksjonene når det kommer til å utvikle og anvende et bookingsystem. I tillegg var de ikke-funksjonelle kravene avgjørende for å sikre kvaliteten og evnen til systemet.

Teknologier som React, Firebase og Redux ble brukt for å utvikle webapplikasjonen. Ved å følge prinsippene om universelt design og retningslinjene i WCAG 2.1, sikret systemet tilgjengelighet ved å bruke ARIA-attributter og høy nok fargekontrast, samt sikkerhet ved å sjekke input-data mot injeksjonsangrep. Personvern ble prioritert ivaretatt ved å anvende GDPR-prinsipper.

Konklusjonen trekker fram hvilke deler av prosjektet som var vellykket, blant annet å ha levert en brukervennlig, sikker og tilgjengelig webapplikasjon. I tillegg blir ikke-implementerte funksjoner presentert sammen med tanker om videre arbeid. Dette inkluderer blant annet at designet må videreutvikles til å være responsivt. Den smidige utviklingsmetodikken, Kanban, ble brukt gjennom prosjektet og tilrettela i stor grad for godt samarbeid.

I det hele og store demonstrerer prosjektet vellykket implementering av moderne teknologier, universelle designprinsipper og retningslinjer fra WCAG 2.1. Dette for å skape et brukervennlig, pålitelig og sikkert system for en effektiv bookingprosess.

# Abstract

The purpose of this thesis was to develop a web-based booking system for Voll gård, with the aim to provide a user-friendly, secure, and accessible platform were bookings are managed efficiently. The system keeps a comprehensive overview of all bookings, and benefit the client as the consumption of time, and therefore money, is reduced.

The web application was designed according to the list of requirements addressed by the client at the beginning of the semester. The requirements were related to the most essential functions when developing and using a booking system. This in addition to non-functional requirements, which was crucial to ensure the quality and ability of the system.

React, Firebase, and Redux technologies have been used to develop the web application. By following the universal design principles and WCAG 2.1 guidelines, the developed system ensures accessibility with features such as ARIA attributes and color contrasts, and security with features such as checking input data against injection attacks. Privacy was prioritized by applying GDPR principles.

The conclusion emphasizes the successful results achieved, such as delivering a user-friendly, secure, and accessible web application. Limitations such as a responsive design and some other functionalities that should have been met, are presented as further work. The Agile development methodology, Kanban, was used through the project and facilitated collaboration and management tasks.

Overall, this project demonstrates the successful implementation of modern technologies, universal design principles, and WCAG 2.1 guidelines to create a user-friendly, reliable, secure solution for an efficient booking process.

# Preface

The following thesis was written as the final project for a Bachelor of Engineering in Computer Science at the Norwegian University of Science and Technology during the spring of 2023. This thesis is the outcome of a system development project titled ”Developing a Booking System for Voll gård”.

The assignment was given by Voll gård, which, among other things, offers farm experiences for the whole family and educational visits about Norwegian livestock and agriculture throughout the seasons of the year. The farm's location near the center of Trondheim makes it very accessible and attractive to people of all ages. A booking system has been wanted for a long time, and as one of the students is employed by the company and has seen the need, the task was formulated in late autumn 2022.

The work carried out this semester has led to essential experiences in teamwork, as well as planning and carrying out a development project. These experiences are forever highly valued and will undoubtedly be useful when entering the workforce.

We want to express our gratitude to Voll gård for the assignment, and for valuable information about the farm and its services.

We would also like to thank those of you who have participated in user testing. We appreciate your time and have greatly benefited from your feedback.

Finally, we want to thank our supervisor through the project, Elise Klæbo Vonstad from NTNU.

Mona Mousa

Mona Mousa

[illegible]

Christel Ossletten

May 2023
Trondheim

# Assignment

This section includes the original assignment description in Norwegian, as well as changes made to the original description. The criteria for the project are fully described in the Vision Document and Requirements Document, respectively in appendices B and C.

## Description

**Arbeidstittel:**
Bookingsystem for besøkende ved Voll gård.

**Hensikten med oppgaven:**
Stiftelsen Voll gård er en ideell og samfunnsnyttig stiftelse bestående av et styre på fem medlemmer som hver representerer følgende organisasjoner:

- Sør-Trøndelag Landbruksselskap
- Trøndelag Bondelag
- Sør-Trøndelag Bonde- og Småbrukarlag
- 4H Trøndelag
- Trondheim kommune

Voll gård er en besøks- og 4H-gård som ligger nær sentrum av Trondheim. Gården har de vanligste husdyrslagene som fantes på gårder før i tiden og fram til i dag, i tillegg til å huse både Kompetansesenter for urban dyrking, Strinda og Jekken frivilligsentral, Voll gårdsbarnehage og Lerkendal bo- og aktivitetstilbud. Gården tilbyr pedagogiske opplegg til skoler og barnehager i Trondheim kommune og arrangerer barnebursdager og Åpen gård for privatpersoner.

Per dags dato blir det brukt mye arbeidstid på mailhåndtering, da booking av skolebesøk, barnehagebesøk, barnebursdager og smådyrpensjonat går gjennom tre forskjellige e-postadresser. Det blir mye fram og tilbake med innhenting av informasjon, og det kan ta lang tid før man endelig får sendt ut en bookingbekreftelse. Hensikten med oppgaven er da å utvikle et bookingssytem for besøkende for å effektivisere booking av ulike tjenester på gården.

**Kort beskrivelse av oppgaveforslag:**
Behovet er altså et effektivt bookingsystem der besøkende selv kan gå inn og booke barnebursdager, opphold på smådyrpensjonat, barnehage- og skolebesøk. Det bør også være mulig å logge seg inn for å endre eller kansellere bookinger.

## Changes

One of the criteria in the original assignment description was that visitors should be able to log in to edit or cancel their bookings. However, this was put aside after interviewing the client during the first meeting at the beginning of the semester. It was concluded not to prioritize a login solution for visitors, but rather to implement the possibility to edit or cancel bookings via a confirmation email if we had time, or as further work.

The conclusion was made based on the user group of these four services. The majority of users are kindergarten teachers and elementary school teachers who would normally have called to book a spontaneous visit, or maybe would like to copy the rest of the staff on an email thread regarding the booking. For these who are used to being able to call in on the day and book a visit, having to register a personal user can seem discouraging and cumbersome.

# Acronyms and Abbreviations

**API** Application Programming Interface

**ARIA** Accessible Rich Internet Applications

**CI/CD** Continuous Integration and Deployment

**CRPD** Convention on the Rights of Persons with Disabilities

**CSS** Cascading Style Sheets

**DBMS** Database Management System

**E.g.** Exempli gratia/for example

**GDPR** General Data Protection Regulation

**HTML** HyperText Markup Language

**HTTP** HyperText Transfer Protocol

**JSON** JavaScript Object Notation

**NoSQL** Not Only Structured Query Language

**NTNU** Norges Teknisk-Naturvitenskapelige Universitet

**OWASP** Open Web Application Security Project

**RDBMS** Relational Database Management System

**SQL** Structured Query Language

**SQLIAs** SQL Injection Attacks

**VS Code** Visual Studio Code

**Web** World Wide Web

**WCAG** Web Content Accessibility Guidelines

**XML** Extensible Markup Language

**XSS** Cross-Site Scripting Attack

# Table of Contents

**List of Figures** **ix**

**List of Tables** **x**

**1 Introduction** **1**

1.1 Context . . . . . 1

1.1.1 Current Solution . . . . . 2

1.2 Problem Statement . . . . . 2

1.3 Structure of the Thesis . . . . . 3

**2 Theory** **4**

2.1 Web Applications . . . . . 4

2.2 Universal Design . . . . . 4

2.2.1 Principles of Universal Design . . . . . 4

2.2.2 Cognitive Overload . . . . . 5

2.3 Databases . . . . . 5

2.3.1 Structured Query Language . . . . . 6

2.3.2 Not Only Structured Query Language . . . . . 7

2.4 Web Security . . . . . 8

2.4.1 Broken Access Control . . . . . 8

2.4.2 SQL Injection . . . . . 9

2.4.3 Cross-Site Scripting Attack . . . . . 10

2.5 Privacy . . . . . 10

2.5.1 GDPR . . . . . 10

2.6 Qualitative Research . . . . . 11

2.6.1 Usability Testing . . . . . 11

2.7 Agile Software Development . . . . . 12

2.7.1 Scrum . . . . . 12

2.7.2 Kanban . . . . . 13

**3 Methodology** **14**

3.1 Research Methods . . . . . 14

3.1.1 Interviews . . . . . 14

3.1.2 Usability Testing . . . . . 14

3.2 Choice of Technologies . . . . . 14

3.2.1 Visual Studio Code . . . . . 14
3.2.2 React . . . . . 14
3.2.3 Node.js . . . . . 15
3.2.4 Redux . . . . . 15
3.2.5 Firebase . . . . . 15
3.2.6 Docker with GitHub Actions . . . . . 15
3.3 Version Control Tool . . . . . 16
3.3.1 GitHub . . . . . 16
3.4 Development Method and Tools for Project Planning . . . . . 16
3.4.1 Kanban Method . . . . . 16
3.4.2 Trello . . . . . 16
3.4.3 Pencil Project . . . . . 17
3.4.4 Distribution of Work and Roles . . . . . 17
3.5 Other Tools . . . . . 17
3.5.1 Microsoft Teams . . . . . 17
3.5.2 Overleaf . . . . . 17
3.5.3 Zotero . . . . . 17
3.5.4 Canva . . . . . 17

**4 Results** **18**
4.1 Scientific Results . . . . . 18
4.1.1 User Interface . . . . . 18
4.1.2 Results of User Testing . . . . . 20
4.1.3 Results of Code Testing . . . . . 21
4.2 Engineering Results . . . . . 21
4.2.1 Goal and objectives . . . . . 21
4.2.2 Functional Requirements . . . . . 22
4.2.3 Non-Functional Requirements . . . . . 24
4.3 Administrative Results . . . . . 25
4.3.1 Time Usage and Distribution of Hours . . . . . 25
4.3.2 Process . . . . . 26

**5 Discussion** **27**
5.1 Discussion of Scientific Results . . . . . 27
5.1.1 Universal Design . . . . . 27
5.1.2 Continuous Integration and Continuous Development . . . . . 28

5.1.3 Limitations of the Design . . . . . . . . . . . . . . . . . . . . . . . . . . 28
5.1.4 Limitations of User Testing . . . . . . . . . . . . . . . . . . . . . . . . . . 28
5.1.5 Limitations of Code Testing . . . . . . . . . . . . . . . . . . . . . . . . . . 28
5.2 Discussion of Engineering Results . . . . . . . . . . . . . . . . . . . . . . . . . . 29
5.2.1 Functional Requirements . . . . . . . . . . . . . . . . . . . . . . . . . . 29
5.2.2 Non-functional Requirements . . . . . . . . . . . . . . . . . . . . . . . . 29
5.3 Discussion of Administrative Results . . . . . . . . . . . . . . . . . . . . . . . . 34
5.3.1 Progress Plan . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
5.3.2 Time Usage . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
5.3.3 Development Method and Cooperation . . . . . . . . . . . . . . . . . . . . 34

**6 Conclusion** **35**
6.1 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
6.2 Further Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36

**7 Social Impact** **38**

**References** **39**

**Appendix** **42**

**A Preliminary Project Plan** **42**

**B Vision Document** **54**

**C Requirements Document** **69**

**D System Document** **98**

**E Usability Testing** **118**

**F Views** **128**

# List of Figures

1 A summary of Voll gård. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
2 Types of SQL and NoSQL databases [7]. . . . . . . . . . . . . . . . . . . . . . . . . . . 5
3 A simple illustration of vertical scaling [10]. . . . . . . . . . . . . . . . . . . . . . . . . 6
4 A simple illustration of horizontal scaling [10]. . . . . . . . . . . . . . . . . . . . . . . . 7
5 Top 10 malicious attacks [20]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
6 The life cycle of a Scrum process [32]. . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
7 An example of a standard Kanban board [35]. . . . . . . . . . . . . . . . . . . . . . . . 13
8 A cutout of the Trello board used during the project. . . . . . . . . . . . . . . . . . . 16
9 User flowchart. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
10 Admin flowchart. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
11 A pie chart representing time spent distributed across activities. . . . . . . . . . . . 25
12 A pie chart representing the time spent across categories. . . . . . . . . . . . . . . . 26
13 Colors contrast . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31

# List of Tables

1 Summary of usability testing results. . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
2 Functional requirements. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
3 Time usage and distribution. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
4 Number of implemented requirements vs number for further work. . . . . . . . . . 29

# 1 Introduction

The system developed during the project is intended to meet a client's needs. An explanation of the context behind this, as well as a representation of the client and it´s current solution, and the problem statement, is provided in this section. The section also includes a description of the thesis structure and an overview of the different sections.

## 1.1 Context

Voll gård is located near the center of Trondheim and is a visitor farm with more than 50,000 annual visitors. The farm is owned by *Sør-Trøndelag Landbruksselskap* and run by *Stiftelsen Voll gård*, which is an ideal, socially beneficial foundation without the aim of profit.

Trondheim municipality is one of five board members in the foundation and contributes with annual grants that enable free farm visits for schools in the municipality. Because of this, one of the most significant purposes of the farm is to give schools and kindergartens lessons related to Norwegian livestock and farming seasons.

The farm organizes various events, for instance, open days for the public every Sunday and birthday celebrations for children. It offers several services such as premises for rent, courses in urban cultivation, and animal boarding for small animals. Stiftelsen Voll gård is also the owner of *Kompetansesenter for urban dyrking* and three different volunteer centers, which also offer a wide range of services.

All the events mentioned above, in addition to other services, must be handled and registered in an organized and intuitive booking system for customers and employees in the foundation. This is important as there is a lot of information to keep track of and because many different people within the foundation are contacted regarding bookings.

Figure 1: A summary of Voll gård.

### 1.1.1 Current Solution

Today, bookings are handled via phone calls and several different email addresses. There are three separate email addresses for booking farm visits, birthday celebrations, and animal boarding, although the same person handles all bookings as of today. This solution remains from earlier when the bookings were distributed among different persons responsible for the respective services. Some bookings may even come through the farm´s Facebook page or via phone calls and emails to employees at other departments on the farm.

The current booking system is very time-consuming as there is a lot of communication back and forth between the customer and the client to answer all questions from both parties and obtain all the information necessary to confirm the booking. The booking details from the customer is entered into a template in Google Docs which includes the company's terms and conditions. This document is sent to the customer, who must then send an email back to confirm that the information is correct and that the terms and conditions are accepted.

When the booking is confirmed the details are entered into an overview in Google Sheets, which is connected to a Google Calendar visible for employees with access.

## 1.2 Problem Statement

The scope of the thesis includes developing a booking system as a web application for Voll Gård. It is important for customers to easily access important information and be able to book services, and for the client to see all bookings and be able to update or cancel them if necessary. Therefore, the question that should be answered through this thesis is:

*How can a web-based booking system simplify the booking process for visitors and provide a meaningful overview of daily activities for the client?*

The problem statement was defined at the beginning of the project and is based on the goals presented in the Preliminary Project Plan, attached as Appendix A.

## 1.3 Structure of the Thesis

The following subsection describes the structure of the thesis and what each chapter covers. A clickable table of contents can be found above.

**Chapter 1: Introduction**
Clarification of the background of the thesis. The client and its current solution are presented. The problem statement and goals are seen in connection with the client and its needs. The chapter also includes an overview of the structure of the thesis.

**Chapter 2: Theory**
Description and explanation of the theoretical background for the work. The purpose of the chapter is to present what knowledge the work is based on and what the reader needs to gain insight into to be able to understand and assess the work presented in the report.

**Chapter 3: Methodology**
A description of the methodologies used during the project. Presents the choice of technology, developments methodologies, and research methodologies.

**Chapter 4: Results**
Presents the project's results in the following three sub-chapters: scientific, engineering, and administrative results.

**Chapter 5: Discussion**
Presents a discussion of the results presented in the previous chapter, regarding the finished product as well as the development process. What went well and what could have been done differently is analyzed.

**Chapter 6: Conclusion**
In the conclusion, the problem statement is answered based on the discussion. The chapter also includes recommendations regarding further work.

**Chapter 7: Social Impact**
The last chapter considers the social impact of developing new technologies. Both the positive and the negative sides are discussed.

# 2 Theory

## 2.1 Web Applications

Web applications are programs that are stored on remote servers and launched on a browser interface over the internet. This gives the user the advantage of not having to manually install and download the application on their device, as well as not having to update it. Web applications can be launched on most browsers as long as the user has a device with internet access, e.g., a computer, laptop, tablet, or smartphone [1].

In 1999 the concept of web applications appeared in Java language. Java Script enabled dynamic pages with interactive elements compared to the earlier static HTML websites. The significant shift from static to dynamic pages took place in 2005 as Ajax got introduced [2].

## 2.2 Universal Design

In Article 2 of the United Nations CRPD, the definition of universal design is as follows:

*"Universal design" means designing products, environments, programs, and services to be usable by all people, to the greatest extent possible, without the need for adaption or specialized design. "Universal design" shall not exclude assistive devices for particular groups of persons with disabilities where this is needed [3].*

WCAG 2.1 was published in June 2018 and is the guideline for making websites and applications more accessible, thus contributing to the principle of universal design. The guidelines contribute to the content being accessible to people with special needs, such as impaired vision, hearing, and motor or cognitive abilities.

### 2.2.1 Principles of Universal Design

Seven universal design principles were formulated in 1997 by an interdisciplinary group at the Center of Universal Design at North Carolina State University in the USA. In Norway, there are laws and regulations about universal design, but it is also important to keep these principles in mind during the development of a product to make it suitable for as many people as possible. The seven principles are listed and described below [4].

**Principle 1: Equitable Use**
The product should be useful and available for people with diverse abilities.

**Principle 2: Flexibility in Use**
The product should cover a wide range of individual preferences and abilities.

**Principle 3: Simple and Intuitive Use**
The product should be easy to understand regardless of the user´s experience, knowledge, language skills, or concentration level.

**Principle 4: Perceptible Information**
The product should communicate the necessary information to the user efficiently, regardless of conditions related to the surroundings or the user's sensory skills.

**Principle 5: Tolerance for Error**
The product should minimize the adverse consequences of accidents or unintended actions.

**Principle 6: Low Physical Effort**
The product should be used efficiently and comfortably with minimal effort or fatigue.

**Principle 7: Size and Space for Approach and Use**
The product should have an appropriate size and space that enable access, reach, operation, and use, regardless of the user´s body size, posture, or mobility.

### 2.2.2 Cognitive Overload

An essential part of universal design is ensuring that the user does not experience cognitive overload. Cognitive load is the total amount of information your working memory can handle. When this limit is exceeded, cognitive overload occurs. This refers to the working memory receiving more information than it can handle comfortably, leading to confusion and overwhelm. Examples of things that can lead to cognitive overload are too many options and information, unnecessary actions, and overstimulation in the form of a chaotic interface [5].

## 2.3 Databases

A database is a structured collection of data typically stored in an electronic computer system. Database system, or just database, is usually a collective term for the data, the database management system (DBMS), and the applications associated with these [6].

As shown in Figure 2, databases are divided into SQL and NoSQL databases. There are also different types within these. The differences are further explained in the two following subsections.

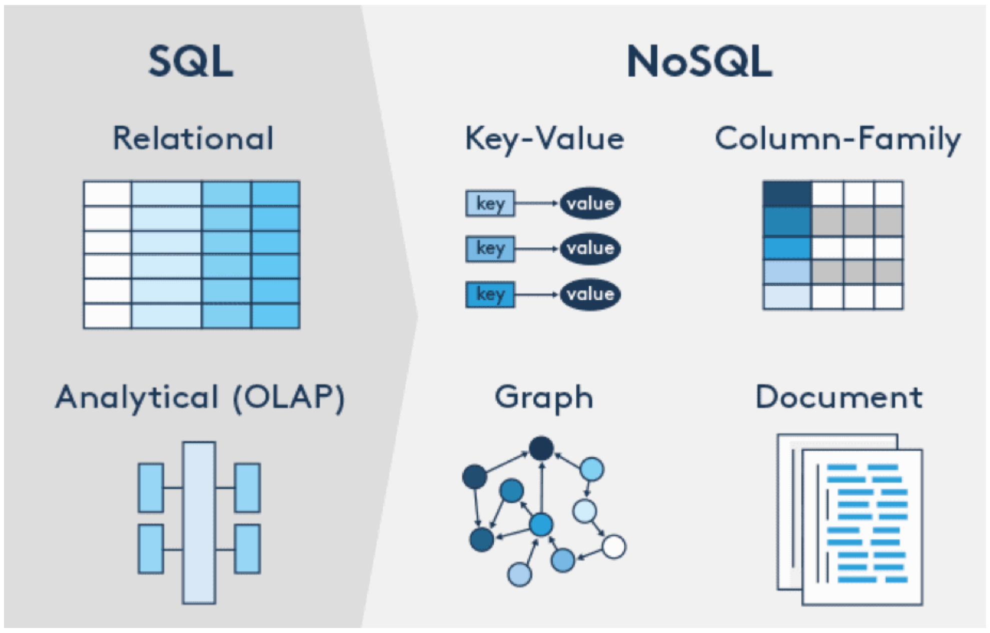


Figure 2: Types of SQL and NoSQL databases [7].

### 2.3.1 Structured Query Language

Structured Query Language (SQL) refers to a standard programming language used to create and manipulate data stored in relational databases [8]. Relational Database Management Systems (RDBMS) are the basis for SQL. In RDBMS, the databases consist of traditional tables of related data entries with rows and columns [9].

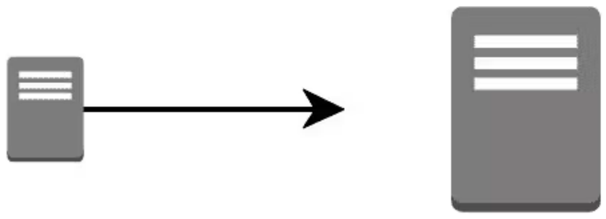


Figure 3: A simple illustration of vertical scaling [10].

SQL databases are vertically scalable, adding additional capacity and resources to one single unit, as shown in Figure 3. A benefit of vertical scaling is that the infrastructure of the database remains unchanged, as the only thing changing is the hardware specifications of the machine running it [10]. The five vital elements of SQL are listed and described below [8].

**Keywords**
Keywords are a set of words that allow the user to perform operations and manipulate the database. Examples are CREATE, SELECT, and DELETE.

**Clauses**
Clauses are built-in features that filter out and retrieve required data from the database.

**Expressions**
Expressions are formulas typically written in a query format.

**Predicates**
Predicates are keywords used to reveal relationships between expressions, resulting in true or false values.

**Queries**
Queries are statements used to request data from a database [8].

### 2.3.2 Not Only Structured Query Language

Not Only Structured Query Language (NoSQL) is a non-relational database that does not only use SQL for queries. Instead of using the traditional tables of rows and columns, the database is optimized for the specific requirements of the type of data being stored [11].

NoSQL databases are horizontally scalable, as illustrated in Figure 4. That means expanding systems by adding units. A benefit of horizontal scaling is that you are not limited to the capacity of one single unit, which makes scaling possible with less downtime [12]. Listed below are the four major and most popular NoSQL databases.

Figure 4: A simple illustration of horizontal scaling [10].

**Document Databases**
In a document database, the data is stored in documents, such as JSON and XML. The document in a document database stores information about objects in field-value pairs. These values can be of different types and structures, e.g., strings, numbers, dates, arrays, and objects [13].

This way of storing data is both fast and intuitive for developers. The method is distributed and gives the ability to scale out horizontally.

**Key-Value Databases**
The key-value database is also known as a hash table or dictionary. In this type of database, the data is stored as a collection of key-value pairs. The key is a constant and acts as a unique identifier that defines the data, while the value is a variable belonging to the set of data or a pointer to it [14]. This type of database is suitable for applications requiring frequent updates and lots of small continuous reads and writes [15].

**Wide-Column Databases**
In a wide-column database, also called a column family database, the data is stored in tables, rows, and dynamic columns. Even though the rows and columns are in the same table, the names and format of the columns can vary from row to row. The wide-column database can be seen as an expansion of the key-value database across several columns. These databases are ideal if the purpose is to distribute large sets of data across multiple rows [16].

**Graph Databases**
A graph database stores the data in nodes and edges. The main purpose of this database is to handle large data sets with complex relationships and interconnections. The nodes represent entities (e.g., people and places), while the edges represent the relationship between the entities [17].

## 2.4 Web Security

Web security is an essential aspect of web applications. Today, web security is a genuine concern related to the Internet [18]. It is also known as "Cybersecurity" [19].

Web security refers to the practice of protecting websites, web applications, and web services from unauthorized access, data theft, and other security threats. This includes safeguarding user data, preventing website defacement or destruction, and ensuring that websites and web applications function as intended without being compromised by attackers.

According to OWASP, there are ten types of malicious attacks, as shown in Figure 5. In the following subsections, two of the types of attack will be further explained. These are broken access control and injection attacks, including SQL injection and XSS attacks.

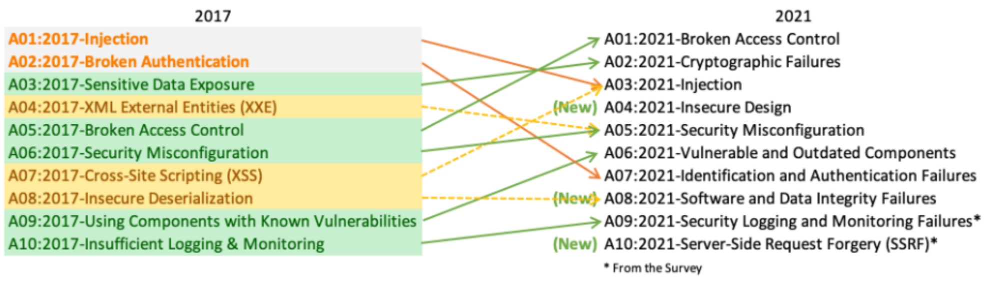

2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

Figure 5: Top 10 malicious attacks [20].

### 2.4.1 Broken Access Control

Access control enforces a policy such that users cannot act outside their intended permissions. Failures typically lead to unauthorized information disclosure, modification or destruction of all data, or performing a business function outside the user's limits [20].

According to OWASP, the common access control vulnerabilities include the following:

- Access needs to be adequately controlled or restricted, leading to potential security risks and unauthorized access to resources.
- Bypassing access control measures by making changes to the URL, modifying the application's internal state, or manipulating the HTML page. Additionally, using a tool to change API requests can be used to bypass access control.
- Permitting viewing or editing someone else's account by providing its unique identifier.
- Accessing API with missing access controls for POST, PUT, and DELETE.
- Force browsing to authenticated pages as an unauthenticated user or selecting pages as a standard user.

### 2.4.2 SQL Injection

SQL injection refers to a class of code-injection attacks in which data provided by the user is included in an SQL query in such a way that part of the user’s input is treated as SQL code. An attacker can submit SQL commands directly to the database by leveraging these vulnerabilities. These attacks are a serious threat to any web application that receives input from users and incorporate it into SQL queries to an underlying database [21].

The following example explains how hackers manipulate a SQL statement to include a malicious payload within the query of the SQL statement [22].

*This is the standard SQL statement:*

```
# Define POST variables
uname = request.POST[’username’]passwd = request.POST[’password’]# SQL
query vulnerable to SQLisql = ”SELECT id FROM users WHERE username=’” +
uname + ”’ AND password=’” + passwd+”’”
# Execute the SQL statement
```

*This is the SQL injection code:*

```
SELECT id FROM users WHERE username=’username’
AND password=’password’ OR 1=1’
```

There are various input mechanisms through which malicious SQL statements can be inserted into an application susceptible to such attacks. The following paragraphs present more details about these mechanisms as described in [21]:

**Injection through user input**
Attackers inject SQL commands by manipulating user input, typically coming from form submissions sent to the application via HTTP GET or POST requests. Web applications can access the user input contained in these requests as any other variable in the environment, making them vulnerable to SQLIAs.

**Injection through cookies**
Cookies store state information on the client’s machine so that the client’s state can be restored when they return to the application. However, since the client has control over the storage of the cookie, a malicious client could potentially tamper with its contents. If a web application uses the cookie’s contents to build SQL queries, an attacker could embed an SQL injection attack in the cookie.

**Second-order injection**
Refers to attacks where attackers manipulate input values in a system or database to indirectly trigger a SQL injection attack when the input is used later.

**Injection through server variables**
Web applications often use server variables containing HTTP, network headers, and environmental variables to log usage statistics and identify browsing trends. However, if these variables are logged to a database without proper sanitization, it can create an SQL injection vulnerability. Attackers can exploit this vulnerability by forging the values placed in HTTP and network headers and inserting an SQL injection attack directly into them. The attack is triggered when the query to log the server variable is issued to the database.

### 2.4.3 Cross-Site Scripting Attack

Cross-site scripting (XSS) is an attack technique that involves the injection of malicious code into a web page to control a user's browser. Originally abbreviated as CSS, it was renamed XSS to avoid confusion with cascading style sheets. While the attack originally involved crossing domains, today, it can occur within a single domain due to the complex nature of modern web applications. XSS has been a significant challenge for web security due to its ability to cause devastating effects [23].

Hackers can use several types of XSS attacks to exploit web vulnerabilities. Reflected XSS, stored XSS, and DOM-based XSS are the most popular attacks.

**Reflected XSS** - The hacker's payload must be included in a request sent to a web server and is then included in the HTTP response. Reflected XSS is a non-persistent form of attack, which means the attacker is responsible for sending the payload to victims and is commonly spread via social media or email [24].

**Stored XSS** - The attacker uses this approach to inject their payload into the target application. If the application does not have input validation, then the malicious code will be permanently stored, or persisted, by the application in a location like a database. The attacker's payload is served to a user's browser when they open the infected page. Victims inadvertently execute the malicious script when they view the page in their browser [24].

**DOM-based XSS** - A more advanced form of XSS attack that is only possible if the web application writes data that the user provides to the DOM. This data is then read by the application and sent to the user's browser. The attacker can inject their payload if the data is not handled correctly. The payload is stored within the DOM and only executes when data is read from the DOM [24].

## 2.5 Privacy

### 2.5.1 GDPR

GDPR stands for General Data Protection Regulation, which is a set of rules and regulations that governs how the personal data of individuals in the European Union (EU) and the United Kingdom (UK) is collected, processed, stored, and protected by organizations [25]. There are seven principles for protecting data that organizations and businesses must follow:

**Lawfulness, fairness, and transparency** - all personal data must be processed lawfully, fairly, and transparently [26].

**Purpose limitation-** data must only be collected and processed for legitimate purposes which are specifically and explicitly stated [26].

**Data minimization-** only data which is relevant for the purpose should be collected and processed [26].

**Storage limitation-** data should not be kept for any longer than is necessary [26].

**Accuracy-** reasonable steps should be taken to ensure that data remains accurate and is kept up to date [26].

**Integrity and confidentiality-** data must be kept securely, and technical and organizational measures should be put in place to protect it from hackers, etc. [26].

**Accountability-** data controllers must be able to demonstrate compliance with all the GDPR principles [26].

## 2.6 Qualitative Research

The definition of qualitative user research is as follows:

*Qualitative user research is the process of collecting and analyzing non-numerical data in the form of opinions, comments, behaviors, feelings, or motivations. Qualitative data aims to give an in-depth look at human behavioral patterns.*

Qualitative research is often carried out as interviews, usability tests, or surveys. It provides an excellent opportunity to receive critical feedback and collect qualitative data. One benefit of using qualitative research is that it can be conducted at any point in the design process and is usually carried out in small groups with 5-8 participants. Both of these are essential factors, especially in small development projects [27].

### 2.6.1 Usability Testing

Usability testing is about testing the product and its design on a group of representative users. This can be done by observing the users and their reactions as they attempt to complete given tasks. Usability testing is usually carried out several times during the product development. In this way, the development team always receives feedback that can improve the product.

Usability tests can either be carried out in-person, remotely, or in a guerilla way. The guerilla method is a very inexpensive and quick way to perform user testing. It usually lasts 10-15 minutes, and the test persons could be found by approaching random people in public or colleagues in the workplace. In-person testing is good for getting a better sense of the test person's reactions and the mood in the room but requires a restrained and empathetic moderator for the notation. Remote usability testing could catch the users in their environment, which is a benefit because that is where the product should be used [28].

## 2.7 Agile Software Development

The most popular software development methodology is the Agile approach. Agile is defined as an iterative software development approach where value is provided to users in small increments rather than through a single significant launch. The requirements and results are continuously evaluated by the team, which leads to changes being implemented very effectively [29]. Two popular agile methodologies are presented in the following subsections.

### 2.7.1 Scrum

Scrum is the most popular framework for Agile software development. The idea behind it is to split the project into as many deliverable milestones as possible to get continuous feedback from the client. These phases last between 1 and 4 weeks and are called sprints [30]. Figure 6 shows the typical life cycle of a Scrum process. Each sprint contains a set of prioritized tasks listed in a product backlog by the Scrum team. The team must choose a Scrum master whose task is to ensure close cooperation across roles and functions, that the Scrum process is followed, and not least, that the team is as productive as possible.

The Scrum team meets for daily stand-ups, which are short and effective meetings where members can discuss progress and identify blockers. The idea is that the meetings are automatically shortened if the group members have to stand, hence the name. During the daily stand-ups, each team member should answer the following three questions [31]:

1. *What did I work on yesterday?*

2. *What am I working on today?*

3. *What issues are blocking me?*

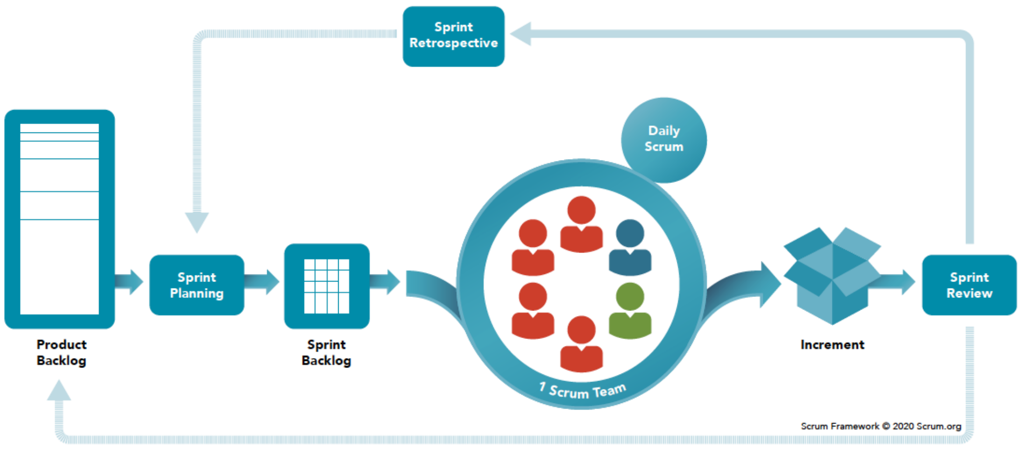


Figure 6: The life cycle of a Scrum process [32].

### 2.7.2 Kanban

Another popular framework for Agile software development is the Kanban method. It originated in the late 1940s in Japan due to Toyota wanting to improve its engineering and production processes. The company visualized its workflow using physical cards on a board to signal the different tasks [33].

An example of a Kanban board is shown in Figure 7. Translated from Japanese, the word *Kanban* means *card you can see* [34]. New cards are added to the board as new tasks appear. The cards get placed in different columns on the board, each of which represents a step in the process. Column headings are typically "to do", "in progress", and "done". On physical boards, sticky notes are often used as cards.

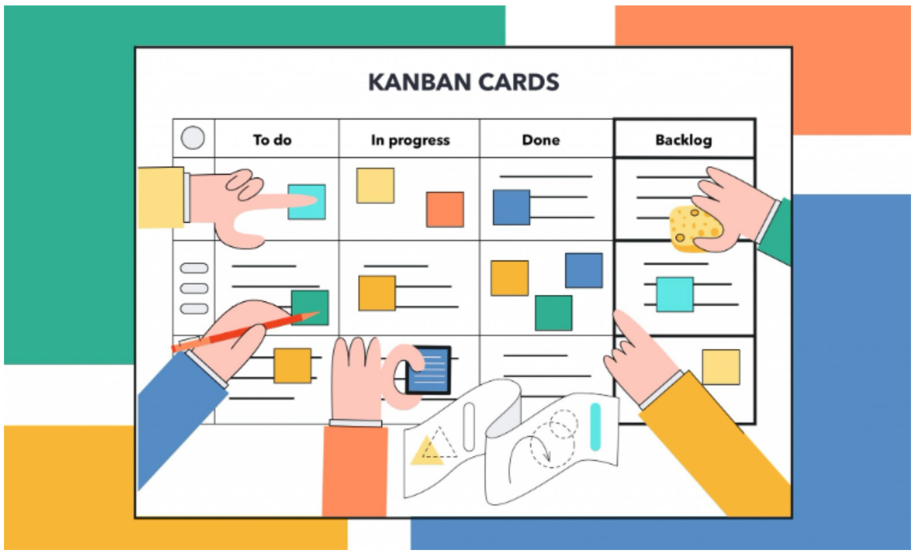


Figure 7: An example of a standard Kanban board [35].

# 3 Methodology

This chapter explains which methods were used throughout the project and how these were chosen based on the theory presented in the previous chapter.

## 3.1 Research Methods

Two qualitative research methodologies were used to comprehend and analyze the client´s requirements and to test the developed system on potential users for improvement and further work. These methods are described in the following two paragraphs.

### 3.1.1 Interviews

During the initial stage of the project, the team interviewed the client to gather information about the basic functionality requirements. Since one of the group members is employed by Stiftelsen Voll gård and actually responsible for the bookings, we had constant access to information and input. This also makes her a representative of the client, which helps them better to understand the needs and requirements through the development process.

### 3.1.2 Usability Testing

Usability testing was carried out in mid-April. Several questions the test subject had to answer before and after the actual testing were formulated. These questions are presented in Appendix E. The testing involved asking the test subject to perform an action, in this case, make a booking, and then observing and noting the test subject along the way. The tests lasted between 20 minutes and half an hour and were carried out in person. The tested version was one of the first versions of the systems, similar to the wireframes.

## 3.2 Choice of Technologies

### 3.2.1 Visual Studio Code

Visual Studio Code was the code editor used during the project. VS Code is a free, open-source, cross-platform development tool made by Microsoft with Electron Framework [36, p. 1-2]. The tool is known to the group from previous projects. It is also the most popular development environment and thus well documented, which is a significant advantage. These were the two main reasons for choosing VS Code.

Some core features VS Code provides are built-in support for coding with many languages, a built-in debugger for Node.js, and version control based on Git [36, p. 3].

### 3.2.2 React

React (also known as ReactJS) is a JavaScript library developed by Facebook to create user interfaces for the web [37]. At the beginning of the project, it was discussed which front-end technology was the best fit for the project. Alternatives were Vue.js and React.js. React was chosen for several reasons, but mainly because it is very popular among developers and has a large community and good documentation, which can help to learn it quickly and find answers if you are stuck on a problem.

### 3.2.3 Node.js

Node.js is an open-source, cross-platform JavaScript runtime environment [38]. Node, React, and Firebase are complementary technologies used to develop the booking system for Voll gård. Node provides the back-end environment, while Firebase is a back-end platform. This language is used to connect React, as the front end, with Firebase.

### 3.2.4 Redux

The team used Redux together with React. Redux is a predictable state container for JavaScript, and it helps to write applications that behave consistently and run in different environments [39]. Using the Redux library helps to avoid difficulties with sending props between components.

### 3.2.5 Firebase

The team chose Firebase to be the back-end platform. Firebase is a mobile and web application development platform backed by Google. It is a one-stop solution to develop high-quality applications by focusing on the user's needs rather than the technical complexities [40]. Firebase cloud system provides SSL encryption data transmission [41]. That means all the connections between the users and Firebase services are encrypted, which can protect the data against unauthorized sides. Additionally, the platform offers several products, such as Firestore and Authentication, which were used in developing this system.

**Cloud Firestore** Cloud Firestore is also a cloud-hosted, NoSQL database [40]. The database in this project contains four collections in addition to the user collection in the Authentication service. Each collection includes documents that have the same schema. The team wrote security rules to manage access to those collections for users and admin.

**Authentication** is taken place by using email and password. Only the admin can be authenticated in the first version of the web application.

### 3.2.6 Docker with GitHub Actions

Docker was used besides the GitHub Actions to do Continuous integration (CI) workflows in this project. The team evaluated whether to use Docker with GitHub Actions or Travis CI. But because the project's GitHub repository is private, Travis CI provides a limited amount of free build time per month, beyond which users must pay for additional build time. Therefore, the team used Docker with GitHub Actions to automate building, testing, and deploying. Also, I've included more information about Docker with GitHub Actions below.

**Docker** is an open-source platform designed to help with application deployment using software containers. This approach means running applications with the complete environment (files, code libraries, tools, and so on). Docker, therefore, similar to virtualization, allows an application to be packaged into an image that can be run everywhere [42].

**GitHub Actions** is a workflow automation tool offered by GitHub. It allows developers to define custom workflows for their software development life cycle directly in their GitHub repositories. These workflows can be used for various tasks like building, testing, and deploying code. More information about how the CI took place can be found in Appendix D.

## 3.3 Version Control Tool

### 3.3.1 GitHub

Using a version control system in the project to collaborate in a remote repository and monitor and handle code changes was a matter of course. The choice was between GitHub and GitLab. The group is familiar with both through university coursework but has used GitLab in most previous projects. Considering the idea that the project should be able to be further developed by others, the choice fell on GitHub, as it is mainly used in working life and, therefore, easier to find documentation on.

## 3.4 Development Method and Tools for Project Planning

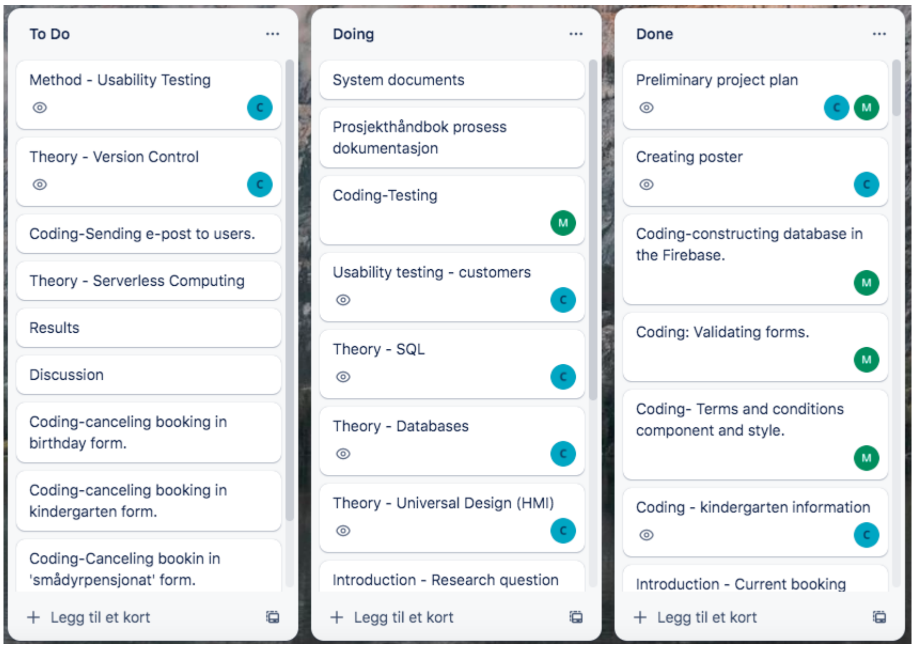
### 3.4.1 Kanban Method

As for project methodology, Kanban was used. The Kanban method was great for distributing tasks and keeping track of what the various team members had to do, were doing, and had already finished. A virtual tool was used for making the Kanban board. This tool is presented below.

### 3.4.2 Trello

As the group members mostly worked separately, it was concluded that a virtual tool had to be used for project management. The project management tool Trello was used for this. A board divided into *to do, doing* and *done* was created early in the project. Everything that had to be done during the project period was put on *to do* and prioritized. As tasks were started, they were moved to *doing* and marked with the name of the group member who worked on it. Completed tasks were continuously moved to *done*.

Tasks could be moved back to *to do* and later discussed in the group if a group member was stuck and needed help to figure it out. In this way, the group had an overview of everything that had to be completed and reassurance that they had worked on different tasks. A part of the Trello board is shown in Figure 8 below.



To Do
- Method - Usability Testing (C)
- Theory - Version Control (C)
- Coding-Sending e-post to users.
- Theory - Serverless Computing
- Results
- Discussion
- Coding-canceling booking in birthday form.
- Coding-canceling booking in kindergarten form.
- Coding-Canceling bookin in 'smådyrpensjonat' form.

+ Legg til et kort

Doing
- System documents
- Prosjekthåndbok prosess dokumentasjon
- Coding-Testing (M)
- Usability testing - customers (C)
- Theory - SQL (C)
- Theory - Databases (C)
- Theory - Universal Design (HMI) (C)
- Introduction - Research question

+ Legg til et kort

Done
- Preliminary project plan (C) (M)
- Creating poster (C)
- Coding-constructing database in the Firebase. (M)
- Coding: Validating forms. (M)
- Coding- Terms and conditions component and style. (M)
- Coding - kindergarten information (C)
- Introduction - Current booking

+ Legg til et kort

Figure 8: A cutout of the Trello board used during the project.

### 3.4.3 Pencil Project

Pencil Project is a downloadable, open-source GUI prototyping tool available for all platforms. The team used this program to create the wireframes in the early planning phase. The program is intuitive and uncomplicated, allowing users to test the prototype by clicking through an interactive version. The wireframes are presented in Appendix C.

### 3.4.4 Distribution of Work and Roles

In the early phase of the project, the team members chose not to assign tasks to specific members, as there was a need for a flexible solution due to very different life situations. All upcoming tasks were listed on the Kanban board in Trello, and each member could choose a suitable task to work on. By the end of every week, the group met to discuss what they had done and what should be done next.

Although the work was flexible, each member had responsibility for some administrative tasks, such as meeting notices and meeting minutes, in addition to updating documentation.

## 3.5 Other Tools

### 3.5.1 Microsoft Teams

Microsoft Teams was used for team collaboration between the two students and between the students and the supervisor. The platform was used to send invitations and set up digital meetings in the calendar, digital meetings, and communication with the supervisor, as well as file storage and communication between the students.

### 3.5.2 Overleaf

The group used Overleaf for writing the main report as one important criterion was collaboration tools. Overleaf is an online LaTeX writing tool. One of the group members had already used Overleaf in previous projects, including another bachelor's thesis, last year. In addition, both members signed up for a free online LaTeX course via the university library. Overleaf offers good documentation, instant access, tracking of changes, and descriptive error messages and lets you connect to tools such as Zotero [43].

### 3.5.3 Zotero

Zotero is a free tool for collecting and organizing research. The group used a shared library on Zotero to manage all sources of information found throughout the project. In this way, useful information found through literature searches earlier in the process could be easily retrieved. The references could be easily entered and automatically implemented in the project and bibliography in the Overleaf project.

### 3.5.4 Canva

Canva is an online graphic design tool used to create illustrations for documents, main report, and web application. One of these illustrations is the photo collage in Figure 1, as well as the photo collages and image carousel in the application.

# 4 Results

This chapter presents the project's results, divided into three parts: scientific, engineering, and administrative.

## 4.1 Scientific Results

The scientific results present and describes the current version of the booking system, which was developed during the project period. This will be the basis for the answer to the problem statement from Chapter 1:

*How can a web-based booking system simplify the booking process for visitors and provide a meaningful overview of daily activities for the client?*

### 4.1.1 User Interface

**Flowcharts**
The web application allows users to get all the necessary information about a specific service after selecting it from the navigation bar at the top of the page. The possibility of booking has been implemented for children´s birthday celebrations, animal boarding, school visits, and kindergarten visits. The booking process is illustrated in Figure 9 as a flowchart.

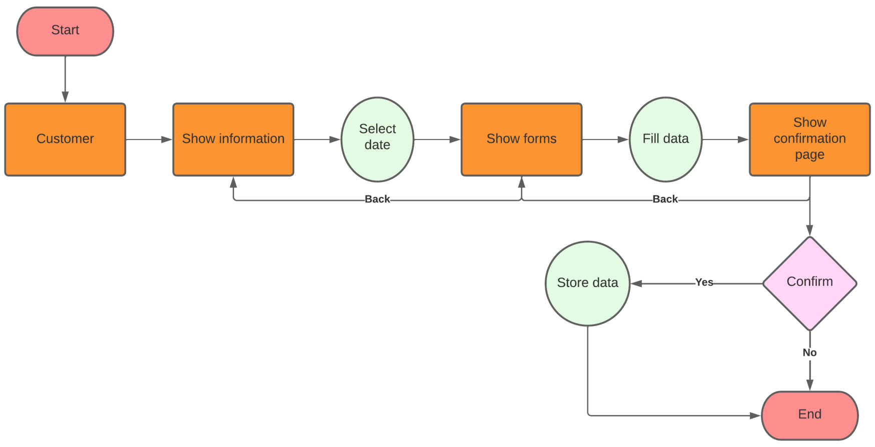


Figure 9: User flowchart.

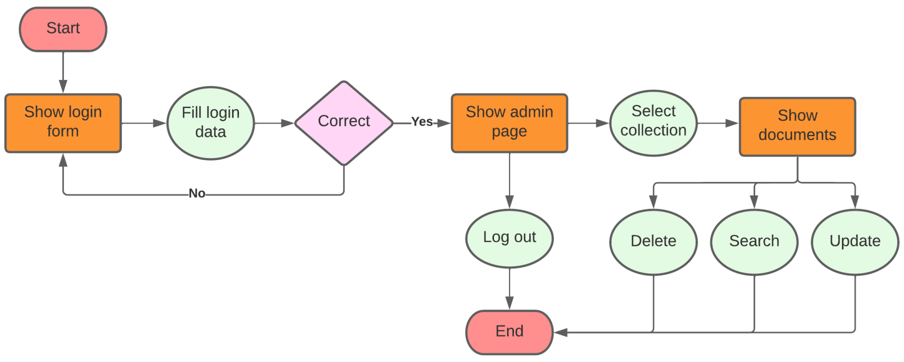


Figure 10: Admin flowchart.

The admin has to authenticate before retrieving, deleting, or updating data. Only the admin can sign up and log into this first version of the web application. These processes are illustrated as a flowchart in Figure 10

The different views are shown in Appendix F, and further described in the following paragraphs:

**Home Page**
This page is the first page the user visits when they want to make the booking or when the user clicks on the logo in the upper left corner. This page contains general information about Voll Gård and the experiences and services provided by the farm. On the navigation bar, there is a logo image on the left and a button to the administration login page on the left. The button is associated with the admin description and icon. The header contains a clickable menu with four different services. By clicking on each one, the user can navigate to another page.

**User Forms**
There are four types of user forms, in which the user can choose the required booking service to fill in data and complete the reservation. The booking forms are for birthday celebrations, animal boarding, school visiting, and kindergarten visiting. The following paragraphs give more information on how to book an event using the web application.

**Birthday Celebration**
After the user selects the birthday celebration from the header list, the user gets navigated to the next page. Here the user gets all information about the booking process, prices, and other related information. After reading this information, the user can choose a suitable date for the celebration. Then, the user navigates to the next page, where it enters all required data, including the time slot. By clicking on submit, the user navigates to the next page, where the user can check all the entered data and read the conditions and terms before completing the booking. If the customer accepts the condition, they can complete the booking. It is also possible to use the navigation back to fill in the data again. Successful booking, navigate the user to the confirmation page.

**Animal Boarding**
When the user selects animal boarding from the header list, the user will be navigated to the next page, where the user can get all information they need to make a booking, including the price and at what time slots the user can bring and pick up their animals. The user has to choose a date for bringing the animal. By clicking on, the user will be navigated to the next page where he has to fill in all required information, including how many nights the animal will be there. After filling in all the required information, the user can navigate to the last page, where the user can see all entered data summarized. But, before the user submits and completes the booking process, the user must accept the conditions and terms of using the service. The user can navigate to the previous page if they want to change booking data before sending the booking document into the Firebase Firestore.

**School Visiting**
In the header list, the user can also select a school visit service. After clicking on the event, the user can be navigated to the first page. The user can read and perceive all the information associated with school services. This information includes the opening times and prices. After the customer perceives all the information, they can select a date to navigate to the second page. On the second page, the user must fill in all compulsory data to navigate to the third and last page. On the last page, the user can check all entered data and navigate back by clicking on the button ”Tilbake” to change the data, or clicking on the ”Gå videre” to complete the booking and navigate to the confirmation page.

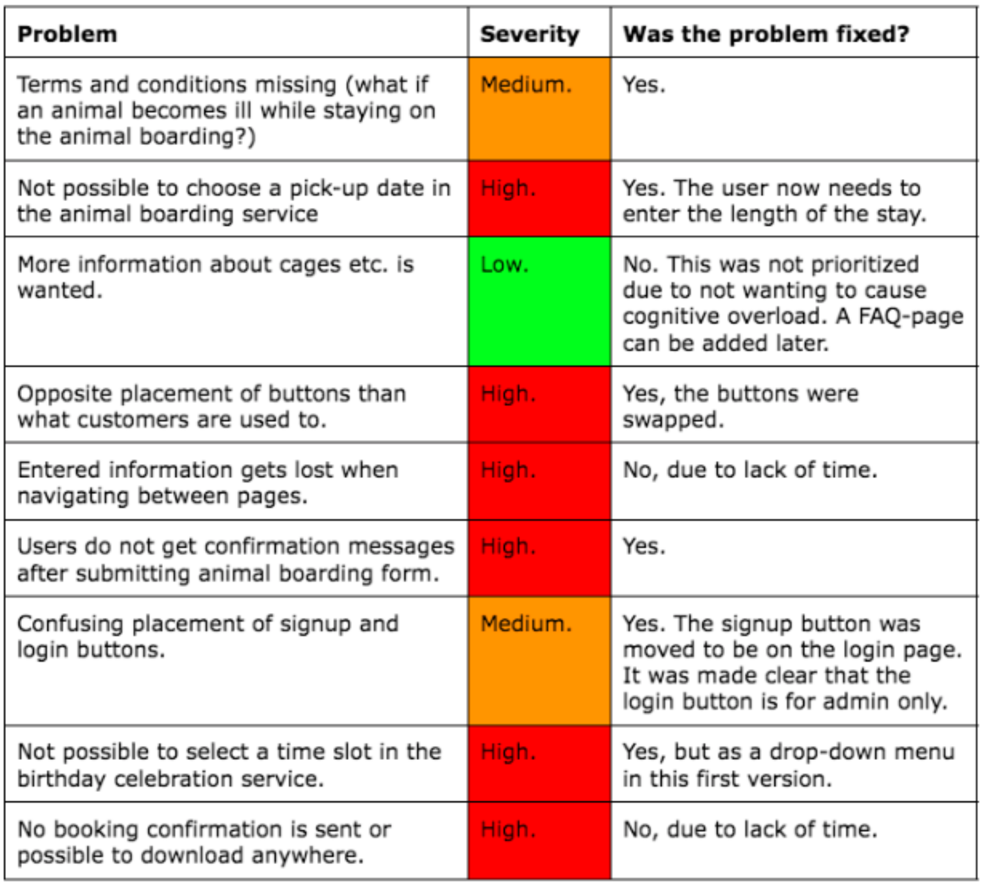
**Kindergarten Visiting**
The last booking service provided by the web application, is kindergarten visits. By selecting this service, the user gets navigated to the information page about the service. Additionally, the user must choose the date before going on to the next page. On the second page, the user will find all the input fields which should be filled in to make this type of booking. After checking and inputting data, the user will navigate to the last page. All the entered data can be read at the end of the process before confirming the booking and sending data into the backend database. It is possible to change the data by navigating back to the previous pages before submitting.

**Admin Pages**
Administrators can signup and login. The admin can see all booking data stored in Firebase on the admin dashboard. Searching for a specific booking event is possible by selecting a date or using an email or telephone number. It is also possible to update a specific field by double-clicking on it. Additionally, a booking can be canceled by clicking on the trash icon. This action can be confirmed by clicking on the right icon or canceled by clicking on the cross icon. The admin can log out and return to the home page.

### 4.1.2 Results of User Testing

Table 1 below presents a summary of the results after usability testing, the severity of each problem, and if it was fixed or not. See Appendix E for more information and a full description.

Table 1: Summary of usability testing results.



| Problem | Severity | Was the problem fixed? |
|---|---|---|
| Terms and conditions missing (what if an animal becomes ill while staying on the animal boarding?) | Medium. | Yes. |
| Not possible to choose a pick-up date in the animal boarding service | High. | Yes. The user now needs to enter the length of the stay. |
| More information about cages etc. is wanted. | Low. | No. This was not prioritized due to not wanting to cause cognitive overload. A FAQ-page can be added later. |
| Opposite placement of buttons than what customers are used to. | High. | Yes, the buttons were swapped. |
| Entered information gets lost when navigating between pages. | High. | No, due to lack of time. |
| Users do not get confirmation messages after submitting animal boarding form. | High. | Yes. |
| Confusing placement of signup and login buttons. | Medium. | Yes. The signup button was moved to be on the login page. It was made clear that the login button is for admin only. |
| Not possible to select a time slot in the birthday celebration service. | High. | Yes, but as a drop-down menu in this first version. |
| No booking confirmation is sent or possible to download anywhere. | High. | No, due to lack of time. |

### 4.1.3 Results of Code Testing

The testing is crucial to ensure that the system performance is optimal and that the system will not crash or collapse when the system is in using phase. Therefore the team writing unit testing first to check that every component work as expected. The testing components cover about 56.4% of the total source code.

## 4.2 Engineering Results

The engineering findings assess productivity in the context of the project's initial objectives.

### 4.2.1 Goal and objectives

The project's primary goal is to develop a web application that facilitates booking, which has been met.

### 4.2.2 Functional Requirements

The team tried to meet all functional requirements presented by the client at the beginning of the project. Unfortunately, some of the requirements could not be implemented in the project's current version. Table 2 illustrates which functional requirements have been met. More details about the functional requirements can be found in Appendix B.

Table 2: Functional requirements.

| **Functional requirement** | **Fulfilled** | **Not fulfilled** | **Comment** |
|---|---|---|---|
| The administrator can sign up, log in, and log out | * | | |
| The administrator can update a booking. | * | | |
| The administrator can filter a booking. | * | | |
| The administrator can cancel a booking. | * | | |
| The administrator can sort the booking list. | | * | |
| The user can select birthday booking. | * | | |
| The user can get all the required information about birthday booking. | * | | |
| The user can select a date in the birthday booking. | * | | |
| The user can select a time slot in the birthday booking. | | * | The time can be selected, but it is still available to others to select the same time slot. |
| The user can fill in all booking details in the birthday booking. | * | | |
| The user can see all the birthday booking information before submission. | * | | |
| The user can navigate to the previous page and change birthday input data. | * | | |
| The user can accept the birthday's terms and conditions before submitting. | * | | |
| The user can get all the required information about an animal hostel booking. | * | | |
| The user can fill in all required information about an animal hostel | * | | In the current product; the user can not choose more than one animal for each submission. |

| **Functional requirement** | **Fulfilled** | **Not fulfilled** | **Comment** |
|---|---|---|---|
| The user can see all animal hostel booking information before submission. | * | | |
| The user can accept an animal hostel's terms and conditions before submitting. | * | | |
| The user can navigate to the previous page to change animal hostel data. | * | | |
| The user can get all the required information about school visit booking. | * | | |
| The user can select a date in the school visiting booking. | * | | |
| The user can fill in all booking details in the school visit booking. | * | | |
| The user can see all the school visiting booking information before submission. | * | | |
| The user can navigate the previous page and change the school visit input data. | * | | |
| The user can get all the required information about kindergarten booking. | * | | |
| The user can select a date in the kindergarten visit booking. | * | | |
| The user can fill in all booking details in the kindergarten visiting booking. | * | | |
| The user can see all the kindergarten visiting booking information before submission. | * | | |
| The user can navigate to the previous page and change the kindergarten visiting input data. | * | | |
| The user can be confirmed when the submission is successful. | * | | After the submission, the user will be navigated to the confirmation page with thanking message |
| The user gets the email after confirmation to cancel a booking. | | * | |

### 4.2.3 Non-Functional Requirements

Seven non-functional requirements are defined in the Vision document, Appendix B. The outcome of these requirements is listed below.

**User-Friendliness**
A wide range of users of different ages and educations will be using the web application. Therefore, it was essential to have a simple and intuitive design. The developed application allows the user to find all information associated with the booking and lets it navigate between different services and pages. Every button and page has a simple and direct language that gives functions related to them. The team took a good time at the beginning of the project to make the design streamlined and easy to use.

**Reliability**
Reliability is an essential aspect of a booking system because the user relies on the system to make their bookings. Any errors could have negative consequences for the use and the farm. Therefore, the system will handle all errors when entering invalid data and personal information. All input fields have been checked to determine whether the input is valid before sending the data into the Firebase Firestore. Moreover, access to the database is controlled by writing security rules. The team also tested the code using unit testing to check most of the components to ensure that the system worked as it should.

**Accessibility**
Accessibility means ensuring that the web-based booking system is designed and developed to be accessible for people with disabilities who want to benefit from the system. The team followed the perceivable WCAG 2.1 principles to achieve this. Some guidelines could be met, but others could not in this first version of the web-based booking system. These will be further work.

**Security**
The system must be secured to avoid malicious attacks and protect users' data. Accessing the web application is not restricted to authenticated users. All users can fill in their booking data without signing up for the system. However, the administrator must be authenticated to have the right to read, update or delete data from the database.

**Maintainability**
Maintainability is essential to ensure that the system can be updated over time. The booking system was divided into small, reusable components, which made them easier to maintain. The documentation of the code has been written, which makes it easier for a developer to track and change coding later. The team also followed other practices to ensure the booking system was maintainable.

**Scalability**
Scalability means the ability of the system to handle a growing number of bookings. The team used Firebase as a back-end platform, which provides a scalable Firebase Firestore with a NoSQL database, authentication, and other services.

**Privacy**
To fulfill the booking process, the user should enter personal data, such as name, telephone number, and email. Therefore, it was necessary to consider privacy through the system's design. Some of the GDPR principles to deal with personal data have been applied, and some need to be further worked on to secure that sensitive data will not be used for unauthorized usage.

## 4.3 Administrative Results

The administrative results present the group's work process, including time usage and distribution of hours.

### 4.3.1 Time Usage and Distribution of Hours

**Time Usage**

Table 3 below shows the total amount of hours spent on the project and how the hours were distributed among the students. The students had a total of 1,000 hours at their disposal for the project and thus spent somewhat more than this.

Table 3: Time usage and distribution.

| Navn | Hours |
|---|---|
| Christel Ossletten | 571,5 |
| Mona Mousa | 590,5 |
| | **1162** |

**Distribution of Hours by Activity**

The pie chart in Figure 11 presents the distribution of hours by activity. This is based on the total of hours both students have spent on each activity.

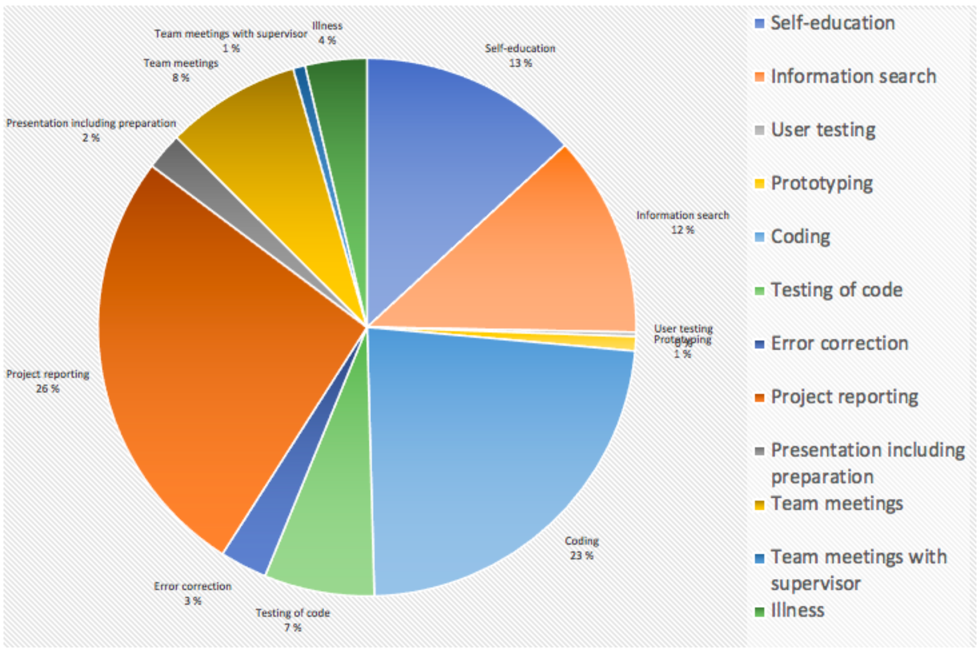


Figure 11: A pie chart representing time spent distributed across activities.

**Distribution of Hours by Category**
The pie chart in Figure 12 presents the distribution of hours by category. This is based on the total of hours both students have spent in each category.

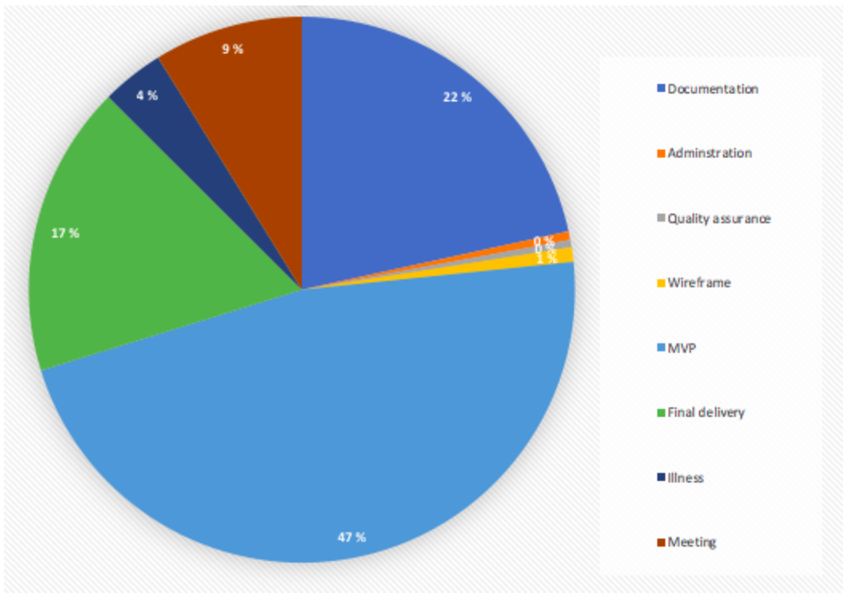


Figure 12: A pie chart representing the time spent across categories.

### 4.3.2 Process

The team benefited greatly from using a flexible variant of the Kanban method. The digital board made it easy to collaborate on the tasks even if the group did not physically sit in the same workplace. Although the group members live in different cities and have very different life situations, they did a lot to make the best of the situation and think that the method worked very well.

# 5 Discussion

The following section discusses the results presented in the previous chapter based on the theories in Chapter 2.

## 5.1 Discussion of Scientific Results

### 5.1.1 Universal Design

The team follows the universal design principles and WCAG 2.1 guidelines to ensure that the developed booking system is accessible and usable for a wide range of users. The following paragraphs explain how the team worked on the design to make it more accessible.

**Simple User Interface** The booking process is straightforward to follow. Additionally, the steps to complete a booking is minimized to include just the required ones to complete the booking. For example, if the user wants to book a birthday celebration, there are only three steps from start to finish. The first step is reading all relevant information and picking a suitable date, then going to the next step, where the user should enter all required data. The last step is to check the entered data, accept the terms and conditions, and confirm the booking.

The language of the web application is Norwegian (bokmål), which is the primary language of the target users. Unfortunately, the system does not support another language, such as English. The information associated with each booking form is presented in a clear and understandable format for all users. Additionally, all data is written in clear and large text, which will suit a wide range of users, including visually impaired ones. The headings are also written in large text and different colors to make it easier for the user to get the information they need quickly. Moreover, there is enough white space between and within the lines to improve readability.

All input data fields are associated with labels that describe what is required to be entered. For example, in the birthday form, the user is asked to enter a telephone number, and the user can perceive the data needed by the description associated with the label. And to further clarify, the entered field is associated with a placeholder, which describes what should be entered in the specific field.

The system also contains validation for input data. The user could get an error message under every input field if they entered unsuitable text or numbers, or left the input fields empty. The error message has a high color contrast to be visible to all users. The input validation can protect the personal data and the system from unauthorized use.

**Accessible** The team wanted to give all users the ability to access and use the developed system according to WCAG 2.1. In the following paragraphs, it will be explained how accessibility information was added to HTML elements using ARIA [44].

All labels are associated with the related input fields using the element "htmlFor" attribute. This can allow the screen reader to identify the purpose of each input field. In addition, all buttons are associated with descriptive text which explains the required action to navigate or submit a form.

The team also tested the system to check the ability to navigate and fill in all data in several ways. All forms, buttons, and links can be accessed using the mouse, touchscreen, and keyboard to make reservations.

ARIA attributes were used to make the system more accessible for users who rely on assistive technologies. To achieve this, the team used an "aria-placeholder" attribute for the input field, which describes the purpose of each input field in all different forms. For example, the input field which asks for the customer´s name in the birthday form is associated with ARIA-placeholder=" Fullt navn". In the same way, some pictures are associated with the "alt" attribute, which can be related to images. Additionally, ”aria-label”, ”aria-selected”, and ”aria-checked” attributes have been used with different HTML elements.

### 5.1.2 Continuous Integration and Continuous Development

At the beginning of the project, the team decided to do CI/CD towards the end of the project. The team used Docker to make CI and automate testing. It was planned to use a hosting service in Firebase to make the system available to users, but the project needs more development before having the system on hosting. Because of this, the team did not prioritize this.

### 5.1.3 Limitations of the Design

Today, quite a few users use mobile devices with small screens. The booking system developed in this project needs to be more responsive, meaning that the views do not adjust based on screen size at this point. This was not prioritized due to the limited time left. However, making the booking system responsive is essential for further work before making the system available for users, even though most admin users and teachers may use a computer to launch the application.

At the beginning of the project, the team considered creating an English version of the application so that it would be more accessible to foreign users. This feature was discarded for several reasons. Primarily because it is of low priority for the client at this stage. The limited time and experience also caused the team to work on other things rather than this functionality.

The developed system does not allow the user to modify their booking details or to cancel the booking after submitting. This makes the system less flexible and controlled. In the beginning, the team had planned to give the users the ability to sign up and log in, but the authentication requirement was discarded after interviewing the client. Therefore, this was replaced with sending the user a confirmation email after submitting, with the possibility to click in to change or cancel the booking. Unfortunately, the team failed to achieve the new functional requirement, so the users do not receive a booking confirmation.

### 5.1.4 Limitations of User Testing

The usability tests were carried out in the middle of April, and the version of the application that was tested was almost identical to the wireframes presented in Appendix C. Since this was an early version, the plan was to carry out more tests as more features were added and the design changed. Unfortunately, this was not feasible due to lack of time, which also resulted in only three test persons. We still see the feedback from these three as very valuable.

### 5.1.5 Limitations of Code Testing

To ensure that the system performs as expected, it should be tested before making it available to users. There are two types to code testing that should be implemented. Unit testing tests the separate components in the react to ensure that it functions as expected. The second type is the integration test, and this testing ensures that the whole system performs efficiently together, and this type is not used through testing the products of this project.

The team tried to test all the components to achieve at least 70% of the complete source code. But testing of Firebase was challenged for several reasons. First, the Firebase is real-time data synchronization [45]; when trying to test a component with Firebase functions, the message the team gets is, ”This usually means that there are asynchronous operations that weren’t stopped in your tests.”. The second reason is that two different Firebase services were used, Authentication and Firebase Firestore. Testing each of these services requires additional tools and is time-consuming. Thirdly, because Firebase is a cloud-based service, it was impossible to mock the database, making testing difficult for the team members. Additionally, the team needed more experience to do this type of testing.

Because of all of the reasons above, the team decided to spend less time on reaching a test grade of 70% of the source code. The time was instead used to make the web application more accessible and secure to give the user a better experience.

The testing coverage is about 56.4% of the whole source in this phase of the project. More about the testing can be found in Appendix D.

## 5.2 Discussion of Engineering Results

### 5.2.1 Functional Requirements

At the beginning of the project, the team interviewed the client to understand the functional requirements. Most of the functionality has been met, and some are left as further work, as seen in Table 2

Table 4: Number of implemented requirements vs number for further work.

| **Implemented** | **Further-work** |
|---|---|
| 27 | 3 |

One of the most critical functional requirements that still need to be met is sending the user an email confirmation of the booking and allowing the user to cancel the reservation via a link in the email. This function had no less priority than other functional requirements, but it was not achievable due to a lack of time and knowledge. Therefore, other tasks were prioritized in this first version. A confirmation email should, in addition to a summary of entered booking details, include information about the service, contact information, and so on. This feature has been replaced by navigating the user to a confirmation page.

Another functional requirement added late in the process is allowing the user to book a stay at the animal boarding for up to four animals simultaneously. In the current version, the customer must place one booking at a time. It should therefore be a different priority to add a functionality that allows the user to add multiple animals to the same booking.

Sorting the bookings on the admin page was less prioritized than other functional requirements. The admin can get all booking documents and delete and update all booking data. Therefore, more critical features were focused on.

### 5.2.2 Non-functional Requirements

The non-functional requirements, as described in the vision document, are user-friendliness, reliability, accessibility, security, maintainability, scalability, and privacy.

**User Friendliness**
The design should be easy to use and contains all the information the user needs to make a booking. This non-functional requirement has been achieved as follows:

Design: The web design is simple. Finding all information and navigating between several forms and pages is intuitive. Every page contains the primary and necessary elements.

Action: In all forms and pages, it is straightforward for users what action they need to take to complete the booking. Every button associated with text explains its purpose; e.g., the "Gå videre" button instructs the user to navigate to the next page. The button "Bekreft" instructs the user to complete the booking.

Navigation: The user can easily navigate to the next or the previous page. The booking process is divided into clear and distinct steps. To book school visits, the user has to read the information and choose a date before navigating to the next step. Then the user should fill in the required data before navigating to the next page, where the user can see a summary of the booking details before completing the booking. In the current version, the problem is the user must enter the data again, not just update the entered data when the user clicks on the back button. This is because the data will be stored in the store and sent to the database after submission.

Information: The booking application provides clear and concise information about the services. It includes a description of the services offered, prices, opening times, and other relevant information.

Flexible Options: Customers can fill in booking data and contact the farm if they need help. In the current version, the user can not modify or cancel a booking themselves but can get it done by a telephone call or email to the company.

**Reliability**
Reliability refers to the system's ability to perform accurately as expected without errors or data loss. And the booking system should be able to handle several reservations simultaneously without crashing. The team achieved reliability as follows:

Using Firebase Security Rules: The team writes security rules to manage access to the database on the Firebase Firestore.The access has been controlled for the authenticated admin's reading, updating, and deleting. At the same time, unauthenticated users can write to the database by submitting booking forms.

Implementing Data Validation: All entered data entered by users are validated on the front end before sending into the Firebase. The input data should be a string or number, limited in length, and not empty in most fields. For example, the user asked to input the email in the birthday form. The input email must be written in specific letters, at most 100 characters. Moreover, the user will be informed about the error clearly to know what the error happened when filling in the data.

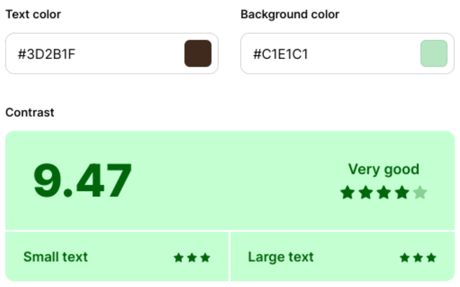
Implementing Unit Testing: The testing is crucial to check the web performs as expected. Therefore, the team tests most components on the front end separately to check each component. But not the entire system could be tested because of Challenges in testing the Firebase platform. More about testing can be found in the system document in Appendix D.

**Accessibility**
The team implements several features according to WCAG 2.1 to make a booking system more accessible for people with disabilities.

Semantic HTML: Using HTML tags to describe the content of the web pages to help screen readers and other assistive technologies understand the content. using <NavLink> tag with descriptive and meaningful text can be seen in the several links in a header; for example, a link to the Birthday page is associated with the meaningful text "Bursdagsfeiring." In the same way, tag <button> using descriptive text, for example, a button to complete booking associated with "Bekrefte bestilling.". All labels associated with the form element using the "htmlFor" attribute or using the <label>

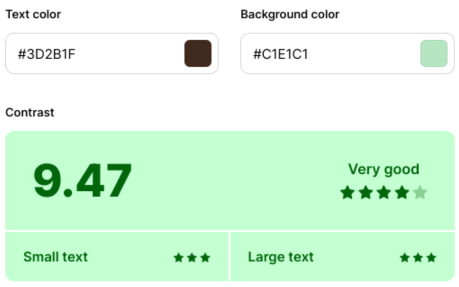
tag with form controls such as <select>, <input>, and <textArea>. This can be seen in all forms in the system. Another example of HTML elements is using table elements(table, tr,th,td) for displaying birthdays, Smådyerpansjonat, school, and kindergarten data, also using <th> to help the user understand the content of each column.

Contrast Color: Using color contrast is an important aspect of supporting accessibility in web-based booking systems. It ensures that users with visual impairments, including color blindness, can easily read and navigate a website.



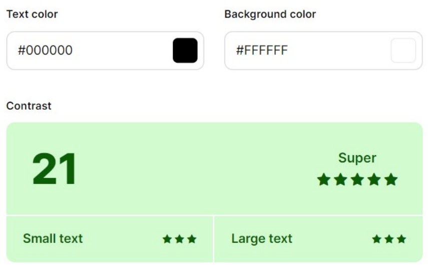
Text color: #3D2B1F
Background color: #C1E1C1
Contrast: 9.47 — Very good ★★★★
Small text ★★★ | Large text ★★★

(a) Contrast for white background and dark brown headings in most of the pages



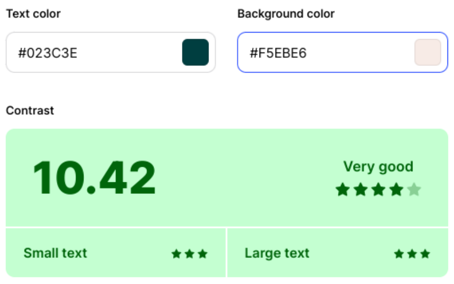
Text color: #3D2B1F
Background color: #C1E1C1
Contrast: 9.47 — Very good ★★★★
Small text ★★★ | Large text ★★★

(b) contrast of dark brown background and green background on the forms



Text color: #000000
Background color: #FFFFFF
Contrast: 21 — Super ★★★★★
Small text ★★★ | Large text ★★★

(c) Contrast for white background and black text in most of the pages



Text color: #023C3E
Background color: #F5EBE6
Contrast: 10.42 — Very good ★★★★
Small text ★★★ | Large text ★★★

(d) contrast of beige background and dark green icons and text

Figure 13: Colors contrast

To meet WCAG 2.1 guidelines, the contrast ratio should be at least 4.5:1 for standard text and 3:1 for large text (Level AA). Therefore, choosing a color for the background and text is very crucial. The background for the web booking system is white, and the text is black, which achieves a high-contrast color combination with a contrast ratio of 21 which is more than 7:1 (Level AAA). The button on the submit forms is dark green, and the text is white, which contrasts with 12:38, which is also level (AAA).

Instructions: The booking system users are of different ages and education. Therefore, the information and instruction for all booking events to guide users are written in simple, clear, and concise plain language. In addition, they are using the headings to break text into sections and organize the information. This can be seen at the head of each page, making it easier for users to scan and find the information they need.

**Security**
Security is crucial to protect users' sensitive data and prevent potential attacks. Through the system, the user must add personal data such as name, telephone, and email to be able to make a booking. Therefore, it is essential to consider security when the team is working on development.

Authentication: Controlling access is essential to secure the web application, especially when accessing the database. The authentication services in Firebase are used to authenticate the users. In the developed booking system, just the administrator can be authenticated. And the authentication is conducted by using email and a password. The user who wants to book a specific event does not need to be authenticated to make bookings.

Injection attacks: The project uses Firebase Firestore, a NoSQL database. The Firestore is designed to be more resistant to SQL injection attacks. However, the team members take precautions to prevent attacks. All input data has been checked before submitting to ensure that it contains just a string or numbers, in addition to length. Moreover, the Firebase security rules have been written to manage access to Firestore.

XSS attacks: To prevent users or Hackers from injecting malicious code into the database or the developed website through HTML, the team sanitizes all input field data. HTML sanitization will remove or escape all dangerous JavaScript or other code. The DOMPurify library has been used to check all input data before submitting the form.

**Maintainability**
Maintainability refers to the ease with which a web-based booking system can be maintained and updated over time. The following describes more how the team tried to achieve this non-functional requirement.

React Components: The project has been divided into smaller and reusable components. This can help to understand, test, and extend the project better. E.g., one component is "TermAndConditions," which is used by two other components, "Birthday" and "Smodyrpensjonat.". In the System document, Appendix D, more information about how the components are established and how they work together is presented.

Documentation of coding: The team writes documentation for all the components. That makes it easier for others in the team or other developers to understand coding for further work, update the existing code, or remember the coding when revisiting it later. In this project, the JavaScript library "JSDoc" has been used to generate the documentation. More about documentation can be found in Appendix D.

Testing: Writing testing catches any errors before making the booking system to production to ensure that the system is functioning as expected. Unit testing has been used to check the components. But the system needs to be tested more to check how all components are integrated by using integration testing to be sure that the system is functioning as expected. More about the testing and the test coverage percent can be found in the System document Appendix D.

GitHub: The team uses GitHub as version control to track any changes in the source code. This facilitates collaboration between developers and allows undo any changes.

**Scalability**
The booking system has been developed using React and Firebase; both are designed to be scalable. Moreover, The team began to create the system with scalability in mind—the ability to add new features and handle more bookings when the system is in production.

Reacts components: The system has been divided into small and reusable components, which makes it easier to add a new component to new features. Besides, using the Redux library, which manages the state through the system. Redux library help to send the data between different component easily and without complexities. Also, use React Router library, which handles navigation in the system. All of those libraries help to make the system more scalable.

Using Firebase Firestore: The Firestore is a NoSQL database. The database in this project is a document database containing four collections. Each collection stores data as objects in field-value pairs. As the amount of data stored in Firestore grows, the system can horizontally scale by adding more servers, enabling the system to handle more data without impacting performance.

**Privacy**
Privacy is critical to protect personal data from unauthorized use. The farm is responsible for legally using the data collected from its customers. However, the team considers the protection of entered data by following the GDPR principles three, and Principle six, which have been explained in the theory section.

Data minimization principle: To make a reservation for any event using the system, just the data related directly to the booking will be persisted. All booking events ask the user to enter their names, email, and telephone number in case the Voll farm wants to have contact with them and to identify them to confirm the booking.

Integrity and confidentiality principle: As mentioned in non-functionality security, all the entered data is checked against injection attacks to prevent hackers from stealing the personal data of the farm's customers.

## 5.3 Discussion of Administrative Results

### 5.3.1 Progress Plan

A Gantt chart was created at the beginning of the project in connection with the writing of the preliminary project plan. The Gantt chart is attached in a zip file containing the attachments to the preliminary project plan. The team decided to use days for each phase of the project instead of hours. This made it somewhat difficult to compare planned and actual time usage, but you can see that it was assumed that most time would be spent on documentation and coding, which was true. It should have been considered that so much time would be spent on self-education and information searches. In retrospect, we see that the Gantt diagram should instead have been based on hourly consumption.

### 5.3.2 Time Usage

A total of 1162 hours was spent on the project. The students had 500 hours each, i.e., 1000 in total, at their disposal and therefore used an extra 162 hours. In retrospect, we could have done more if we had distributed the self-education and information search differently. Both didn't need to read about and understand everything, as only one person performed some parts. At the same time, the project would have benefited from an extra team member, as the workload was heavy. Another option would have been to implement fewer services instead of starting with four at once. There was no one at the company with development experience or a related technical background, which was quite challenging as it was difficult to get any guidance or help.

### 5.3.3 Development Method and Cooperation

At the beginning of the project, the team evaluated what workflow methodology would suit the team members' life circumstances. One team member must work regular hours every day, and the other must take care of the children. That made it challenging for team members to be in the same place or work simultaneously. Therefore, the team had to decide which methodologies were more suitable to achieve the project goals and objectives. There were two choices; Scrum or Kanban. Because the Kanban methodology was more flexible and the best choice, the team decided to use it.

The project was divided into small tasks and items, and each task was first represented on a card. The team decided to put all the required tasks at once in the "To Do" stage on the Kanban board in the virtual tool Trello. The team thought it would be best to have all the tasks and make them available to work on, instead of dividing them into iterations. This was to give the team members more flexibility and make it more transparent what must be done.

The team also had a strategy on how to work with challenging tasks. When the member chose to work on a specific task and got a problem accomplishing it, the team discussed this problem and tried to solve it. If they could not solve the problem, the non-completed task would be returned to the "To Do" column. The team could try to work on this task later when they had more information and if there was time to try again. In this way, the team could deal with problems that arose through working.

The team had good cooperation. An agreement stating that each member could choose suitable tasks to work on, was written at the beginning of the project. This flexibility had to be limited with the responsibility to deliver the project on time. Therefore, there was no conflict, and the team managed to achieve the project's goal. At the beginning of the project the team worked on campus on Fridays, but decided to meet digitally for at least one hour the rest of the weeks.

# 6 Conclusion

## 6.1 Conclusion

The project's primary goal was to develop a web-based booking system for Voll gård, which can facilitate bookings for the customers, streamline the client's working day, and provide control and overview of all bookings. By leveraging modern technologies, such as React, Firebase, and Redux, the web application has been designed to be user-friendly and accessible for most of the farm's customers.

Two research methodologies have been used to collect all the required features and to test the developed system. The client was interviewed at the beginning of the project to understand the system requirements. In the middle of April, the developed booking system was tested by three persons through qualitative user testing for improvement. These methodologies helped to analyze and improve the system in the project's final phase.

The theories and literature necessary to develop a friendly and secure web-based system have been covered in the theory section. Because security and privacy are crucial, primarily since the developed booking system collects personal data to make a booking, it was essential to apply this theory to ensure the legal handling of personal data, such as following GDPR guidelines and checking input data against illegal attacks. Additionally, the system will be used by a wide range of users of different ages, cultures, educations, or disabilities, meaning that the developed system must be accessible to most users. The color contrast in the whole system is high and in level(AAA) according to WCAG 2.1 principle one. Additionally, all the information is written in clear and large text. Other guidelines of WCAG 2.1 principle one have been met to make people with disabilities interact with the booking system efficiently.

Although the project's goal has been met, some features still need to be fulfilled in the subsequent phases. The design needs to be more responsive, which means the users will not be able to benefit from this system on small devices. User testing was also conducted on the first version of the design, which was similar to the wireframes. After this, the booking system has been developed further, but the new improvements have yet to be tested. Source code testing did not reach a 70% coverage rate due to challenges with testing Firebase. The test coverage rate ended up being 56,4%.

The cooperation between the team members was also essential to get the best results. The team chose Kanban as an agile methodology. The tasks that should be done were on a digital board on Trello, and each member was free to select a task to work on. This flexibility helped the team to manage their different life circumstances and to deliver the project at the determined time.

Although the team spent more hours on the project than planned, it would have been an advantage to adjust down the number of services that were implemented, and rather focus on more important functions that could not be implemented due to lack of time.

The web application can be developed and enhanced to be more accessible and secure. Hence, to make the client and users more satisfied. Some features must be met before the system is hosted on the Internet, whereas others can be added in subsequent phases.

In conclusion, the web-based booking system project for Voll Gård has successfully developed a user-friendly, accessible, and secure product. This product simplifies the booking process for customers and the client. The customers can make their bookings efficiently, and the client can consume time and money on other things, in addition to having more control over the booking process. The developed product has limitations that can lead to further work, but in general, the team is proud to manage to deliver the project at the determined time. Moreover, the project provided the team with valuable experiences and enhanced their web development and project management knowledge.

## 6.2 Further Work

Some of the required functionalities, which were addressed at the beginning of the project, still need to be met by the end of the project. In addition to new features, which the client did not require, the team members think it is necessary to make the web application more responsive and accessible.

The following features should be satisfied regarding the problem statement addressed at the beginning of the thesis. Some of them are necessary to make the web application more beneficial. The features are listed from high to low priority.

**Privacy-** Protecting the user's data is crucial. So, it is essential to keep all personal data stored after booking for a limited time. This information should be deleted regularly and automated by limiting the time of holding the personal data to a maximum of six months after booking. The client should inform customers about their privacy practices, and ask them to accept that personal data will be stored by ticking a box.

**Sending a booking confirmation-** After the user submits the reservation with all the required information, they should receive an email with the booking details and a button to cancel the booking if something unexpected happens. The user can cancel the booking on the current version by contacting the admin.

**Calendar customized by the client-** Because there is so much going on at the farm at the same time, and no two weeks are the same, the client should be able to put dates and time slots as available in the calendar. Some days they do not have the time for visitors at all, and in cases like these it is important that this date is disabled in the calendar.

**Disable already booked dates and time-slots-** When the user books a birthday event, checking the time availability is impossible. Therefore, it is essential to show the available time slots and disable the reserved time when it is unavailable to avoid conflict with other reservations.

**Optimization for mobile devices-** The current web application is not responsive for smaller screens. Since many users make reservations on their mobile devices nowadays, it is essential to consider this feature in subsequent development.

**User testing with more customers-** The team could not test enough samples of users to check functionality and non-functionality features at the latest version of the application. Testing is essential throughout the development process to ensure that the web application satisfies the requirements of target users. It will be necessary to consider this in the next version.

**Adding audio to the several forms -** Incorporating audio into a booking system can help to make the system more accessible to a broader range of users, particularly those with disabilities or other challenges. By providing clear, concise instructions and allowing users to fill out forms using their voice, a booking system can ensure that everyone has equal access to the services and opportunities it provides.

**Booking a stay on the animal boarding for more than one animal-** On the current booking system version, the user can only book a stay for one animal on each form. The user must fill in several forms if they want to book for more than one animal, which consumes time. Hence, this functionality should be met in the sequence versions.

**Custom booking forms-** The current web application allows the booking of four types: birthday, animal boarding, school, and kindergarten. The client wants to be able to add and customize new types of bookings, but this was not a high prioritized and can be done as further work.

**User authentication -** On the current version, the user can make a reservation without needing to be authenticated. The user should be able to see their bookings, update them, and delete them. This feature was not required at this phase of the project, but can be done in the subsequent versions.

**Price estimate and online payment -** The payment of birthday celebrations and stays at the animal boarding is now done by Vipps at the end of the event. A possibility could be automating payment in the web application to make it more efficient. The user then receives a receipt after paying to ensure booking and paying. This functionality was not required in the first version, as there are often changes from the original number of participants for a birthday celebration, or an extra night at the hostel, which will change the amount of the payment. This is often not known until the same day. A price estimate based on the information provided would have been an idea, and was also in the feedback from one of the test persons.

# 7 Social Impact

Beyond the functional benefits mentioned above, the developed booking system also had a profound social impact.

One of the most significant ways booking systems have had a social impact is by increasing the ability to access various services provided by the farm. In the old booking system, the users have to call or exchange emails with employees to understand how to make a booking and to book. That consumes time and resources, which can make the booking process difficult and less efficient, in addition to misunderstandings arising from human mistakes in sending or receiving information. With a web application, these barriers can be overcome. Users can reserve services remotely and plan their visits, reducing the likelihood of being turned away due to capacity or other limitations.

The customers get detailed information about the services, including prices and terms and conditions. This increases the transparency provided by the farm, helps prevent illegal activities, and reduces the risk of overpaying. In addition, the product can significantly enhance the customer experience and thus increase satisfaction. Customers can easily see the availability of services and activities and make bookings at their convenience. This streamlined process reduces the risk of error and miscommunication, which could be a problem in the old system. A positive customer experience can also lead to positive word-of-mouth reviews, which can benefit Voll gård and the community.

Automating bookings can also positively impact the productivity and efficiency of the farm business. In this way, the employees can be free to focus on other tasks, such as providing good customer service and being there for the visitors. Moreover, the web application can provide valuable data about reservations which the farm can use to improve its services and optimize its operations. These efficiencies can lead to cost savings, which can be reinvested to give customers better experiences.

Moreover, the virtual booking system can also positively impact the environment. By making the booking process more accessible for all users to plan and book their events in advance, it can reduce last-minute cancellations and no-shows. This, in turn, reduces the waste and saves resources. Some people visit the farm to get information about bookings and make reservations. Still, by automating the booking system, the customer no longer needs to visit the farm physically, which helps to reduce carbon emissions from transportation. The paper use is also significantly reduced, as all information, as well as terms and conditions, as all information is provided online.

Automating booking systems can also positively impact customers by protecting them from discrimination. The system helps to remove the potential for human bias by allowing the customer to make the reservation without needing to contact humans physically.

On the other hand, there are also potential negative social impacts of automating the booking system. Privacy is crucial, and with the developed system, the users will enter their data, such as name, email, and telephone number. By using the internet, there is a concern that privacy will be violated and personal data will be used for unauthorized purposes. Therefore, it is essential to make a web-based booking system more secure by testing all input data and website performance. Additionally, the GDPR guidelines must be followed to protect the personal data and privacy of the users.

## References

[1] 'What is web application (web apps) and its benefits', Software Quality. (), [Online]. Available: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app (visited on 30th Apr. 2023).

[2] O. Uryutin. 'A brief history of web app', Medium. (13th Sep. 2018), [Online]. Available: https://oleg-uryutin.medium.com/a-brief-history-of-web-app-50d188f30d (visited on 30th Apr. 2023).

[3] 'Article 2 – definitions — united nations enable'. (), [Online]. Available: https://www.un.org/development/desa/disabilities/convention-on-the-rights-of-persons-with-disabilities/article-2-definitions.html (visited on 15th May 2023).

[4] 'Universell utforming', Bufdir. (), [Online]. Available: https://www.bufdir.no/likestilling/universell-utforming/ (visited on 4th May 2023).

[5] https://www.smashingmagazine.com/author/dannyhalarewich. 'Reducing cognitive overload for a better user experience', Smashing Magazine. Section: General. (0), [Online]. Available: https://www.smashingmagazine.com/2016/09/reducing-cognitive-overload-for-a-better-user-experience/ (visited on 4th May 2023).

[6] 'What is a database?' (), [Online]. Available: https://www.oracle.com/database/what-is-database/ (visited on 4th May 2023).

[7] jennifer. 'Exploring the differences between SQL and NoSQL databases: Is oracle a SQL or NoSQL database?', rkimball.com. (), [Online]. Available: https://www.rkimball.com/exploring-the-differences-between-sql-and-nosql-databases-is-oracle-a-sql-or-nosql-database/ (visited on 2nd May 2023).

[8] 'What is SQL? definition, elements, examples, and uses in 2022', Spiceworks. (), [Online]. Available: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-sql/ (visited on 3rd May 2023).

[9] 'SQL introduction'. (), [Online]. Available: https://www.w3schools.com/sql/sql_intro.asp (visited on 3rd May 2023).

[10] 'What is database scaling?', MongoDB. (), [Online]. Available: https://www.mongodb.com/databases/scaling (visited on 3rd May 2023).

[11] martinekuan. 'Non-relational data and NoSQL - azure architecture center'. (), [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data (visited on 3rd May 2023).

[12] 'Scaling horizontally vs. scaling vertically', Section. (), [Online]. Available: https://www.section.io/blog/scaling-horizontally-vs-vertically/ (visited on 2nd May 2023).

[13] 'Document database - NoSQL', MongoDB. (), [Online]. Available: https://www.mongodb.com/document-databases (visited on 2nd May 2023).

[14] 'What is a key-value pair (KVP)? definition and examples', Indeed Career Guide. (), [Online]. Available: https://www.indeed.com/career-advice/career-development/key-value-pair (visited on 2nd May 2023).

[15] 'What is a key-value database?', Redis. (), [Online]. Available: https://redis.com/nosql/key-value-databases/ (visited on 2nd May 2023).

[16] T. C. Development. 'Wide-column database', ScyllaDB. (), [Online]. Available: https://www.scylladb.com/glossary/wide-column-database/ (visited on 2nd May 2023).

[17] 'Introduction to graph database on NoSQL', GeeksforGeeks. Section: DBMS. (25th Dec. 2021), [Online]. Available: https://www.geeksforgeeks.org/introduction-to-graph-database-on-nosql/ (visited on 2nd May 2023).

[18] S. Kumar, R. Mahajan, N. Kumar and S. K. Khatri, 'A study on web application security and detecting security vulnerabilities', in *2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Sep. 2017, pp. 451–455. DOI: 10.1109/ICRITO.2017.8342469.

[19] ‘What is web security?’ (), [Online]. Available: https://www.goodfirms.co/glossary/web-security (visited on 16th May 2023).

[20] ‘OWASP top 10:2021’. (), [Online]. Available: https://owasp.org/Top10/ (visited on 14th May 2023).

[21] W. G. J. Halfond, J. Viegas and A. Orso, ‘A classification of SQL injection attacks and countermeasures’,

[22] ‘Applied network security : Master the art of detecting and averting advanced network security attacks and techniques’. (), [Online]. Available: https://web.s.ebscohost.com/ehost/ebookviewer/ebook/bmxlYmtfXzE1MTMzNTNfX0FO0?sid=03482c06-4145-4274-9ff2-e60ffee354ab@redis&vid=0&format=EB&lpid=lp_175&rid=0 (visited on 15th May 2023).

[23] Hanqing Wu and Liz Zhao, *Web Security : A WhiteHat Perspective*. Boca Raton: Auerbach Publications, 2015, ISBN: 978-1-4665-9261-2. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=974054&site=ehost-live&scope=site.

[24] ‘What is cross-site scripting (XSS)? how to prevent it?’, Fortinet. (), [Online]. Available: https://www.fortinet.com/resources/cyberglossary/cross-site-scripting (visited on 16th May 2023).

[25] ‘What are the 7 main principles of GDPR?’ Section: Articles. (15th Nov. 2022), [Online]. Available: https://www.gdpreu.org/7-main-data-protection-principles-under-gdpr/ (visited on 16th May 2023).

[26] ‘GDPR principles’, GDPR Toolkit. (), [Online]. Available: https://www.gdpr-toolkit.co.uk/principles/ (visited on 16th May 2023).

[27] ‘Quantitative vs. qualitative UX research [complete guide]’. Running Time: 857 Section: UX Design. (29th Jan. 2021), [Online]. Available: https://careerfoundry.com/en/blog/ux-design/quantitative-vs-qualitative-ux-research/ (visited on 19th May 2023).

[28] ‘What is usability testing?’, The Interaction Design Foundation. (), [Online]. Available: https://www.interaction-design.org/literature/topics/usability-testing (visited on 19th May 2023).

[29] ‘Agile software life cycle, methodology, examples’, Spiceworks. (), [Online]. Available: https://www.spiceworks.com/tech/devops/articles/what-is-agile-software-development/ (visited on 16th May 2023).

[30] ‘Agile scrum methodology — scrum life cycle phases and basics’. (), [Online]. Available: https://www.rfwireless-world.com/Articles/agile_scrum_methodology.html?fbclid=IwAR0N9b04vNO8CBI6rCyPp3hzPAXjS EyaVNA9rKli-bv6wUeVygjY (visited on 16th May 2023).

[31] ‘Standups for agile teams — atlassian’. (), [Online]. Available: https://www.atlassian.com/agile/scrum/standups?fbclid=IwAR3BO4aPk2T0K2Q3PJT-I8yYo9aCOPiKH9ncCVjsWbZ1OLBTJ4PObIE2j_U (visited on 16th May 2023).

[32] ‘What is scrum?’, Scrum.org. (), [Online]. Available: https://www.scrum.org/resources/what-scrum-module (visited on 16th May 2023).

[33] ‘What is kanban methodology? the ultimate guide — wrike’. (), [Online]. Available: https://www.wrike.com/kanban-guide/what-is-kanban/ (visited on 29th Apr. 2023).

[34] ‘What is kanban? a definition from WhatIs.com’. (), [Online]. Available: https://www.techtarget.com/whatis/definition/kanban (visited on 19th May 2023).

[35] A. Sergeev. ‘Online kanban board’, Hygger: Project Management Software & Tools for Companies. (27th May 2016), [Online]. Available: https://hygger.io/blog/online-kanban-board/ (visited on 16th May 2023).

[36] Alessandro Del Sole, *Visual Studio Code Distilled - Evolved Code Editing for Windows, macOS, and Linux*. Second Edition. Cremona, Italy: Apress, 7th Jun. 2021, ISBN: 978-1-4842-6900-8. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4842-6901-5 (visited on 29th Apr. 2023).

[37] E. Elrom, ‘Learn react basic concepts’, in *React and Libraries: Your Complete Guide to the React Ecosystem*, E. Elrom, Ed., Berkeley, CA: Apress, 2021, pp. 1–19, ISBN: 978-1-4842-6696-0. DOI: 10.1007/978-1-4842-6696-0_1. [Online]. Available: https://doi.org/10.1007/978-1-4842-6696-0_1 (visited on 27th Apr. 2023).

[38] ‘Node.js’, Node.js. (), [Online]. Available: https://nodejs.org/en (visited on 27th Apr. 2023).

[39] ‘Getting started with redux — redux’. (18th Aug. 2022), [Online]. Available: https://redux.js.org/introduction/getting-started (visited on 27th Apr. 2023).

[40] Harmeet Singh and Mayur Tanna, *Serverless Web Applications with React and Firebase : Develop Real-time Applications for Web and Mobile Platforms*. Birmingham: Packt Publishing, 2018, ISBN: 978-1-78847-741-3. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1789489&site=ehost-live&scope=site.

[41] W.-J. Li, C. Yen, Y.-S. Lin, S.-C. Tung and S. Huang, ‘JustIoT internet of things based on the firebase real-time database’, in *2018 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE)*, Feb. 2018, pp. 43–47. DOI: 10.1109/SMILE.2018.8353979.

[42] Rafał Leszko, *Continuous Delivery with Docker and Jenkins : Create Secure Applications by Building Complete CI/CD Pipelines, 2nd Edition*. Birmingham: Packt Publishing, 2019, vol. 2nd ed, ISBN: 978-1-83855-218-3. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2149487&site=ehost-live&scope=site.

[43] ‘Overleaf features & benefits’. (), [Online]. Available: https://www.overleaf.com/about/features-overview (visited on 28th Apr. 2023).

[44] ‘Using ARIA’. (4th May 2017), [Online]. Available: https://www.w3.org/TR/aria-in-html/ (visited on 19th May 2023).

[45] ‘Firebase products’. (), [Online]. Available: https://firebase.google.com/products-build (visited on 15th May 2023).

# Appendix

## A Preliminary Project Plan

**123**

# Developing a Booking System for Voll Gård

Preliminary Project Plan

**Version <1.5>**

## Revision history

| Date | Version | Description | Authors |
|---|---|---|---|
| 20/01/23 | 1.1 | Started distributing work and writing what we could before kick-off meeting with client and supervisor. Planned main activities | Christel O., Mona M. |
| 23/01/23 | 1.2 | Working with Gantt-chart and briefing | Christel O., Mona M. |
| 24/01/23 | 1.3 | Project description and goals after meeting with client | Christel O., Mona M. |
| 26/01/23 | 1.4 | Quality assurance and risk analysis | Christel O., Mona M. |
| 27/01/23 | 1.5 | Completion of Gantt-chart and preliminary project plan after meeting with supervisor | Christel O., Mona M. |

## Table of Contents

1. Goals and Framework ....4
1.1. Briefing ....4
1.3 Impact Goals ....5
1.4 Needs and Limitations ....5
2. Organization ....6
3. Implementation ....6
3.1 Main Activities ....6
3.1.1 Documents ....6
3.1.2 Meeting Summons and Meeting Minutes ....6
3.1.3 Coding ....7
3.1.5 Presentations ....7
3.2 Milestones ....7
4 Follow-up and Quality Assurance ....8
4.1 Quality Assurance ....8
4.2 Reporting ....8
5. Risk Analysis ....9
6. Appendixes ....11
6.1 Schedule – Gantt Chart ....11
6.2 Address List ....11
6.3 Agreement Documents ....11
6.3.1 Arbeidskontrakt ....11
6.3.2 Standardavtale ....11

# 1. Goals and Framework

## 1.1. Briefing

Voll gård is a 4-H farm located close to the center of Trondheim where children can learn by doing. Thousands of people visit the farm every year; the majority are kindergarten children and pupils from 1st to 7th grade wanting to learn about Norwegian agriculture and animal husbandry. The farm is also open to the public and families who wish to visit the animals.

4-H is a network of youth organizations whose mission is engaging youth to reach their fullest potential while advancing the field of youth development. [1] The 4-H pledge goes as follows:

I pledge my *head* to clearer thinking,
My *heart* to greater loyalty,
My *hands* to larger service,
And my *health* to better living,
For my club, my community, my country, and my world. [2]

The farm offers various other services such as a horse club for children, animal boarding, renting premises, and hosting multiple events, including children´s birthday parties. The booking of these services is currently made by phone and three different emails. This makes it very time-consuming, as communication must go back and forth before a confirmation can be sent to the customer.

The problem has been debated for a long time, and there has been a desire for a functioning booking system where customers can book services at available times. One of the members of the bachelor group, Christel, is employed at the farm and is the one who receives these bookings daily. After discussing with the manager and other employees, it was concluded that developing a booking system would be suitable for a bachelor´s thesis.

## 1.2 Research Questions/Project Description and Goals

This project aims to develop a booking system accessible through the web. The group will focus on developing a booking system for educational visits, children´s birthday parties and animal boarding. Users can view and book available time slots in a calendar view. The client must be able to select these available time slots as the capacity of employees changes throughout the year based on how much farm work there is, holiday closures, etc. Through the thesis, the following question will be attempted to be answered:

*How can a web-based booking system simplify the booking process for visitors and provide a meaningful overview of daily activities for the client?*

The goals of the project were first defined in appendix 6.3.1 - *Arbeidskontrakt.* At the end of the thesis, the group expects that Voll gård will have a functioning booking system integrated into their website and that it can be further developed. In addition, the thesis shall be written scientifically so that others with the same technical background can read and understand the content. Even people without technical knowledge should be able to read it and understand the technology used for the booking system. Also, the group members are ambitious to get a good grade for the thesis.

## 1.3 Impact Goals

The impact goals are defined by the client and bachelor group and will benefit both parties.

- The product streamlines booking handling for the client.

- The product is clear and makes it easy for visitors to book services.

- The group acquires new knowledge and experience and feels confident in the technologies used during the project.

- The project is motivating, and the experience the group acquires during the project is relevant to working life.

## 1.4 Needs and Limitations

The group has 1000 working hours at its disposal, which means that each student has 500 working hours. As time is limited, proof of concept would be more feasible than guaranteeing a fully functioning booking system at the end of the project.

A meeting with the supervisor will be held every two weeks. There will also be an ongoing dialogue with the client and meetings when necessary.

The project will need a database for storing booking data and a host to store the database, and software needs, such as project management software and optional software for programming.

## 2. Organization

The stakeholders who are involved in this project are as follows:

- The group of students who will carry out the project. The members of the bachelor group are Christel Ossletten and Mona Mousa.
- Stiftelsen Voll gård which is the client, represented by CEO Asbjørn Barlaup.
- A supervisor from NTNU, Elise Klæbo Vonstad, that will follow the group throughout the project.
- Teachers, students and families who would like to book a visit to the farm.

## 3. Implementation

### 3.1 Main Activities

To answer the research question, several activities should be completed. Writing all required documents, the main report, and coding will be done by the students.

#### 3.1.1 Documents

First, the group members will have a kick-off meeting between the students, the supervisor, and the client, which has already been conducted on 24.01.23, and 27.01.23. Through this meeting, the standard agreement will be signed, and a discussion about how the work should be done until delivering the final product and answering the research questions. After completing the kick-off meeting, we should write the preliminary project plan document that describes the research question, limitations, involved stakeholders, risk analysis, etc. This document will be submitted on Blackboard on 27.01.23.

The group members will interview the client to clearly understand the website's requirements and basic functionality, which is essential to benefit from it. Potential visitors are also going to be interviewed about how they think the system works today, and what they wish for in a booking system. After having an idea about the requirements, the group will work to develop a prototype. Getting approval from the client before starting to code is essential. This prototype will be updated if it is necessary. The requirements document should be written, after delivering the preliminary project plan, at the end of week 5. At the same time, the group will start writing a vision document which will be updated until submitting all process documentation. The group will work on the system documentation simultaneously with coding at the beginning of February, and it will be revised until the document is submitted by the end of May.

#### 3.1.2 Meeting Summons and Meeting Minutes

It is essential to have frequent meetings with the supervisor and client. The group members will send a meeting summon to the supervisor every 2 weeks and meet the client when necessary. Meeting notes will be uploaded on Microsoft Teams so that the supervisor can approve them. By the end of the thesis, the project handbook, which includes essential meeting summons and meeting minutes, will be delivered as part of the documentation.

### 3.1.3 Coding

After developing a preliminary prototype for the website and choosing technology, such as a programming language and platform that gives the required functionality, the group will start to code. The plan is to begin coding in week 6. It is essential to acquire knowledge about new technologies chosen in parallel with writing code. To ensure the coding quality, testing and automating the integration of code changes to get a single website project is crucial. Gaining knowledge about continuous integration is important and depends on the group member's time and ability to learn. Because the product is a website for online booking, two projects should begin simultaneously; a front-end project and a back-end project. Both projects will be on GitLab.

Unfortunately, there are no employees with technical qualifications at the farm. Therefore, the group members will contact the supervisor and friends with these qualifications if we need help.

### 3.1.4 Main Report

Writing the main report is central to the thesis. The group members should work with several sections; introduction, theories, methods, discussion, results, and further work. The plan is to start writing within week 6. The introduction, containing research questions, will be written first. Theories and methods will be next in the schedule and will be written simultaneously. Theories and methods will be initiated and take 4 to 5 weeks to complete. It is necessary to revise the previous sections until the delivery of the thesis.

Results and discussion will be started in the last few weeks before delivery. Before beginning to write it, it is crucial to have a product and relevant theories and methods. Several books and trustworthy resources should be read through this process to write this report scientifically. All references shall be proofread to ensure proper crediting. The abstract will be written at the end when the report's main sections are completed.

### 3.1.5 Presentations

Within week 13, the group must create a poster and give a presentation in English of the research questions for other groups. In addition, the final presentation will give a summary of the thesis. This presentation will be delivered as a video on 26 May at the latest.

## 3.2 Milestones

| When | What |
|---|---|
| January 27 | Submission of pre-project plan |
| Week 13 | Submission of poster and presentation in English |
| May 22 | Delivery of process documentation and product |
| May 26 | Delivery of the digital presentation |

# 4 Follow-up and Quality Assurance

## 4.1 Quality Assurance

Several measures should be taken to ensure the delivery of a high-quality thesis at the end. The main report should be given considerable attention as it is the most important part of the project. It is essential to be careful with citations and references. It is necessary to ensure that the references and citations are correct to avoid plagiarism by reviewing the list of references. Written words should be on academic level, especially words of the main report. It is not easy to differentiate between academic and slang words; therefore, online programs can help to write better and ensure that the text is academic. To write theories and methods, scientific books will have higher priority than websites and other resources. Also, theories and methods should be relevant and precise without including irrelevant and excess information.

Regarding coding, the group needs to learn the technology that will be used, and because of this, it will sometimes be essential to get help from someone with a good background in the technology used. The group will not hesitate to ask for help if needed. Writing tests is a vital part of writing good code. Testing will be written before pushing to ensure that the code does the necessary functions. As well as regular coding revisions during the project, it is crucial to ensure the coding is acceptable and easy to read.

How the group members work together is vital to get a high-quality thesis. To achieve this, the group will meet regularly by using teams and every Friday at the university to work together and inspire each other. Also, the group members will meet the supervisor and client regularly and send relevant meeting summons with the related cases we should discuss when it is necessary. The group would like to receive feedback and advice from the supervisor to write better meeting summons and meeting minutes. Involving the client and supervisor is essential throughout the working process. All process documents will be updated until delivering the thesis to ensure that all information is updated according to the changes the supervisor and the client ask for.

## 4.2 Reporting

The group and supervisor will meet digitally every two weeks, preferably every other Friday from 09.00 to 10.00. The minutes of meetings will be uploaded on Microsoft Teams the same day.

The students will try to meet on campus every Friday during the project. Christel is working at the farm from 08.00 to 14.00 Monday to Thursday and every other Sunday, which means that she will be working with the project evenings and weekends. Mona has children in kindergarten and will therefore work on the project during the daytime. The group can meet digitally after the children´s bedtime when necessary and will of course have dialogue about the project process throughout the day.

## 5. Risk Analysis

| | Risk | Description | Probability | Severity | Action to Minimize Risk |
|---|---|---|---|---|---|
| 1 | Internal Risks | | | | |
| 1.1 | The Conflict between the team members. | Reasons for conflicts are several, such as a team member not doing the work that should be done, lack of respect, and others. | Low | Catastrophic | Team members should talk to each other and try to find a solution to this problem. |
| 1.2 | Different languages and cultures. | Team members from different countries with different languages, which can sometimes lead to misunderstandin g | Low | Major | Team members should explain to others in the team the cultural differences and use English sometimes if it is necessary to explain unclear words. |
| 1.3 | Illness (own or family member) | The limited time force team members to work intensively. The whole project would be affected if one of the team members or their families were ill. | Low | Catastrophic | It is not easy to do something if the illness lasts for several weeks. However, if it was for a few days, the other members could do extra work to avoid problems with delivering. |

| 2 | External Risks | | | | |
|---|---|---|---|---|---|
| 2.1 | Time is limited | The thesis is conducted over one semester. Through this time, the scientific report should be written, in addition to coding and documents of process. | 🟥 High | 🟧 Major | All group members should regularly work independently and help each other if necessary. |
| 2.2 | Technologies' problems | Technologies used to accomplish the thesis are essential. Therefore, any problems with the version controls, such as GitLab, could lead to stopping all the projects. | 🟩 Low | 🟫 Catastrophic | It is difficult to control this problem. However, team members should be aware of the issues that technologies could raise. |

## References

1. *The California 4-H Youth Development Program – Directions for the Decade Ahead (*2003). Available at: https://4h.ucanr.edu/ (Retrieved 23 January 2023).

2. *4-H Pledge Head, Heart, Hands and Health* (n.d.). Available at: https://4-h.org/about/what-is-4-h/4-h-pledge/ (Retrieved 23 January 2023).

# 6. Appendixes

## 6.1 Schedule – Gantt Chart

The Gantt Chart is attached as a PDF in a zip folder.

## 6.2 Address List

**Client**
Stiftelsen Voll gård
Adress: Gamle Jonsvannsveien 1, 7049 Trondheim
Contact person: CEO Asbjørn Barlaup
Phone.: 996 99 299
E-mail: asbjorn@vollgard.no

**Supervisor**
Elise Klæbo Vonstad
Phone: 734 13 069
E-mail: elise.k.vonstad@ntnu.no

**Students**
Mona Mahmoud Mousa
Phone: 923 34 083
E-mail: monamm@stud.ntnu.no

Christel Mari Ossletten
Phone: 451 15 848
E-mail: chrisoss@stud.ntnu.no

## 6.3 Agreement Documents

The two documents are attached as PDF-files in the zip folder due to file size.

### 6.3.1 Arbeidskontrakt

Agreement to define/describe how the bachelor group will collaborate throughout the project.

### 6.3.2 Standardavtale

A tri-party agreement is  between NTNU (head of department and supervisor), the client, and the students.

## B Vision Document

123

# Utvikling av et bookingsystem for Voll gård
# Visjonsdokument

**Versjon 1.9**

# Revisjonshistorie

| Dato | Versjon | Beskrivelse | Forfatter |
|---|---|---|---|
| 01/02/23 | 1.0 | Startet på problem- og produktsammendrag | C. Ossletten |
| 03/02/23 | 1.1 | Funksjonelle og ikke-funksjonelle egenskaper | C. Ossletten |
| 06/02/23 | 1.2 | Fortsatte på problem- og produktsammendrag | C. Ossletten |
| 07/02/23 | 1.3 | Startet på brukere og interessenter | C. Ossletten |
| 22/02/23 | 1.4 | Brukere og interessenter | C. Ossletten |
| 24/04/23 | 1.5 | Funksjonelle og ikke-funksjonelle egenskaper | C. Ossletten |
| 25/04/23 | 1.6 | Introduksjon | C. Ossletten |
| 26/04/23 | 1.7 | Ferdigstilling | C. Ossletten |
| 29/04/23 | 1.8 | Ferdigstilling | C. Ossletten |
| 18/05/23 | 1.9 | Justeringer og gjennomlesning før levering | C. Ossletten |

# Innholdsfortegnelse

**1 Innledning** **4**

**2 Sammendrag problem og produkt** **5**

2.1 Problemsammendrag 5

2.2 Produktsammendrag 5

**3 Overordnet beskrivelse av interessenter og brukere** **6**

3.1 Oppsummering interessenter 6

3.2 Oppsummering brukere 7

3.2.1 Administratorer 7

3.2.2 Besøkende 7

3.3 Brukermiljøet 7

3.4 Sammendrag av brukernes behov 7

3.5 Alternativer til produktet 8

**4 Produktoversikt** **8**

4.1 Produktets rolle i brukermiljøet 8

4.2 Forutsetninger og avhengigheter 8

**5 Produktets funksjonelle egenskaper** **9**

5.1 Tabell over produktets funksjonelle egenskaper og beskrivelser av disse 9

**6 Ikke-funksjonelle egenskaper og andre krav** **13**

6.1 Kvalitetsattributter 13

6.1.1 Brukervennlighet 13

6.1.2 Pålitelighet 13

6.1.3 Tilgjengelighet 13

6.1.4 Sikkerhet 13

6.1.5 Vedlikehold 13

6.1.6 Skalerbarhet 14

6.1.7 Personvern 14

6.2 Begrensninger 14

6.2.1 Tid 14

6.2.2 Faglige ressurser innad i bedriften 14

**7 Referanser** **14**

# 1 Innledning

Visjonsdokumentet blir skrevet som en del av dataingeniørstudiet ved NTNU og 3.års-faget IDATT2990 *Bacheloroppgave* (del 2 av 2). Tittelen på bacheloroppgaven er *Utvikling av et bookingsystem for Voll gård*, med Voll gård som oppdragsgiver.

Hensikten med visjonsdokumentet er å gi en oversikt over oppgaven og hvordan produktet er tenkt utviklet. Det inneholder et sammendrag av dagens problem og produktet som skal utvikles, beskrivelse av interessenter og brukere, samt produktoversikt, funksjonelle og ikke-funksjonelle egenskaper og krav.

Voll gård er eid av Stiftelsen Voll gård, og er en besøks- og 4H-gård som ligger nær Trondheim sentrum.

4H er et nettverk av ungdomsorganisasjoner med forskjellige prosjekter innen blant annet helse, forskning, dyrehold og jordbruk. Det er et positivt miljø der ungdommer får veiledning fra voksne mentorer og oppfordres til å ta på seg lederroller innad i organisasjonen. (4-H u.d.)

Gården er åpen for privatpersoner og har over 50.000 årlige besøkende som ønsker å bruke arenaen og hilse på dyrene. Hovedformålet er å tilby undervisning knyttet til norsk husdyrhold og jordbruk til skoler og barnehager. Dette muliggjøres ved hjelp av støtte fra Trondheim kommune.

Gården tilbyr ulike opplevelser og tjenester, som per dags dato bookes over telefonsamtaler og e-postveksling. Dagens løsning er utdatert og tidkrevende, og målet med prosjektet er derfor å utvikle et bookingsystem i form av en webapplikasjon. Hensikten med dette vil være å effektivisere bookingprosessen for besøkende og ikke minst arbeidshverdagen til oppdragsgiver.

# 2 Sammendrag problem og produkt

## 2.1 Problemsammendrag

Som besøksgård tilbyr Voll gård en rekke opplevelser og tjenester til sine kunder i tillegg til den vanlige gårdsdriften. Dette medfører mye logistikkarbeid, da alt skal inn i en kalender og gå opp med resten av arbeidet som foregår på gården. Kapasiteten på disse opplevelsene og tjenestene varierer naturligvis gjennom året.

Å gjennomføre bookinger på gården med dagens løsning er innviklet og ikke minst tidkrevende for både oppdragsgiver og kunder. En av årsakene til at det tar lang tid er at kundene ikke finner den informasjonen de er ute etter samlet på ett sted. Dette medfører mye kommunikasjon frem og tilbake før alt av spørsmål er besvart. Det blir også noe venting på svar innimellom for både oppdragsgiver og deres kunder, og begge partene må bli enige om en dato som passer.

En vellykket løsning med et moderne bookingsystem vil inneholde all informasjonen kundene trenger for å kunne gjennomføre bookingen ut fra ledige datoer i en kalender, og effektivisere bookingprosessen betraktelig.

## 2.2 Produktsammendrag

Produktet er et bookingsystem som skal utvikles for Voll gård, som har behov for et velfungerende og effektivt system der besøkende kan booke ulike opplevelser og tjenester på gården.

I tillegg til å muliggjøre booking av bursdagsfeiringer, opphold på smådyrpensjonat, skole- og barnehagebesøk, skal det utvikles en administratorside der det kan opprettes brukere for de i bedriften som trenger tilgang til bookingoversikten. Administratorer skal kunne endre og kansellere innkomne bookinger.

Dette i motsetning til dagens system hvor bookinger foregår over telefonsamtaler og flere ulike e-postadresser. Kommunikasjonen går for det første tregt, og når alt av spørsmål endelig er besvart og datoen spikret, sender oppdragsgiver ut en bookingbekreftelse. Denne skrives fra en mal i Google Dokumenter, som blant annet inneholder alt av bookingdetaljer og gårdens vilkår for bookingen. Etter at oppdragsgiver har sendt bekreftelsen, må kunden sende en mail tilbake igjen for å akseptere vilkårene og bekrefte at alt stemmer. Deretter er bookingen klar og kan fylles inn i et Google Regneark som er koblet til de ansattes Google Kalender.

# 3 Overordnet beskrivelse av interessenter og brukere

## 3.1 Oppsummering interessenter

| **Hvem** | **Utdypende beskrivelse** | **Rolle under utviklingen** |
|---|---|---|
| Oppdragsgiver | Oppdragsgiver er Voll gård, representert av daglig leder Asbjørn Barlaup og Christel Ossletten. | Oppdragsgivers rolle under utviklingen er å gi utviklerne et bilde av dagens problem og ikke minst hvilke behov og krav som stilles til produktet.<br><br>Det vil være jevnlig kontakt med oppdragsgiver for å orientere om fremdrift og drøfting av produktet til nå. |
| Utviklere | Gruppen består av to 3.årsstudenter ved dataingeniørstudiet på Institutt for datateknologi og informatikk (IDI) ved NTNU, Mona Mousa og Christel Ossletten. | Utviklerne skal i løpet av våren utvikle et bookingsystem for oppdragsgiver som en del av sin bacheloroppgave.<br><br>*Christel er i tillegg ansatt i Stiftelsen Voll gård og ansvarlig for bookinger, og vil derfor samtidig fungere som en representant fra oppdragsgiveren.* |
| Veileder | Representant fra NTNU og gruppens veileder er Elise Klæbo Vonstad. | Veileders rolle under utviklingen er i hovedsak å veilede og rådgi studentene under skrivingen. |
| Besøkende (kunder) | I hovedsak foreldre, dyreeiere, lærere og pedagogiske ledere i barnehager. | Besøkendes rolle under utviklingen er å delta i brukertesting. |

## 3.2 Oppsummering brukere

### 3.2.1 Administratorer

Ansatte i bedriften regnes som brukere av løsningen, da de trenger oversikt over bookingene og mulighet til å endre eller kansellere disse gjennom en administratorside i webapplikasjonen. I tillegg til en administratorbruker for bookingansvarlig, vil det være nødvendig å kunne opprette flere brukere da forskjellige ansatte har sine ansvarsområder.

### 3.2.2 Besøkende

I tillegg til de ansatte, er bedriftens besøkende brukere av løsningen. Webapplikasjonen lages for at besøkende skal kunne finne den informasjonen de trenger og fylle ut nødvendig informasjon for å booke et opphold eller en tjeneste på gården. I all hovedsak er dette foreldre, dyreeiere, lærere og pedagogiske lærere i barnehager. Det er ønskelig at potensielle besøkende skal være med på brukertesting under utviklingen.

## 3.3 Brukermiljøet

Bookingsystemet som utvikles skal være tilgjengelig fra oppdragsgivers sosiale medier og hjemmeside. Dagens hjemmeside er svært utdatert og laget i publiseringssystemet Wordpress, noe som gjør den lite fleksibel og vanskelig å implementere noe inn i. Oppdragsgiver har som mål å få oppgradert hjemmesiden etterhvert, men enn så lenge utvikles bookingsystemet i dette prosjektet som en egen webapplikasjon.

Det er viktig at brukergrensesnittet er lett navigerbart og forståelig, spesielt med tanke på kunder som har booket tjenester i årevis og er fornøyd med, og vant til, den nåværende løsningen. Oppdragsgiver ønsker å bruke moderne farger og fine bilder til å fange kundenes oppmerksomhet. I tillegg utvikles systemet med tanke på å kunne videreutvikles, da gården tilbyr utallige opplevelser, tjenester og produkter som etterhvert bør implementeres i samme system dersom det skal være et poeng i å bruke det.

## 3.4 Sammendrag av brukernes behov

Oppdragsgiver har behov for å effektivisere bookinger og frigjøre de ansattes tid til andre oppgaver på gården. I systemet må det være mulig å opprette administratorbrukere der ansatte fra gården kan logge seg inn for å få oversikt over bookingene framover, samt endre eller kansellere bookinger ved behov. Bookingoversikten skal kunne sorteres og filtreres.

Besøkende skal kunne bruke webapplikasjonen til å finne ledige datoer og den informasjonen de trenger for å, i første omgang, kunne booke feiring av barnebursdag, smådyrpensjonat, skole- og barnehagebesøk. Det skal være lett å finne kontaktinformasjonen til oppdragsgiver dersom noe er uklart. Besøkende skal kunne akseptere vilkår og bekrefte bookingen, for deretter å få tilsendt bookingdetaljer på e-post.

Under utviklingen av systemet må det tas hensyn til at kundene legger inn sensitiv informasjon i form av personopplysninger når de booker tjenester på gården. Det er svært viktig at disse ivaretas og ikke kan komme på avveie.

## 3.5 Alternativer til produktet

Et alternativ til gruppens produkt, er dagens løsning med booking over e-post og telefonsamtaler. Selv om dette gjerne styrker forholdet mellom bedriften og besøkende, og gir besøkende en følelse av å bli personlig ivaretatt, kan det fort virke mot sin hensikt dersom responstiden er lang på grunn av mange henvendelser, vanskeligheter med å finne en ledig dato og så videre. I tillegg er dette en tungvint og umoderne løsning.

Et annet alternativ er å bruke allerede eksisterende løsninger. Et eksempel er Zoho Bookings, som er en nettbasert programvare som håndterer bookinger og hva enn det måtte innebære. Det kan for eksempel være kalendersynkronisering, avbestilling og betaling. I tillegg kan bedriften selv velge tilgjengelige tider og legge inn begrensninger for hvor lang eller kort tid i forveien en booking kan finne sted. (Zoho Corporation, u.d.)

# 4 Produktoversikt

## 4.1 Produktets rolle i brukermiljøet

For oppdragsgivers del, vil produktet effektivisere bookinghåndtering og i stor grad lette arbeidsmengden. Dagens system med e-post og telefonsamtaler er svært utdatert, og en ny løsning vil ta seg godt ut. I tillegg vil både oppdragsgiver og besøkende tjene på at bookingen skjer kjapt uten å måtte vente på svar fra den ene eller den andre parten.

All nødvendig informasjon, samt svar på det bedriften oftest får spørsmål om, vil være opplyst om på den første siden i bookingprosessen. Dette vil trolig være til stor hjelp for besøkende som ønsker å booke en tjeneste, og for noen er det kanskje avgjørende for om bookingen gjennomføres eller ikke. Dette kan sammenlignes med kunder som går videre til neste butikk dersom prisen på en vare ikke er oppgitt eller de ikke finner riktig størrelse, i stedet for å spørre betjeningen om hjelp.

## 4.2 Forutsetninger og avhengigheter

Utviklerne i prosjektet trenger tilgang til bilder og nødvendig informasjon om de ulike tjenestene som gården tilbyr for å kunne utvikle webapplikasjonen.

For å bruke webapplikasjonen trenger besøkende og administratorer tilgang på en PC, mobil eller nettbrett med internettilgang og en nyere nettleser. Begge partene må selvsagt også ha en viss oversikt over deres fremtidige tilgjengelighet.

Besøkende trenger ikke å opprette en bruker for å legge inn bookinger, men administratorer må ha en bruker for å få oversikt over bookingene og tilgang til sensitive opplysninger som for eksempel kontaktinformasjonen til de besøkende.

# 5 Produktets funksjonelle egenskaper

## 5.1 Tabell over produktets funksjonelle egenskaper og beskrivelser av disse

| | **Funksjonell egenskap** | **Beskrivelse** |
|---|---|---|
| | Detaljer | |
| 1 | Header | Skal inneholde logo som er linket tilbake til forsiden av webapplikasjonen til venstre, og knapp for admininnlogging til høyre. |
| 2 | Meny | Menyen ligger under headeren og inneholder knapper til de respektive bookingsidene (bursdagsfeiring, smådyrpensjonat, skole- og barnehagebesøk). |
| 3 | Forside/informasjon | Inneholder informasjon om gården og de bookingmulighetene applikasjonen tilbyr. |
| 4 | Footer | Skal inneholde adresse, kontaktinformasjon (e-post og telefon), samt linker til sosiale media. |
| | Administrator | |
| 5 | Logge inn | Personer som håndterer bookinger, heretter kalt *administratorer*, skal kunne logge inn på en egen administratorside i webapplikasjonen. |
| 6 | Opprette ny bruker | Påloggede administratorer kan opprette nye administratorbrukere. |
| 7 | Bookingoversikt | Oversikt over alle innkomne bookinger. |
| 8 | Søk, sortering og filtrering | Bookingene i oversikten skal kunne sorteres og filtreres. Et søkefelt der man f.eks. kan søke på navn eller andre detaljer skal implementeres. |
| 9 | Endre bookinger | Administrator skal kunne endre bookingdetaljer. |

| | | |
|---|---|---|
| 10 | Kansellere bookinger | Administrator skal kunne kansellere bookinger. |
| 11 | Logge ut | Administrator skal kunne logge ut. |
| | Booke bursdagsfeiring | |
| 12 | Menyvalg | Brukeren klikker seg inn på siden for booking av bursdagsfeiring via en knapp i menyen øverst på siden. |
| 13 | Informasjonsside | Etter å ha klikket seg inn kommer brukeren til en side med nødvendig informasjon (hvordan opplegget er, varighet, mulige tilvalg, priser osv.). |
| 14 | Velge dato | Brukeren velger ønsket dato ut fra en kalender nederst på siden. |
| 15 | Velge tidspunkt | Brukeren velger ønsket tidspunkt for feiringen før den klikker seg videre til neste side. |
| 16 | Bookingdetaljer | Brukeren fyller ut nødvendige detaljer for å gjennomføre bookingen. Her skal det også være mulig å krysse av for tilvalg (hesteaktivitet, eventuelt hvilken type hesteaktivitet, og leie av bålhuset), samt skrive en kommentar. |
| 17 | Oppsummering | På neste side får brukeren en oppsummering av bookingen og en opplisting av bedriftens vilkår. Det skal være mulig å klikke seg tilbake til forrige side for å endre opplysninger. |
| 18 | Bekrefte bookingen | Brukeren er nødt til å huke av for at opplysningene stemmer og at bedriftens vilkår godtas for å kunne bekrefte bookingen. |
| 19 | Tilbakemelding | Brukeren får tilbakemelding om at bookingen er mottatt og at bekreftelse sendes til oppgitt e-postadresse. |
| 20 | Bekreftelse på e-post | Bookingbekreftelse sendes til oppgitt e-postadresse. Skal inneholde de samme opplysningene som i oppsummeringen, samt annen viktig informasjon. |
| | Booke smådyrpensjonat | |

| | | |
|---|---|---|
| 21 | Menyvalg | Brukeren klikker seg inn på siden for booking av smådyrpensjonat via en knapp i menyen øverst på siden. |
| 22 | Informasjonsside | Etter å ha klikket seg inn kommer brukeren til en side med nødvendig informasjon (generell informasjon, mulige tilvalg, priser osv.). |
| 23 | Velge datoer | Brukeren velger leverings- og hentedato ut fra en kalender nederst på siden. |
| 24 | Bookingdetaljer | Brukeren fyller ut nødvendige detaljer for å gjennomføre bookingen. Her skal det også være mulig å krysse av for tilvalg (kloklipp og lånebur), skrive en kommentar, samt legge til flere dyr ved å trykke på en knapp. |
| 25 | Oppsummering | På neste side får brukeren en oppsummering av bookingen og en opplisting av bedriftens vilkår. Det skal være mulig å klikke seg tilbake til forrige side for å endre opplysninger. |
| 26 | Bekrefte bookingen | Brukeren er nødt til å huke av for at opplysningene stemmer og at bedriftens vilkår godtas for å kunne bekrefte bookingen. |
| 27 | Tilbakemelding | Brukeren får tilbakemelding om at bookingen er mottatt og at bekreftelse sendes til oppgitt e-postadresse. |
| 28 | Bekreftelse på e-post | Bookingbekreftelse sendes til oppgitt e-postadresse. Skal inneholde de samme opplysningene som i oppsummeringen, samt annen viktig informasjon. |
| | Booke skolebesøk | |
| 29 | Menyvalg | Brukeren klikker seg inn på siden for booking av skolebesøk via en knapp i menyen øverst på siden. |
| 30 | Informasjonsside | Etter å ha klikket seg inn kommer brukeren til en side med nødvendig informasjon. |
| 31 | Velge dato | Brukeren velger ønsket dato for besøket ut fra en kalender nederst på siden. |

| | | |
|---|---|---|
| 32 | Bookingdetaljer | Brukeren fyller ut nødvendige detaljer for å gjennomføre bookingen. |
| 33 | Oppsummering | På neste side får brukeren en oppsummering av bookingen. Det skal være mulig å klikke seg tilbake til forrige side for å endre opplysninger. |
| 34 | Bekrefte bookingen | Brukeren må lese gjennom oppsummeringen og klikke på en knapp for å bekrefte bookingen. |
| 35 | Tilbakemelding | Brukeren får tilbakemelding om at bookingen er mottatt og at bekreftelse sendes til oppgitt e-postadresse. |
| 36 | Bekreftelse på e-post | Bookingbekreftelse sendes til oppgitt e-postadresse. Skal inneholde de samme opplysningene som i oppsummeringen, samt annen viktig informasjon. |
| | Booke barnehagebesøk | |
| 37 | Menyvalg | Brukeren klikker seg inn på siden for booking av barnehagebesøk via en knapp i menyen øverst på siden. |
| 38 | Informasjonsside | Etter å ha klikket seg inn kommer brukeren til en side med nødvendig informasjon. |
| 39 | Velge dato | Brukeren velger ønsket dato for besøket ut fra en kalender nederst på siden. |
| 40 | Bookingdetaljer | Brukeren fyller ut nødvendige detaljer for å gjennomføre bookingen. |
| 41 | Oppsummering | På neste side får brukeren en oppsummering av bookingen. Det skal være mulig å klikke seg tilbake til forrige side for å endre opplysninger. |
| 42 | Bekrefte bookingen | Brukeren må lese gjennom oppsummeringen og klikke på en knapp for å bekrefte bookingen. |
| 43 | Tilbakemelding | Brukeren får tilbakemelding om at bookingen er mottatt og at bekreftelse sendes til oppgitt e-postadresse. |
| 44 | Bekreftelse på e-post | Bookingbekreftelse sendes til oppgitt e-postadresse. Skal inneholde de samme |

| | | |
|---|---|---|
| | | opplysningene som i oppsummeringen, samt annen viktig informasjon. |

# 6 Ikke-funksjonelle egenskaper og andre krav

## 6.1 Kvalitetsattributter

En av kategoriene innen ikke-funksjonelle egenskaper er *kvalitetsattributter*, som vil si de egenskapene som bestemmer den generelle kvaliteten i systemet.

### 6.1.1 Brukervennlighet

For oppdragsgiver er brukervennlighet en av de viktigste egenskapene til systemet. Systemet må være intuitivt, ryddig og lett å forstå seg på. Det ble blant annet tatt en avgjørelse om at kunder ikke skal måtte opprette brukere for å legge inn bookinger, da dette fort oppleves som tungvint for de som er vant til å bare kunne ringe inn og booke besøk på kort varsel. Dette gjelder spesielt for barnehager og skoler. Det hadde i tillegg fort blitt opprettet svært mange brukere, da det ikke alltid er samme person som legger inn bookinger fra samme skole eller barnehage.

### 6.1.2 Pålitelighet

Handler om at systemet skal være pålitelig og oppfylle brukernes krav. Dette kan for eksempel være at bookinger faktisk blir registrert og at systemet skal fungere slik som det er tenkt.

### 6.1.3 Tilgjengelighet

Handler om at systemet skal være tilgjengelig ved behov og ha lite nedetid. Dette skal blant annet sikres ved å hoste en server gjennom Firebase.

### 6.1.4 Sikkerhet

Det er viktig at systemet er sikkert da besøkende legger inn sensitive kontaktopplysninger ved booking. Firebase fungerer som en vert for SSL (Secure Sockets Layer), som typisk krypterer forbindelsen mellom klient og server, og forhindrer datatyveri eller manipulasjon på nettverkslaget. (Singh og Tanna, 2018, s. 170) Dette, i tillegg til at utviklerne skriver tester, er med på å sikre systemet.

### 6.1.5 Vedlikehold

Handler om at systemet skal være mulig å oppdatere og intuitivt dersom vedlikehold er nødvendig. Dette er spesielt viktig siden tanken er at systemet bør videreutvikles før bruk, og må kunne vedlikeholdes av andre i fremtiden for å fungere. For dette er det blant annet viktig med god dokumentasjon og kommentarer på kodingen.

### 6.1.6 Skalerbarhet

Skalerbarhet handler om at systemet skal kunne takle en økende mengde bookinger.

### 6.1.7 Personvern

Dette punktet handler om at all sensitiv informasjon som legges inn av kunder og lagres i systemet, i all hovedsak kontaktinformasjon, skal forvares på en sikker og trygg måte.

## 6.2 Begrensninger

Den andre kategorien innen ikke-funksjonelle krav er *begrensninger*. Det vil si de begrensningene som er pålagt systemet. (Visure Solutions, Inc. u.d.)

### 6.2.1 Tid

Studentene har 500 timer hver, det vil si 1000 timer totalt, til rådighet i løpet av prosjektet. Lite tid er den største begrensningen, spesielt siden mye av tiden går med til selvlæring og rapportskriving. Systemet bør derfor videreutvikles før det kan tas i bruk.

### 6.2.2 Faglige ressurser innad i bedriften

Bedriften har ingen utviklere innad i bedriften, noe som betyr at alt må gjøres fra start og at studentene ikke har noen i bedriften de kan spørre om råd ved behov.

# 7 Referanser

Singh, H. og Tanna, M. (2018) *Serverless Web Applications with React and Firebase: Develop Real-time Applications for Web and Mobile Platforms.* 1.utg. Packt Publishing.

Visure Solutions, Inc. (u.d.) *Ikke-funksjonelle krav: Typer, eksempler og tilnærminger.* Tilgjengelig fra: https://visuresolutions.com/no/requirements-management-traceability-guide/non-functional-requirements (Hentet: 26.februar 2023).

Zoho Corporation (u.d.) *Features.* Tilgjengelig fra: https://www.zoho.com/bookings/features.html (Hentet: 14.april 2023).

4-H (u.d.) *What is 4-H?* Tilgjengelig fra: https://4-h.org/about/ (Hentet: 20.mars 2023).

## C Requirements Document

123

# Developing a Booking System for Voll gård
# Requirements Document

**Version 1.3**

VOLL GÅRD
HELE BYENS BONDEGÅRD

NTNU
Kunnskap for en bedre verden

## Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 16/02/23 | 1.0 | Starting user stories, wireframes, and domain models. | C. Ossletten, M. Mousa |
| 18/04/23 | 1.1 | Updating user stories, domain models, and finishing sequence diagrams. | M. Mousa |
| 10/05/23 | 1.2 | Adding images and cleaning/restructuring the document. | C.Ossletten |
| 19/05/23 | 1.3 | Final adjustments. | C.Ossletten |

# Table of Contents

**1 Introduction** **4**
**2 User Stories** **4**
**3 Domain Model** **7**
**4 Sequence Diagram** **8**
4.1 Birthday Celebration 8
4.2 Animal Boarding 9
4.3 School Visit 10
4.4 Kindergarten Visit 11
4.5 Admin 12
4.5.1 Login Page 12
4.5.2 Sign Up Page 13
4.5.3 Admin Dashboard 14
**5 Prototyping** **15**
5.1 Wireframes 15
5.1.1 Home Page 15
5.1.2 Birthday Celebration 16
5.1.3 Animal Boarding 19
5.1.3 School Visit 22
5.1.4 Kindergarten Visit 25
5.1.5 Admin Page 28

# 1 Introduction

The requirement document is written as a part of the computer engineering program at NTNU and the 3rd year course IDATT2900 Bachelor's Thesis (part 2 of 2). The title of the bachelor's thesis is "Development of a Booking System for Voll gård", with Voll gård as the contractor.

The requirement document is written during the planning phase to develop the booking system, and it is used as a blueprint for the team members to design and build the website. The purpose of this document is to outline the specific features, functions, and capabilities the booking system must have to meet the needs of its users.

The document contains user stories that give the details of users' requirements and their scenarios and domain models that define how the several components work together. In addition, it includes prototypes to provide a visual representation of the product's intended design and functionality.

# 2 User Stories

In the user stories listed below, users are visitors wanting to book a service on the farm, while administrators are employees at Voll gård with access to the booking data.

---

**As a user**, I want to be able to get information about birthday celebrations so that I can book a birthday event on my desired date and time.

**Scenario:** Book birthday celebration on the farm.

When I select a specific date after reading the necessary information,
I should be able to enter my contact information (name, email, and telephone number), choose a time, be able to choose horse activity and bonfire as an add-on, specify the type of horse activity, number of children, age of the child, and add a comment if needed.
Then I should be able to see the entered booking details, as well accepting terms and conditions, before submitting it.
I will then receive feedback that the booking has been registered, and receive an email with the details.

---

**As a user**, I want to get information about the animal boarding service so that I can book a stay for my animal on my desired date.

**Scenario:** Book stay at animal boarding.

When I select a specific date for delivery after reading the necessary information,
I should be able to enter my contact information (name, email, and telephone number), number of nights for the stay, select type of animal, gender and whether it is

neutered/sterilized, and choose other services such as claw clip, borrow cage, and add a comment if needed.
Then I should be able to see the entered booking details, as well accepting terms and conditions, before submitting it.
I will then receive feedback that the booking has been registered, and receive an email with the details.

---

**As a user**, I want to get information about school visits so that I can book a school visit for my class on my desired date.

**Scenario:** Book a school visit on the farm.

When I select a specific date after reading the necessary information,
I should be able to enter the school's name, class level, number of adults, number of children, desired educational subject, contact information (name, email, and telephone number), and a comment if needed.
Then I should be able to see the entered booking details before submitting.
I will then receive feedback that the booking has been registered, and receive an email with the details.

---

**As a user,** I want to get information about kindergarten visits so that I can book a kindergarten visit for my class on my desired date.

**Scenario:** Book a kindergarten visit on the farm.

When I select a specific date after reading the necessary information,
I should be able to enter the name of the kindergarten, name of the department, number of adults, number of children, whether an educational visit is desirable and, if so, on which subject, contact information (name, email, telephone), and a comment if needed.
I will then receive feedback that the booking has been registered, and receive an email with the details.

---

**As a user**, I want to be able to navigate to the previous page so that I can edit booking details before submitting.

**Scenario:** Modify booking information before submitting it.

When I click on the "Tilbake" button on the booking summary page,
I should be able to navigate to the previous page and edit booking details.
Then I should be able to submit new booking data.
I will then receive feedback that the booking has been registered, and receive an email with the details.

---

**As an administrator**, I want to be able to create an admin account so that I can log in to the admin page.

**Scenario:** Register new admin account.

When I click on the “Opprett konto” button,
I should be able to enter a name, password, confirmed password, and code,
And if the code is correct, I have registered a new user and can navigate to the login page.

---

**As an administrator**, I want to be able to log in to the admin dashboard.

**Scenario:** Log in as administrator.

When I click on the “Admininnlogging” button,
I should be able to enter email address and password,
And if the password is correct, after clicking the “logg inn” button, I am sent to the admin dashboard with an overview of all bookings.

---

**As an administrator**, I want to be able to filter, sort and search bookings based on different data so that I can find specific bookings.

**Scenario:** Filter, sort and search through bookings.

When I select a specific booking list,
I should be able to select a specific date, or enter details such as email or telephone number,
And I will get an overview of all suitable bookings.

---

**As an administrator,** I want to be able to modify booking details so that I can help my users to update their booking information.

**Scenario:** Modify the booking information.

When I double-click on a particular field of booking data on a specific booking,
I should be able to change the content of the field with new input,
And the booking details and booking list will get updated with this new information.

---

**As an administrator**, I want to be able to delete specific bookings.

**Scenario:** Delete a specific booking.

When I click on the trash icon,
I should be able to confirm deleting the booking,
And I will delete the specific booking from the selected booking list.

---

# 3 Domain Model

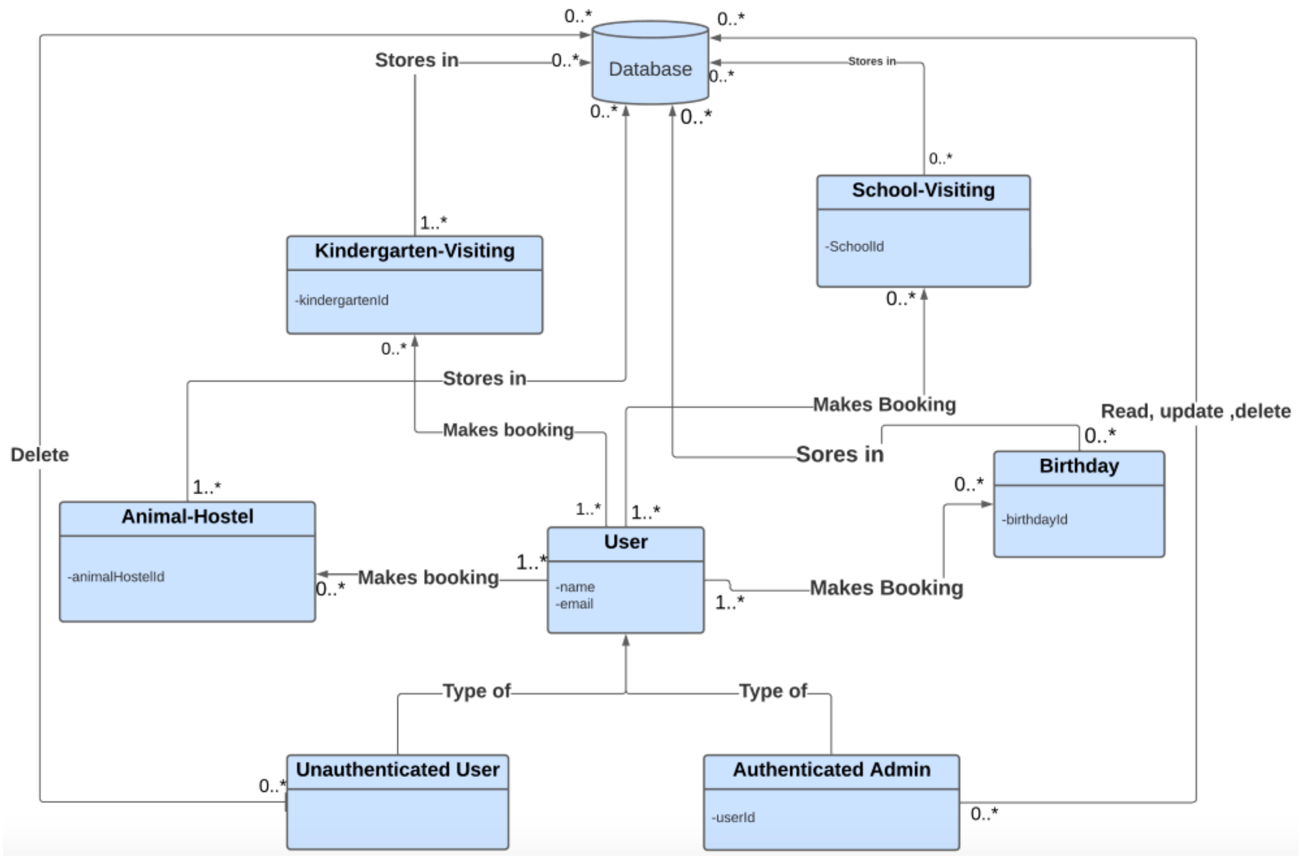


Figure 1: A user entity can make zero to many birthday, animal hostel, kindergarten, or school entities, and all booking data stored in the database.

# 4 Sequence Diagram

## 4.1 Birthday Celebration

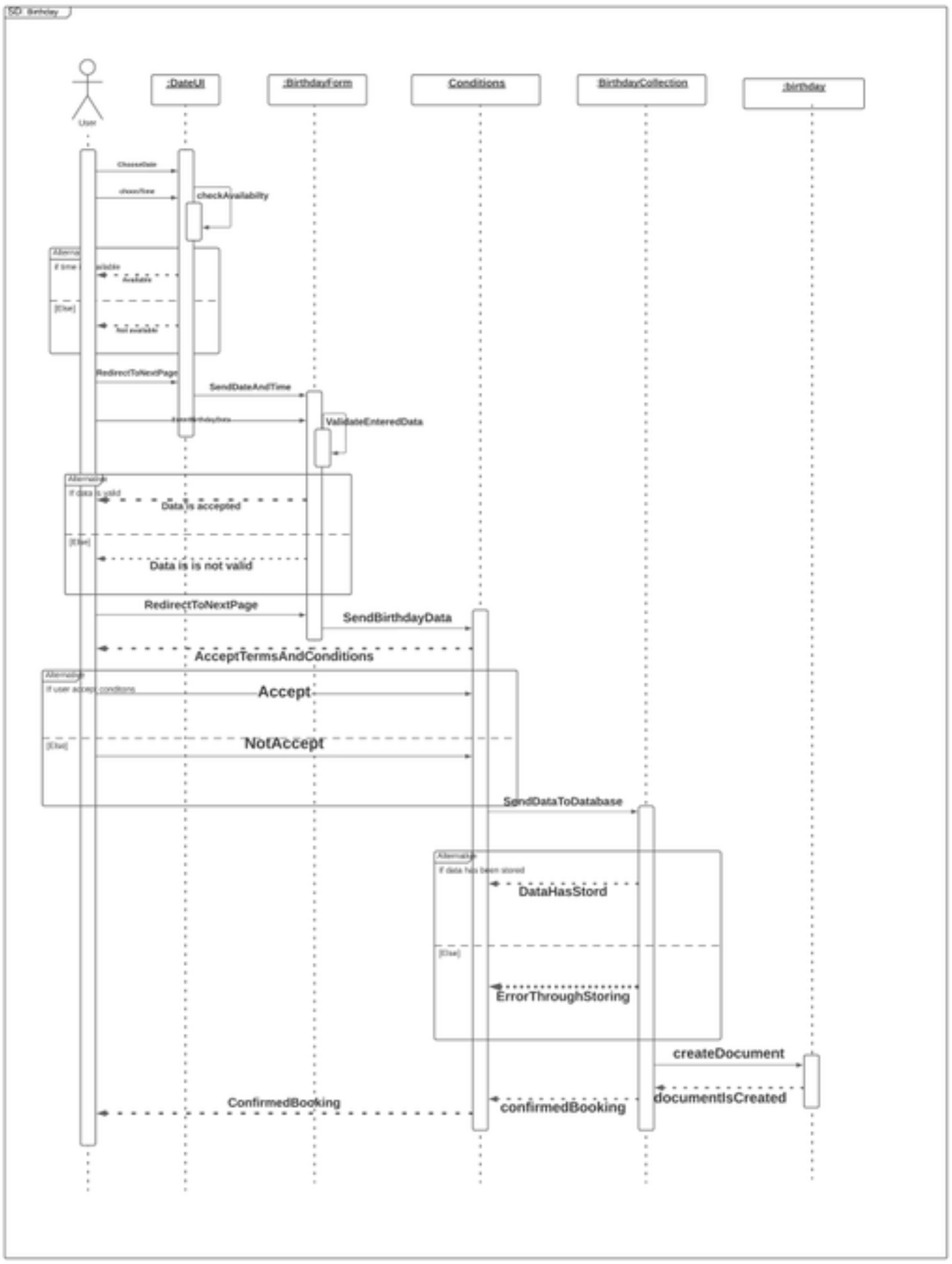


Figure 2: Birthday sequence diagram

## 4.2 Animal Boarding

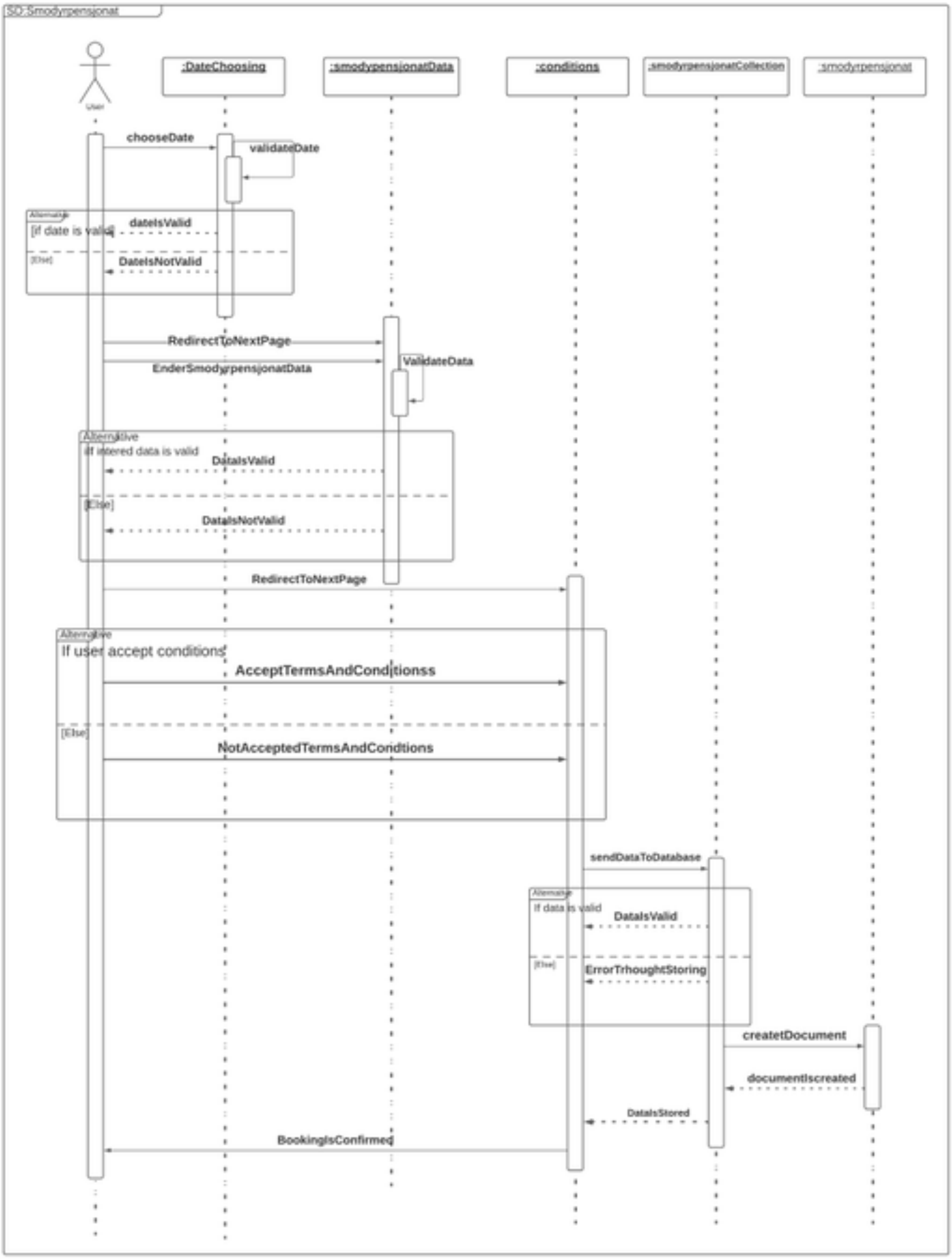


Figure 3: Animal boarding sequence diagram

## 4.3 School Visit

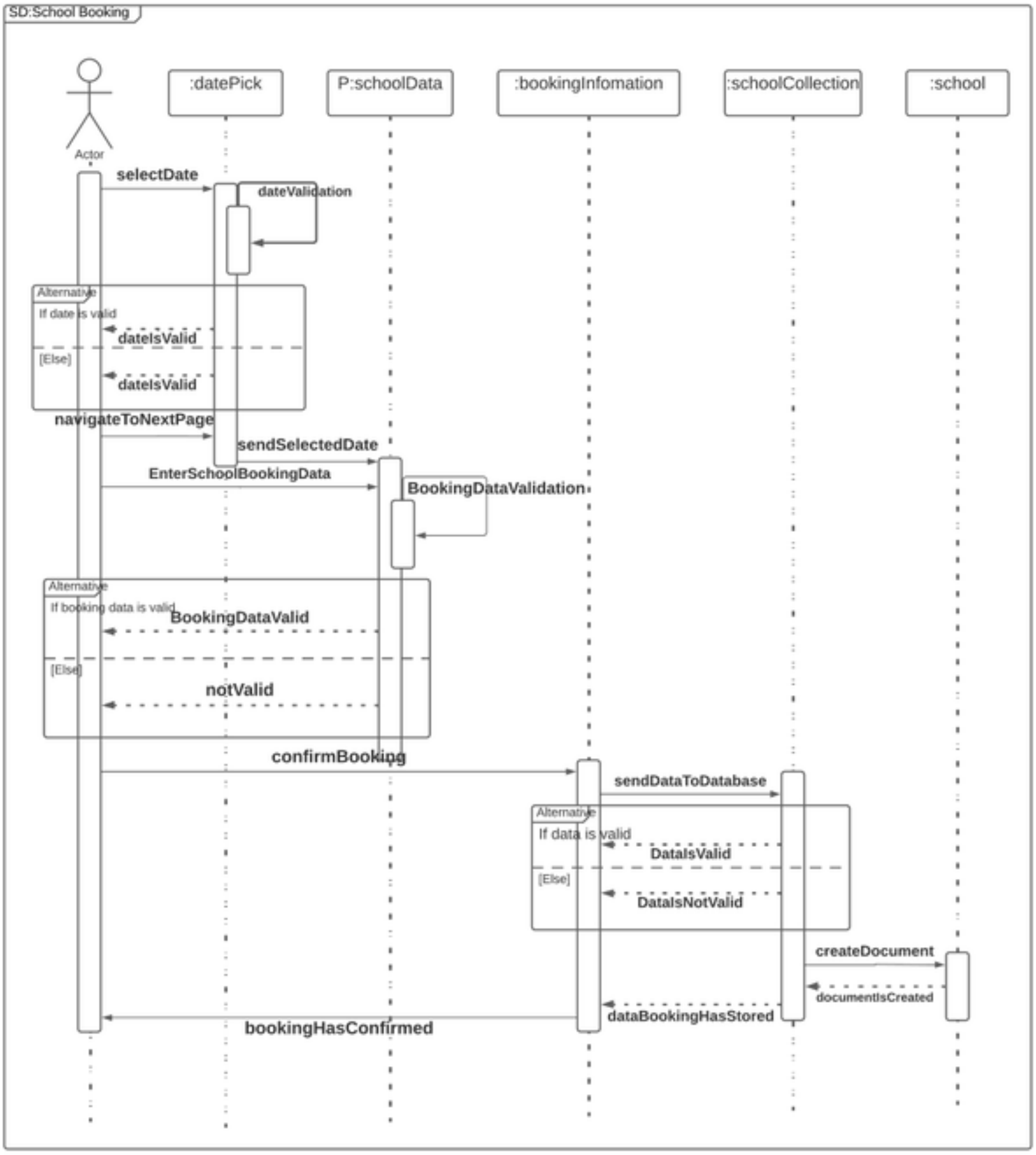


Figure 4: School visit sequence diagram

## 4.4 Kindergarten Visit

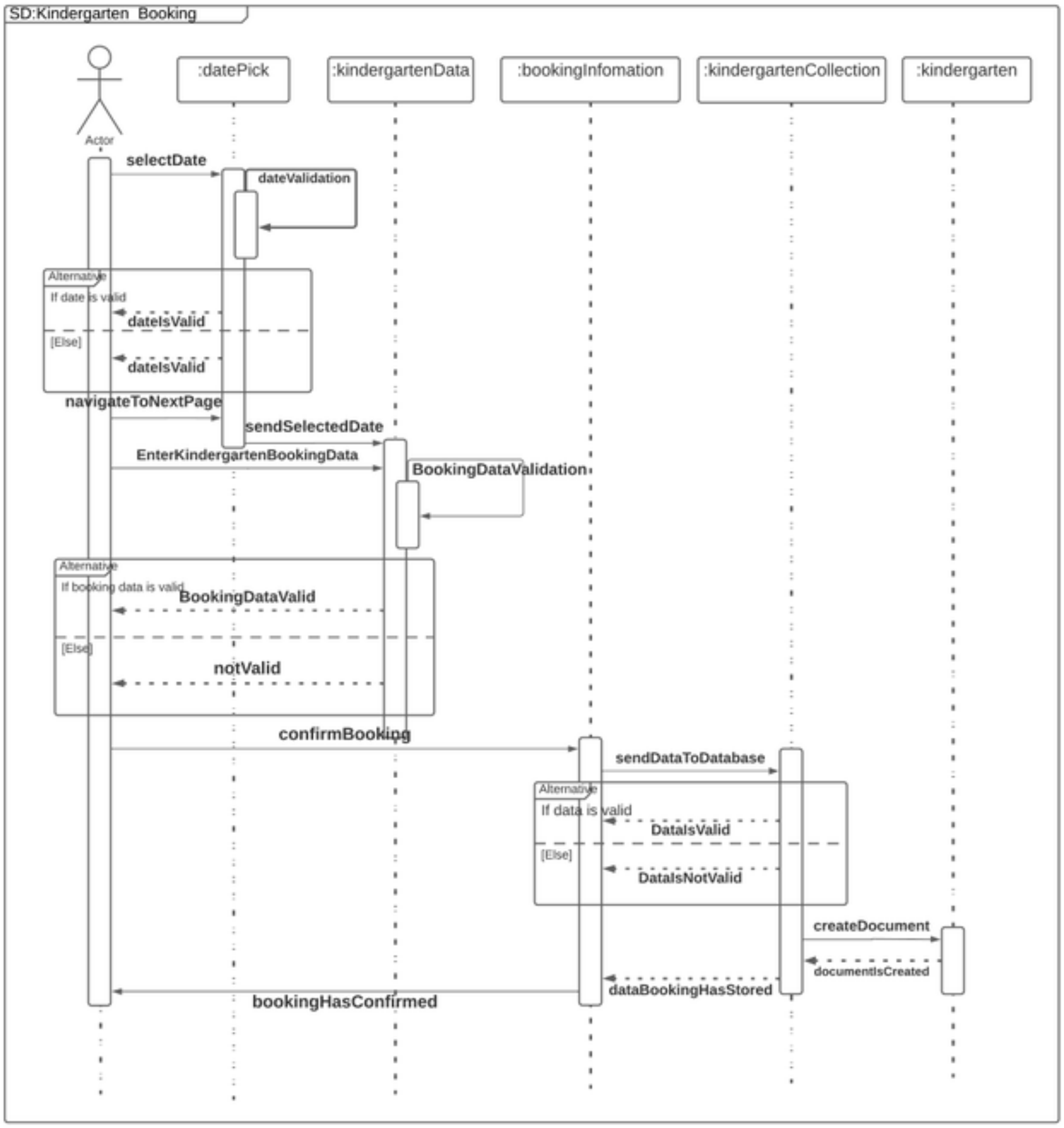


Figure 5: Kindergarten visit sequence diagram

## 4.5 Admin

### 4.5.1 Login Page

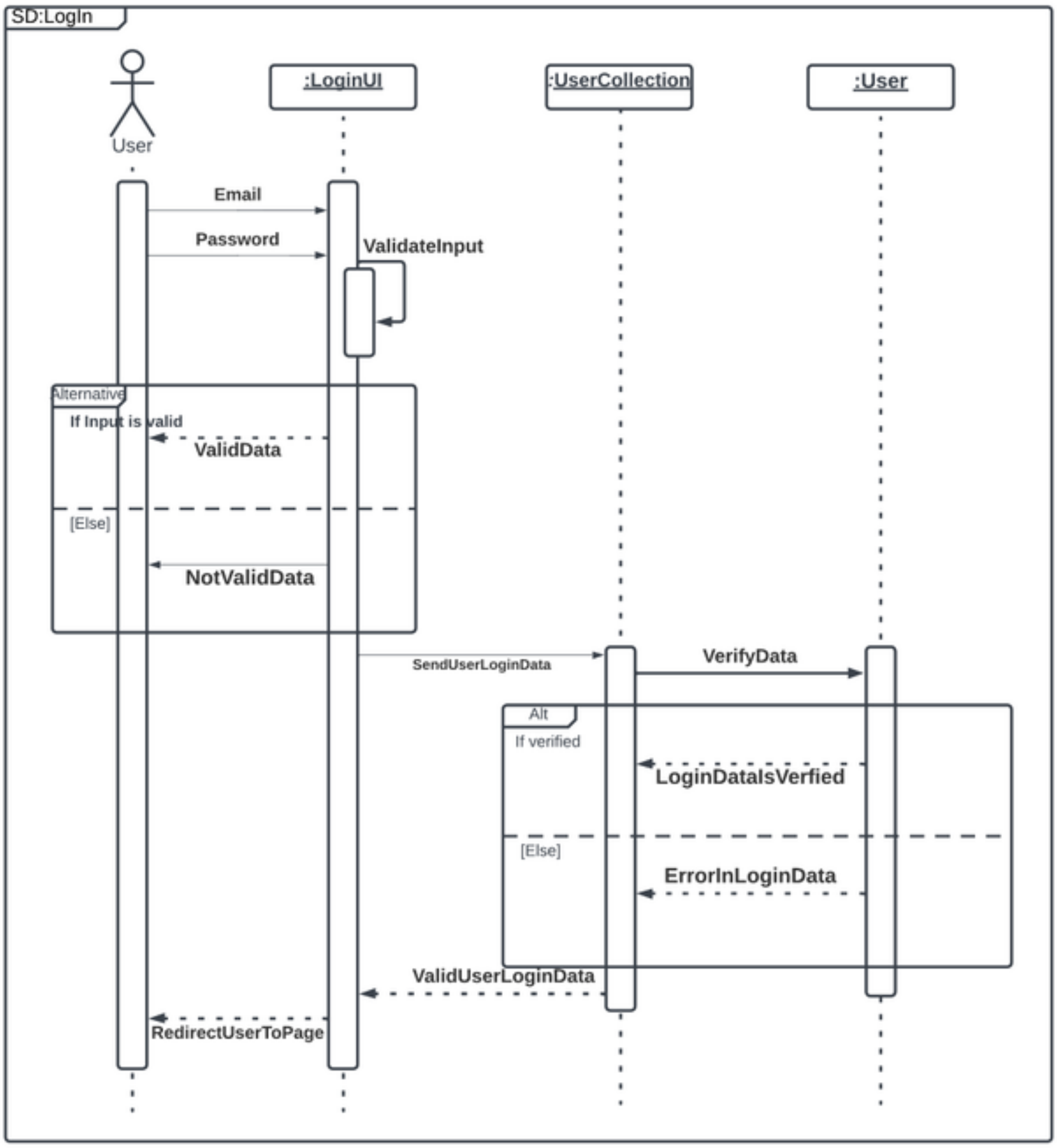


Figure 6: Login in sequence diagram

## 4.5.2 Sign Up Page

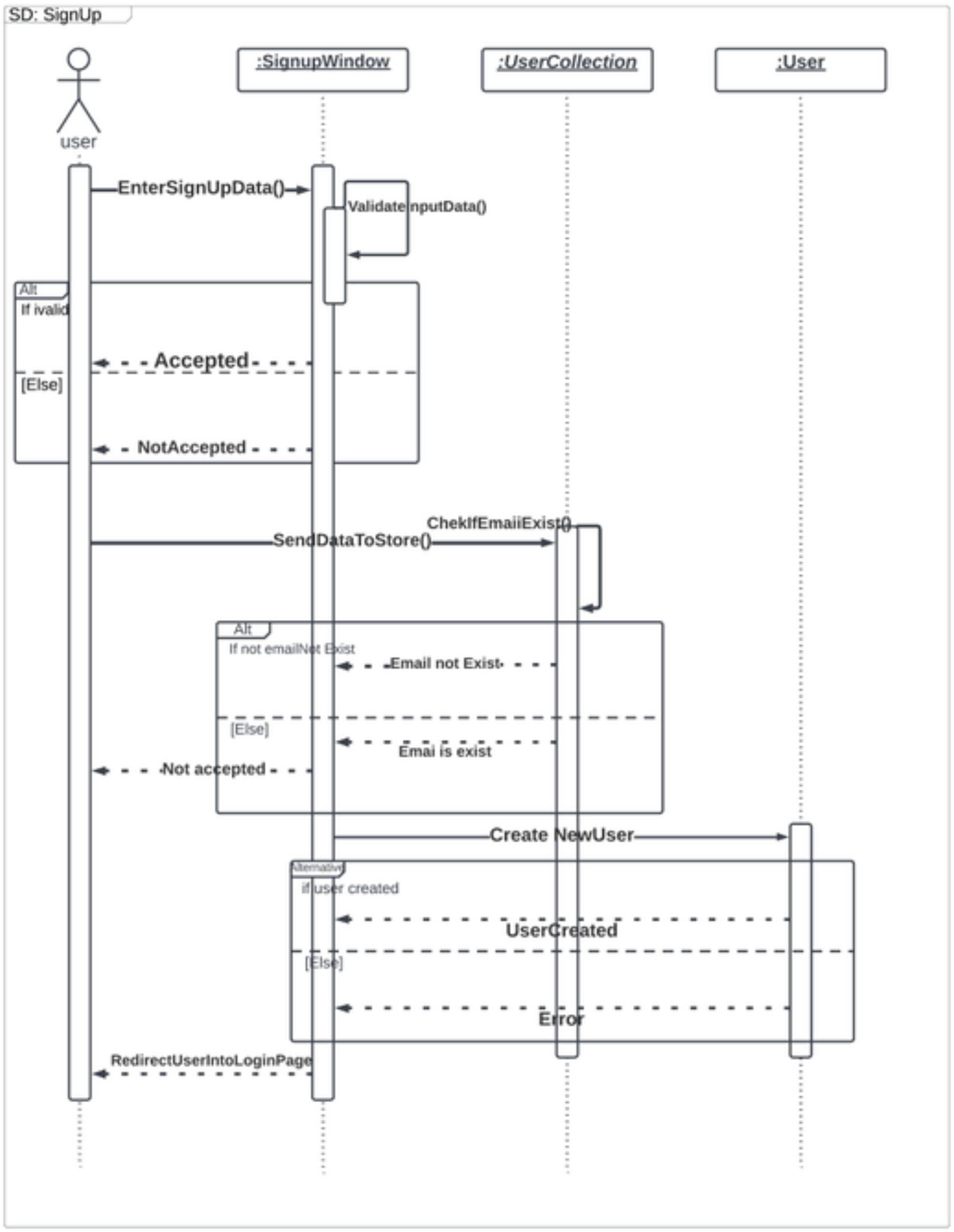


Figure 7: Signup sequence diagram

## 4.5.3 Admin Dashboard

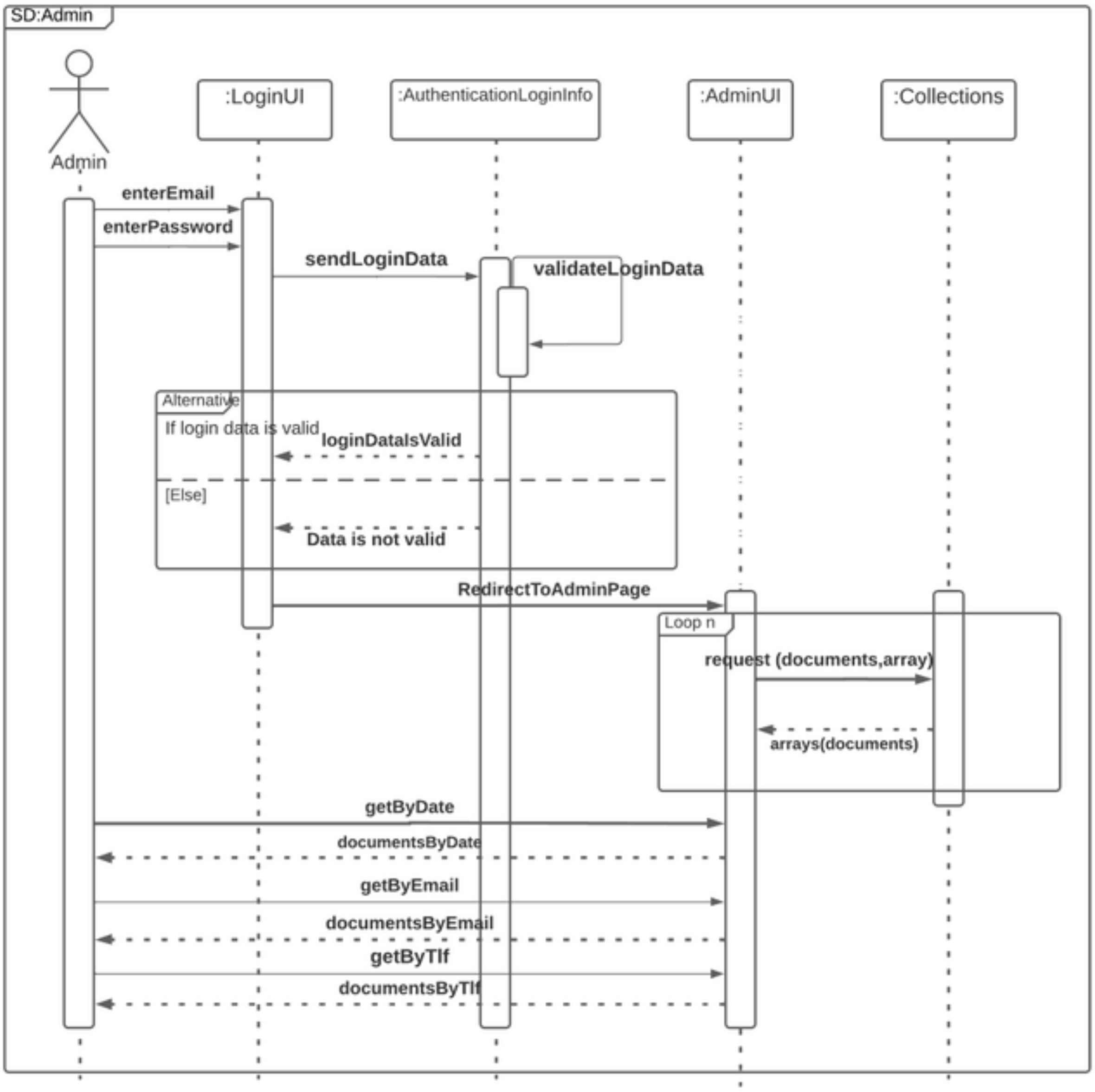


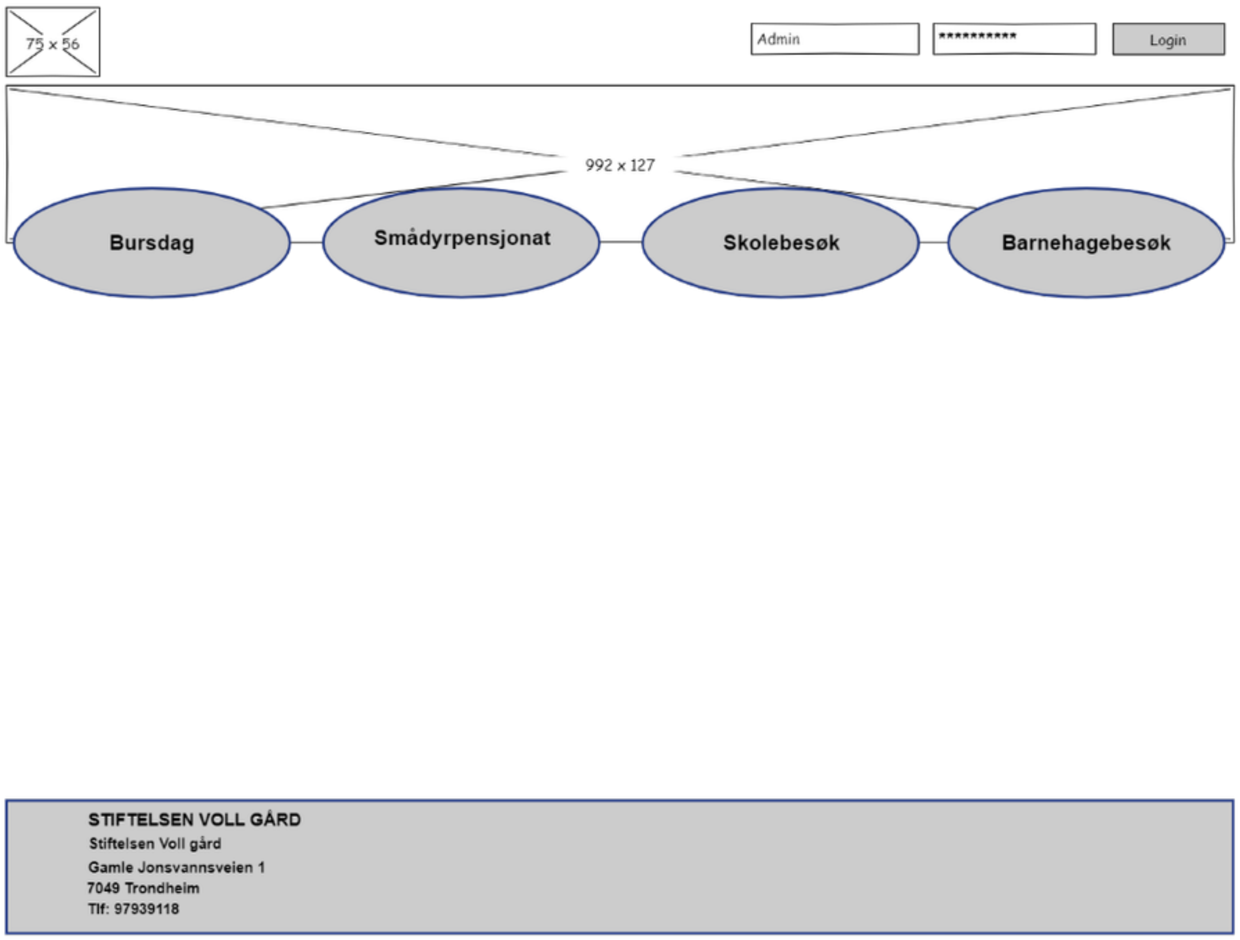
Figure 8: Admin dashboard sequence diagram

# 5 Prototyping

## 5.1 Wireframes

The following wireframes were created in the beginning of the project. The tool used was Pencil Project, which is a free, open source tool. The actual wireframes are clickable, but this feature is not available here as the following figures only are screenshots.

### 5.1.1 Home Page



STIFTELSEN VOLL GÅRD
Stiftelsen Voll gård
Gamle Jonsvannsveien 1
7049 Trondheim
Tlf: 97939118

Figure 9: The home page includes information about the farm and their services. In the navigation bar at the top of the page is the logo on the left and a button for admin login on the right. The available services are presented as buttons in a menu. The buttons are linked to the different booking pages.

## 5.1.2 Birthday Celebration

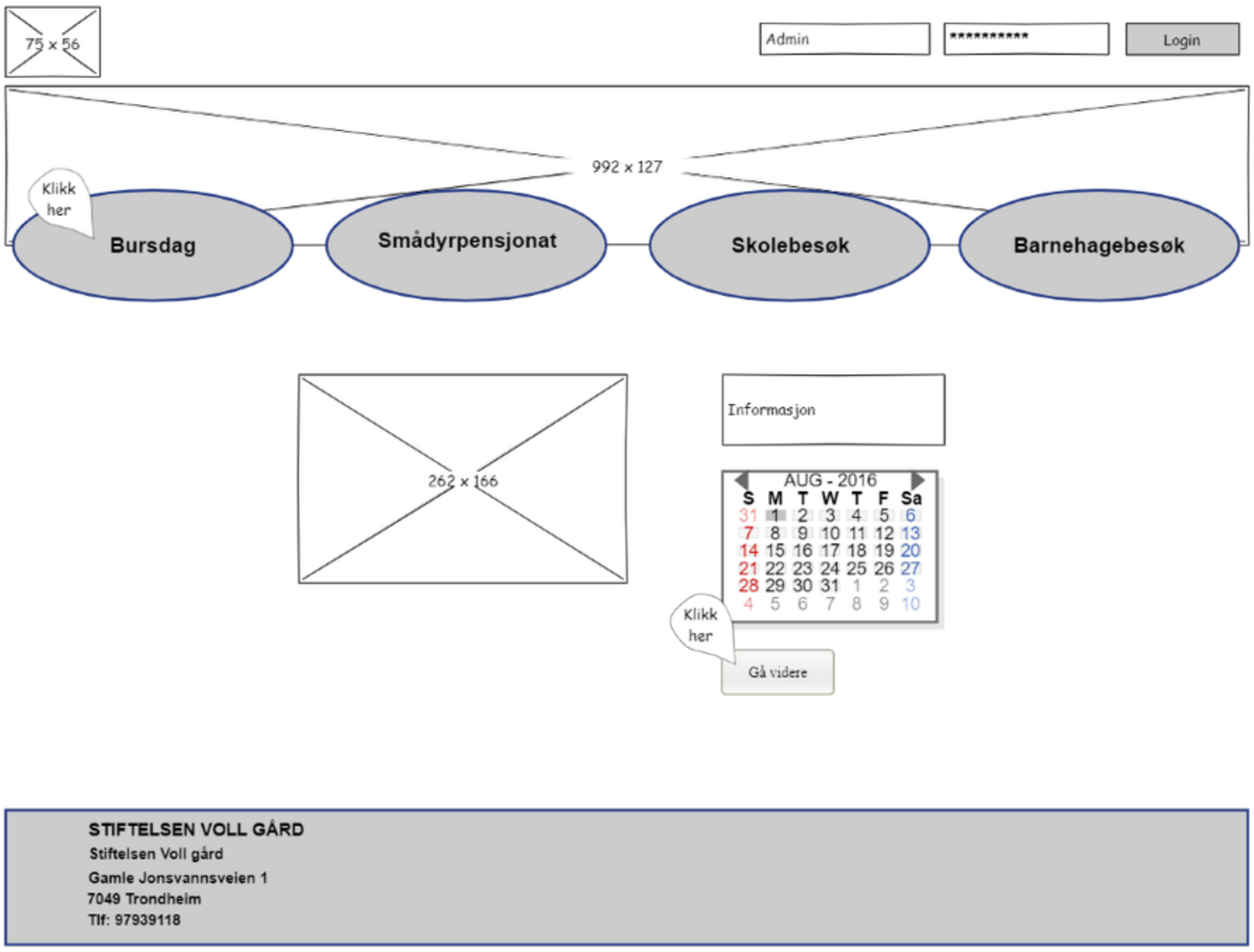


Figure 10: When clicking "bursdag" from the home page, the user is sent to the main page for birthday celebration booking. This page includes images and information about the service, as well as a calendar where the user can choose a suitable date for the event.

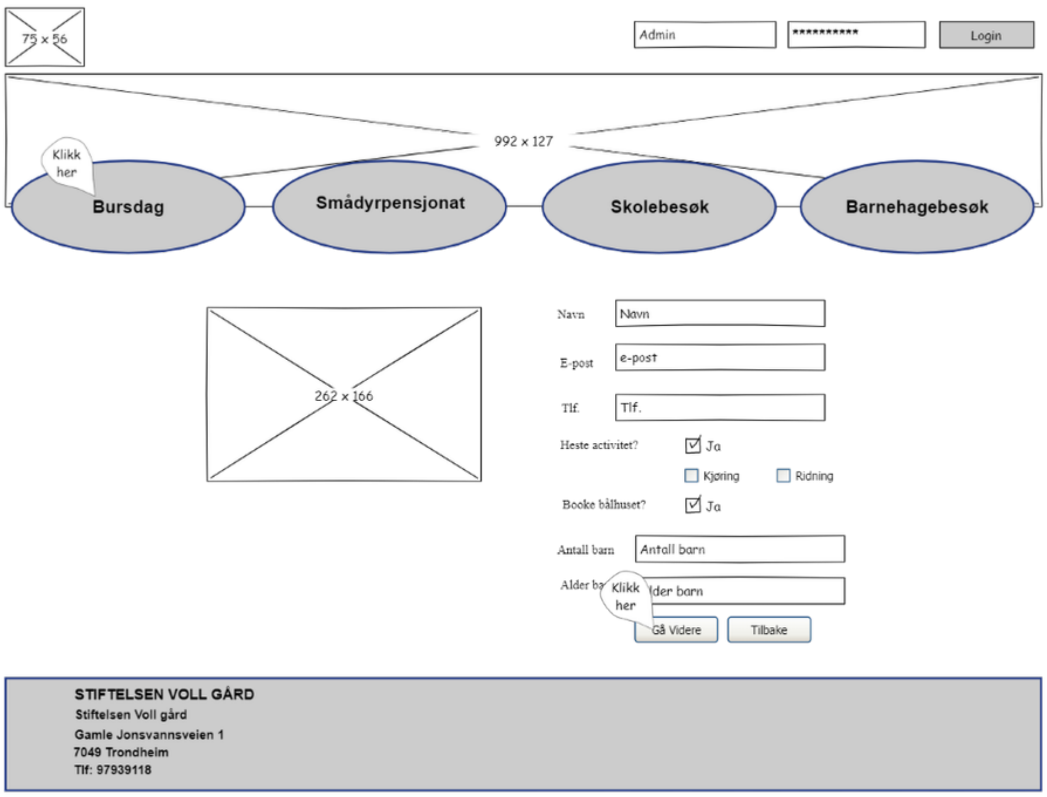


Figure 11: After choosing a suitable date and clicking on, the user comes to a form where all necessary information must be entered.

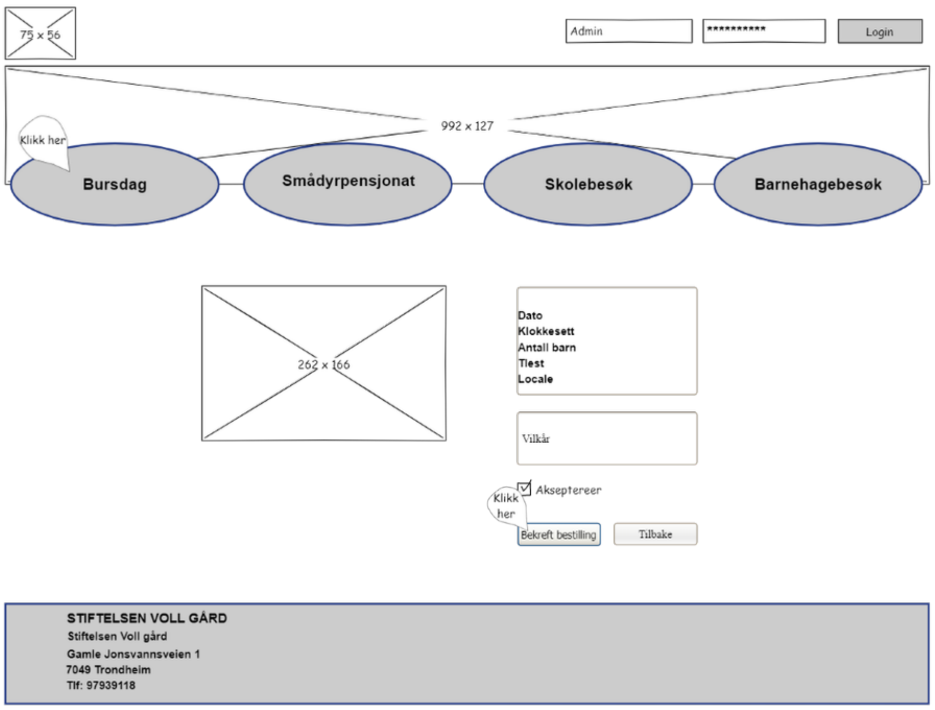


Figure 12: After clicking on again, the user comes to a booking summary where entered details, as well as terms and conditions, are presented and must be accepted to confirm the booking.

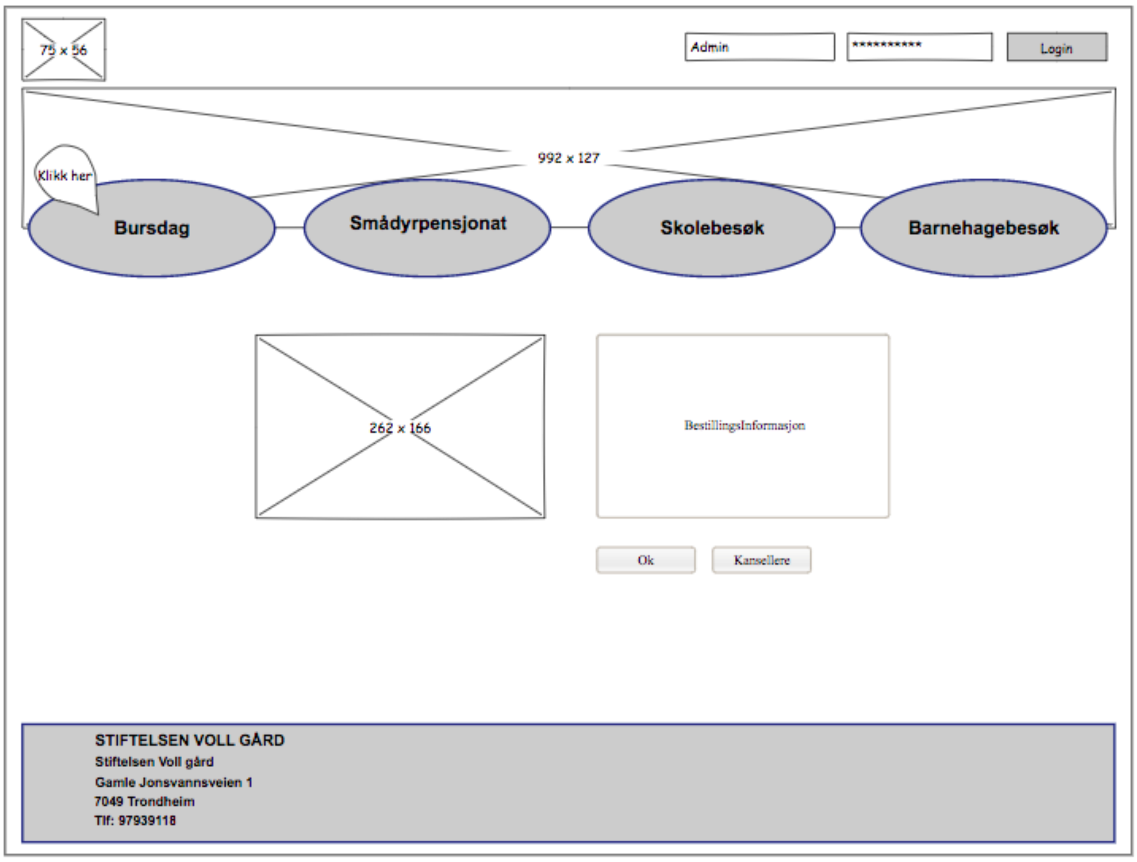


Figure 13: Finally, the user receives a confirmation that the booking has been received.

## 5.1.3 Animal Boarding

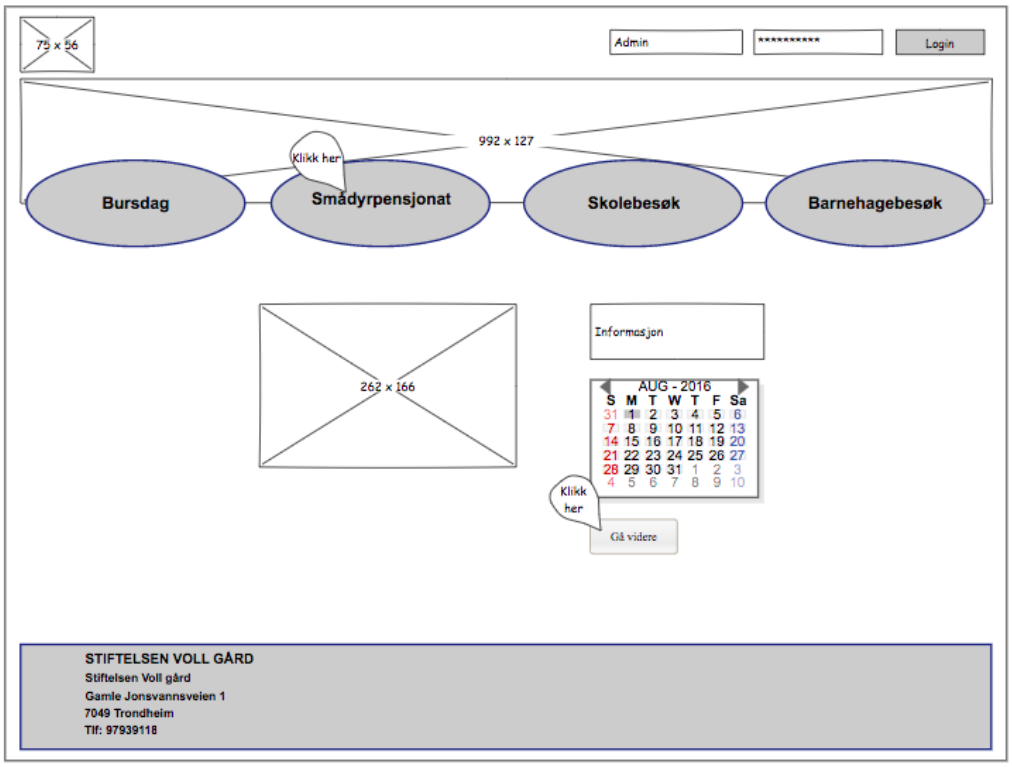


Figure 14: When clicking "smådyrpensjonat" from the home page, the user is sent to the main page for animal boarding booking. This page includes images and information about the service, as well as a calendar where the user can choose the date on which the animal is to be delivered to the farm.

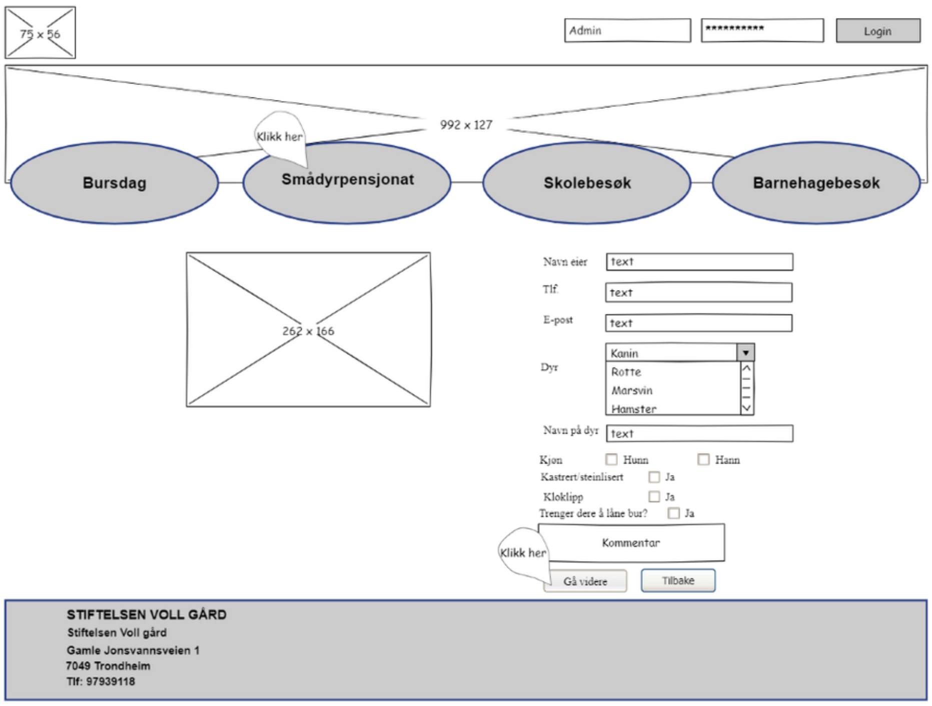


Figure 15: After choosing a suitable date and clicking on, the user comes to a form where all necessary information must be entered.

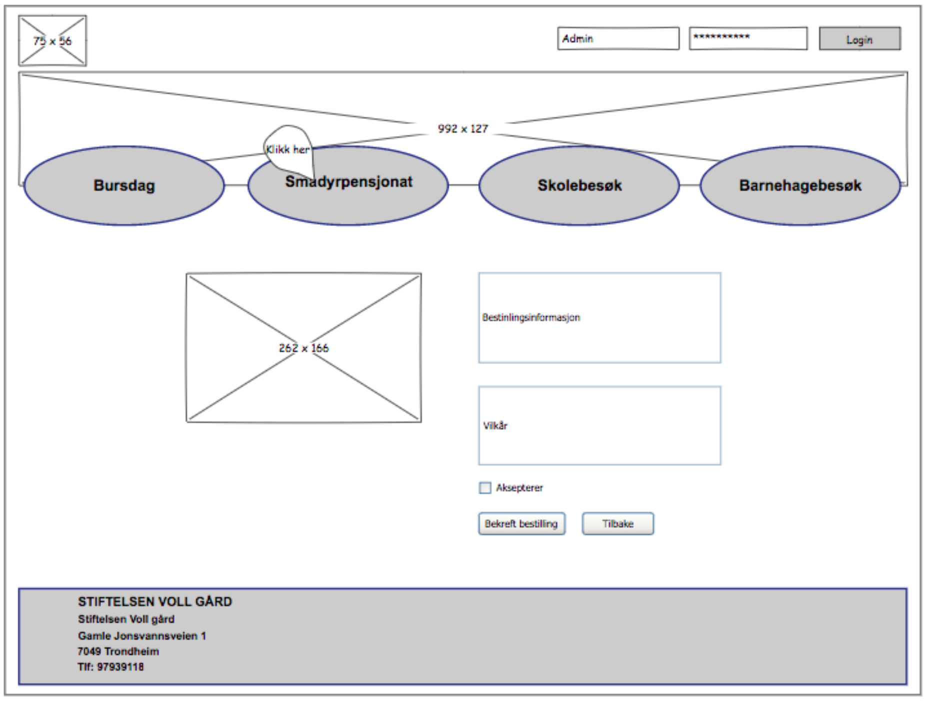


Figure 16: After clicking on again, the user comes to a booking summary where entered details, as well as terms and conditions, are presented and must be accepted to confirm the booking.

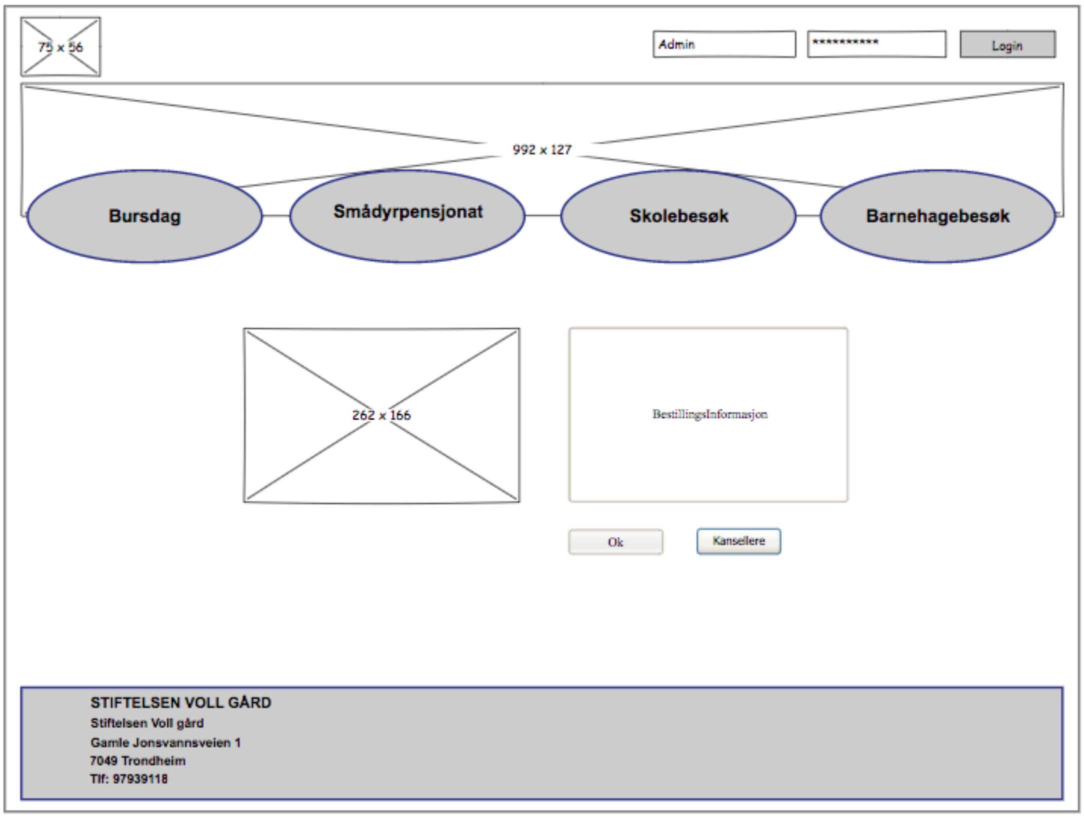


Figure 17: Finally, the user receives a confirmation that the booking has been received.

## 5.1.3 School Visit

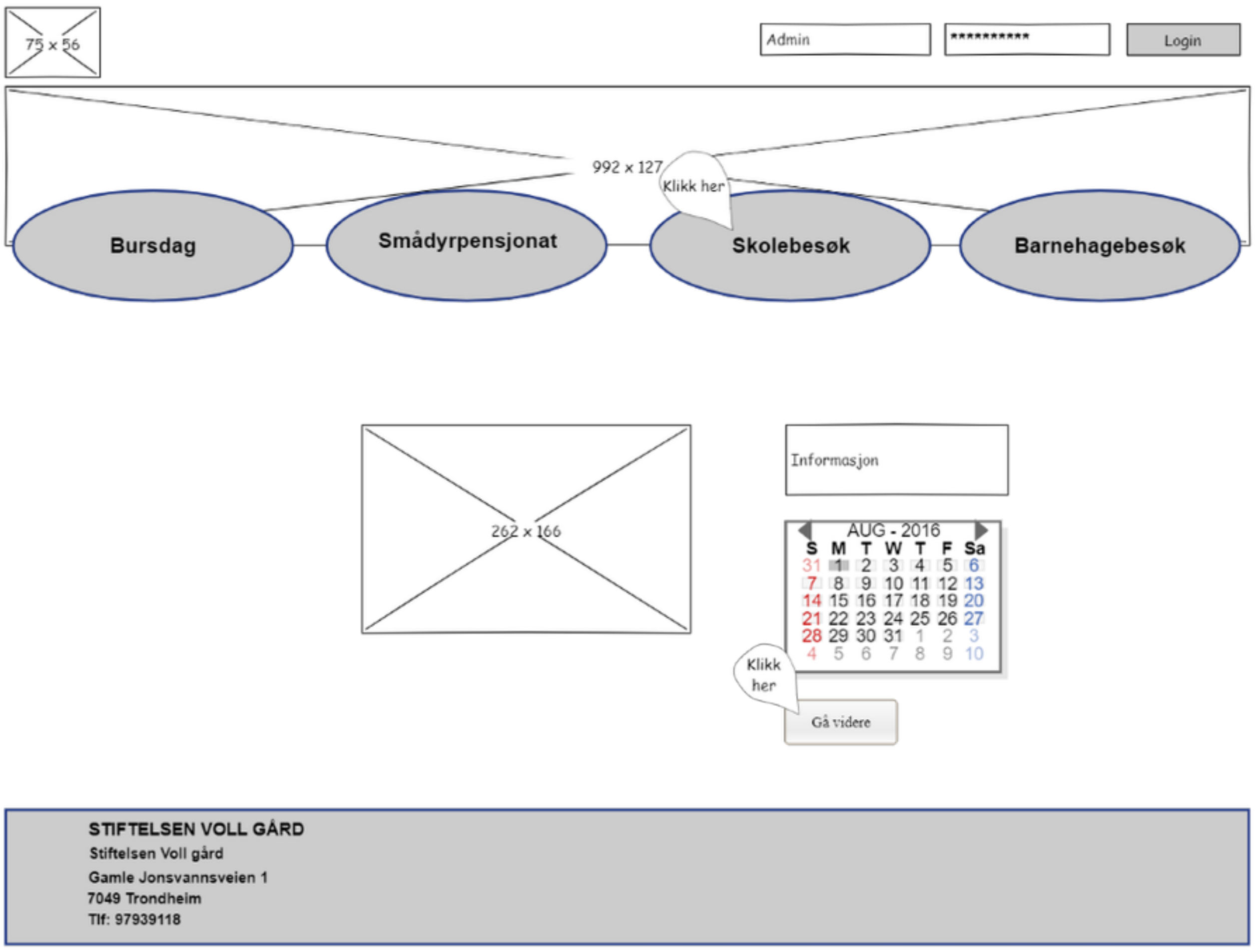


Figure 18: When clicking "skolebesøk" from the home page, the user is sent to the main page for booking school visits. This page includes images and information about the service, as well as a calendar where the user can choose a suitable date for the visit.

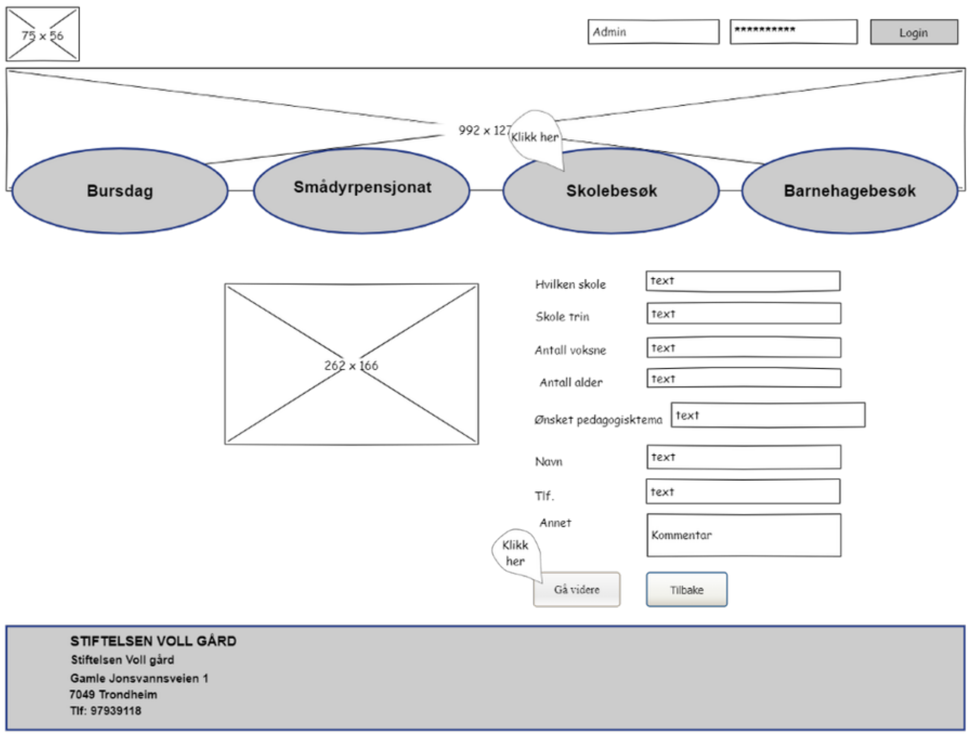


Figure 19: After choosing a suitable date and clicking on, the user comes to a form where all necessary information must be entered.

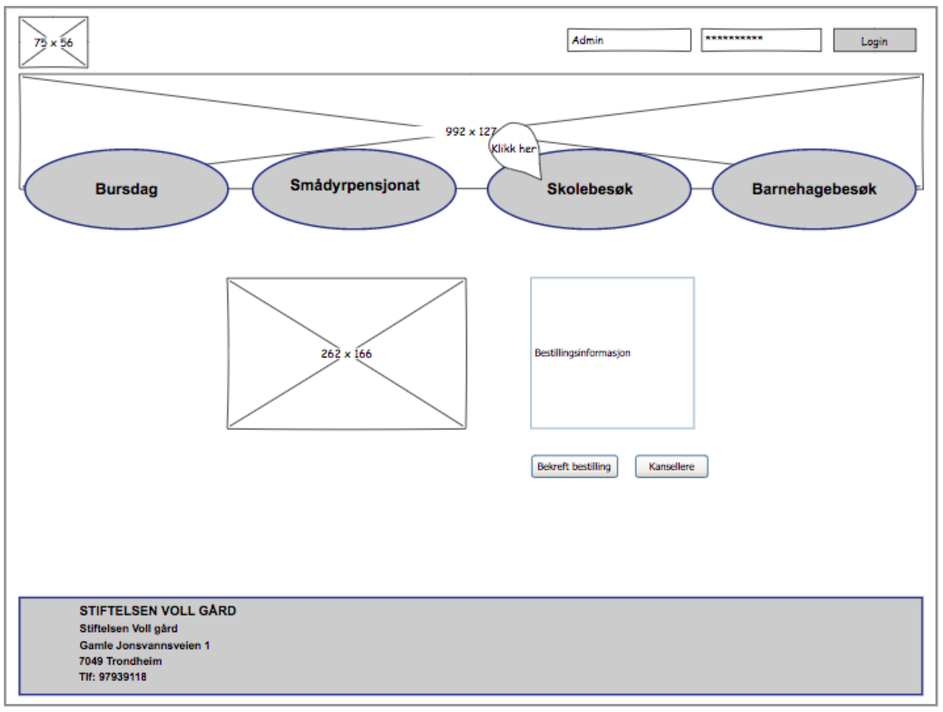


Figure 20: After clicking on again, the user comes to a booking summary where entered details are presented and must be accepted to confirm the booking.

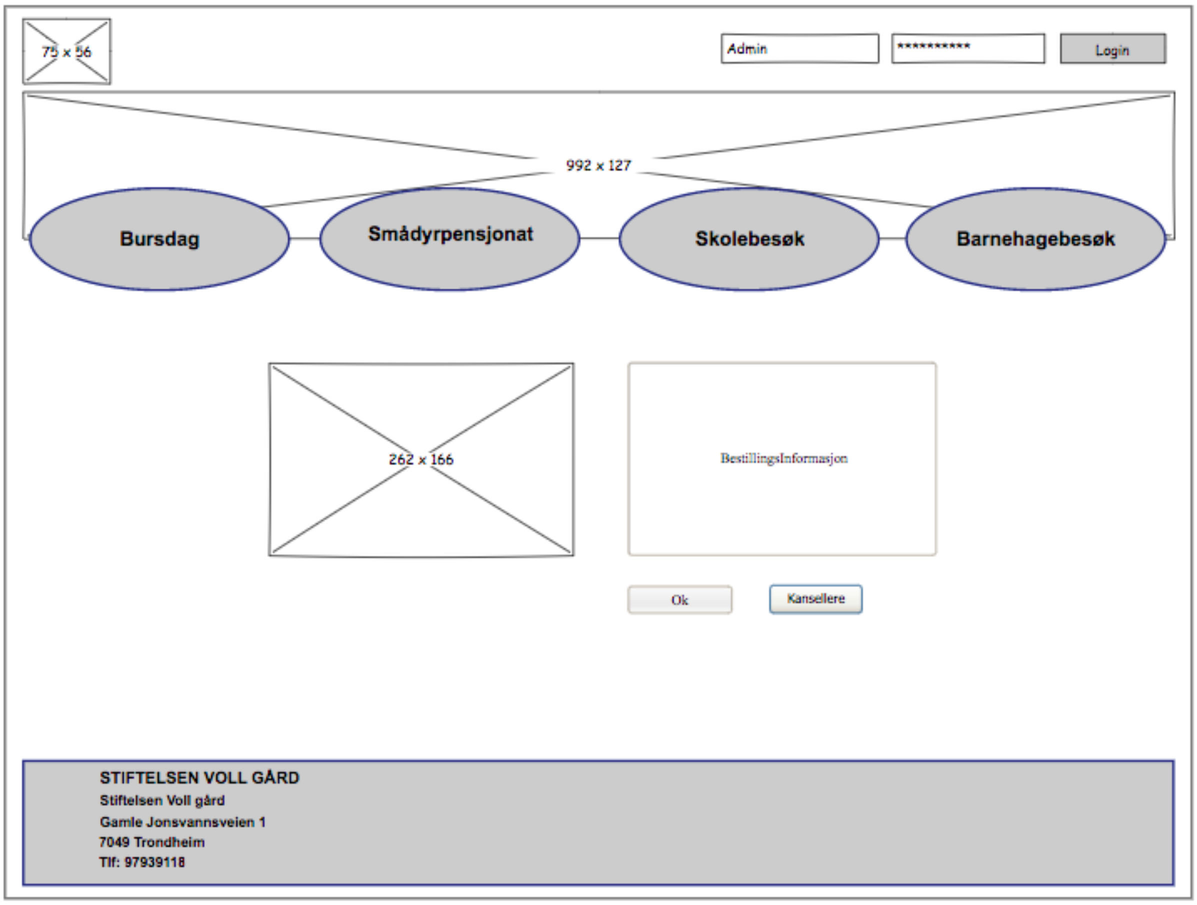


Figure 21: Finally, the user receives a confirmation that the booking has been received.

## 5.1.4 Kindergarten Visit

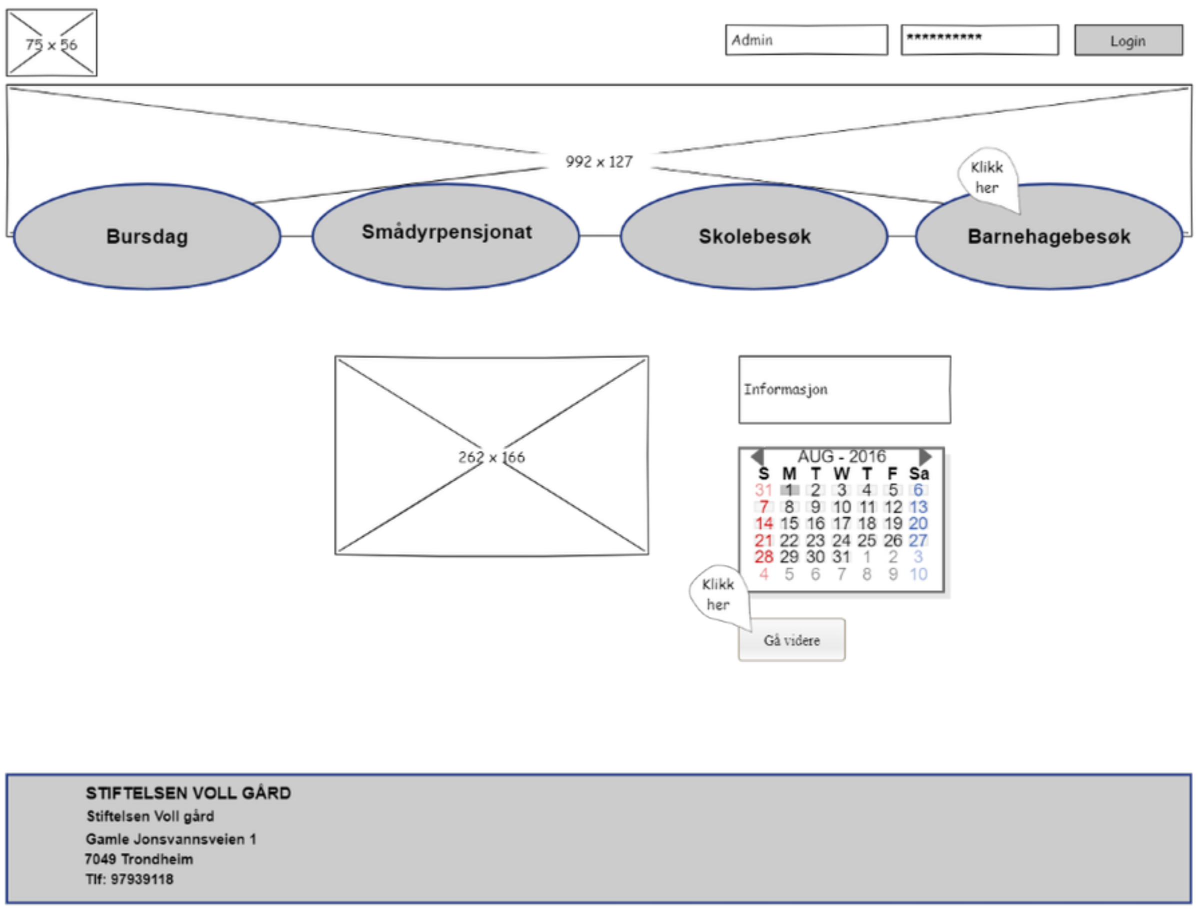


Figure 22: When clicking "barnehagebesøk" from the home page, the user is sent to the main page for booking kindergarten visits. This page includes images and information about the service, as well as a calendar where the user can choose a suitable date for the visit.

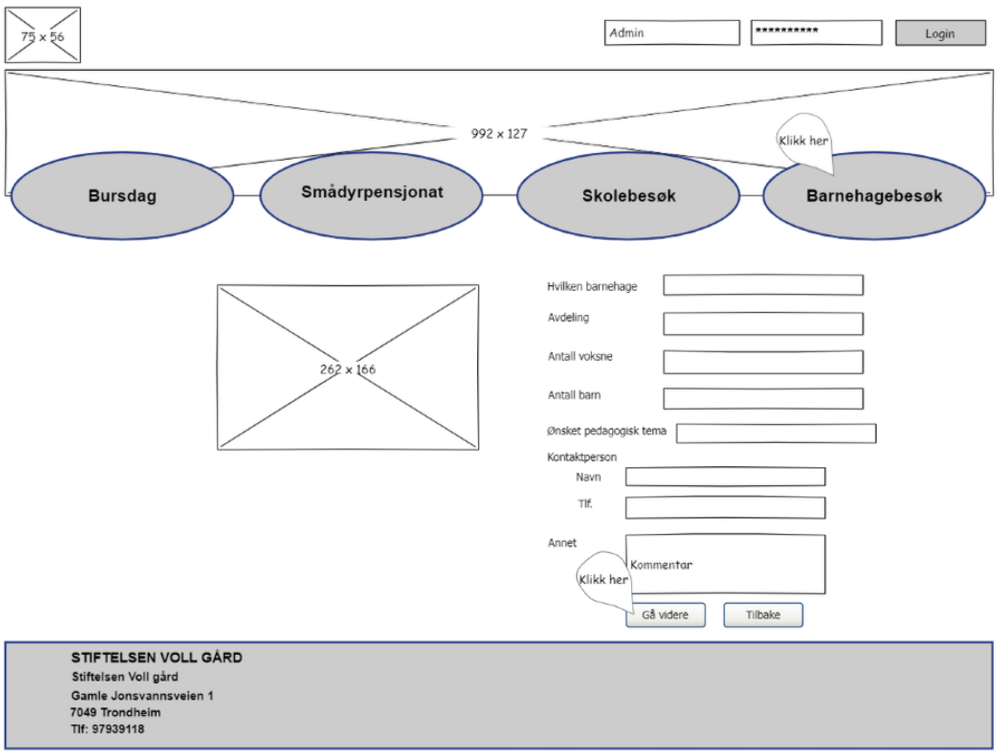


Figure 23: After choosing a suitable date and clicking on, the user comes to a form where all necessary information must be entered.

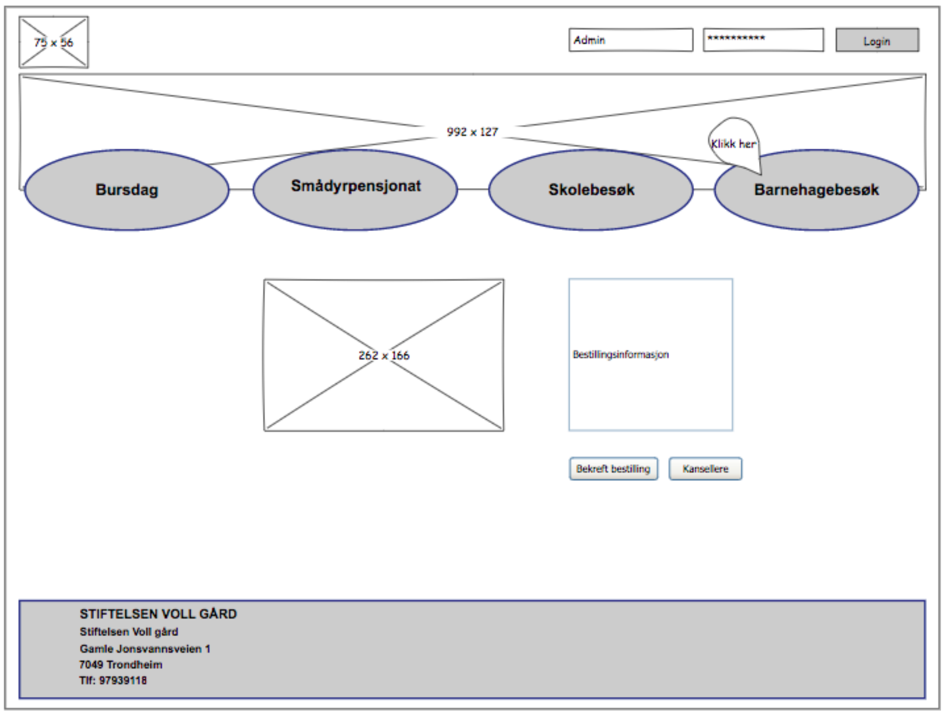


Figure 24: After clicking on again, the user comes to a booking summary where entered details are presented and must be accepted to confirm the booking.

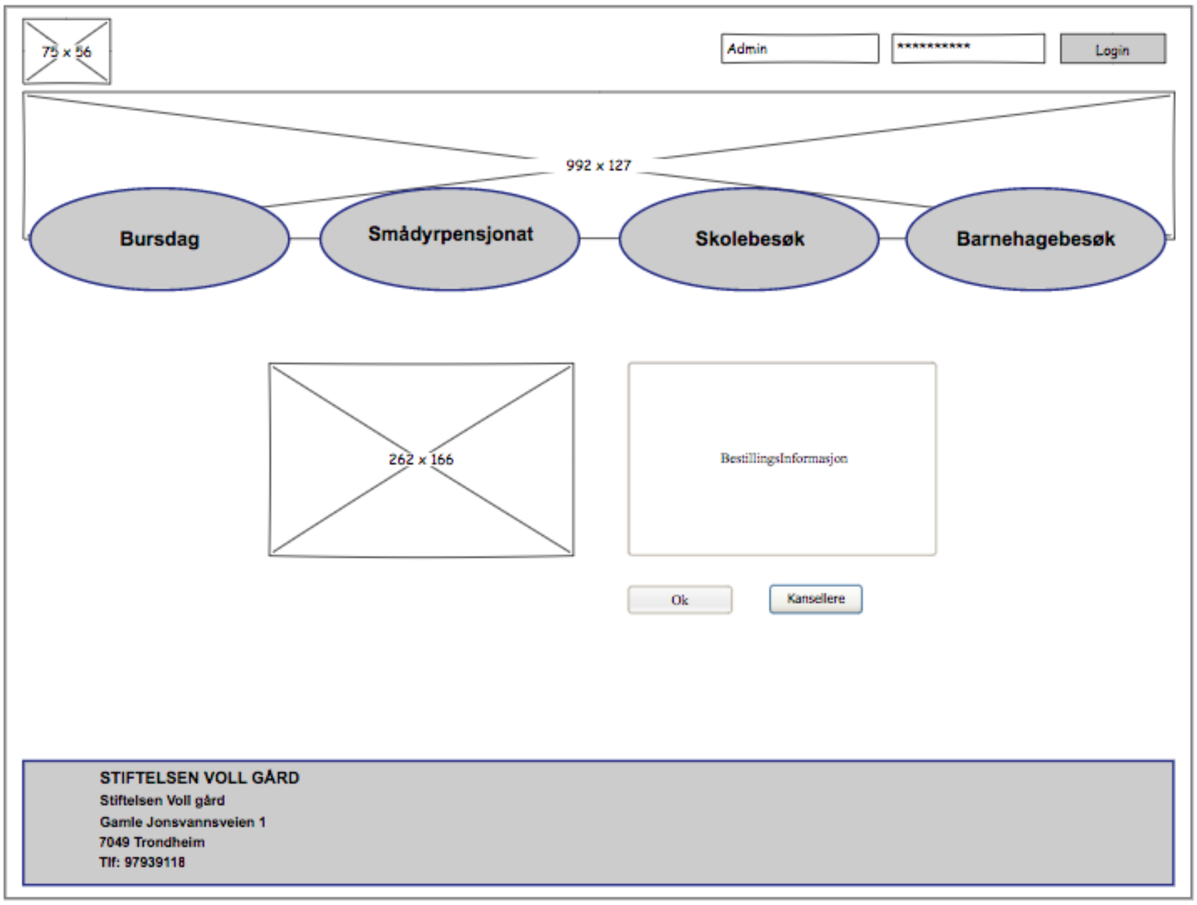


Figure 25: Finally, the user receives a confirmation that the booking has been received.

## 5.1.5 Admin Page

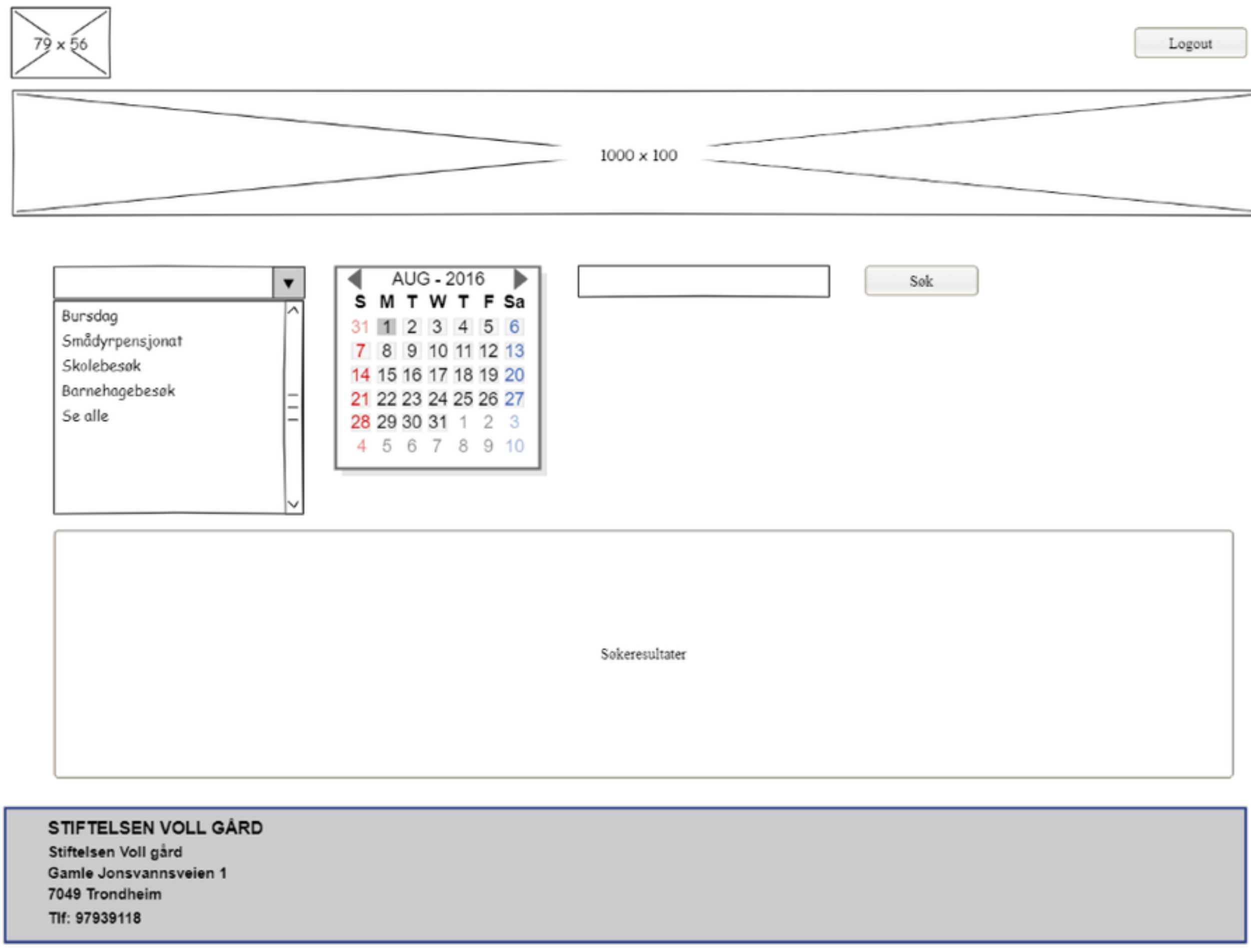


Figure 26: Wireframe of the admin dashboard with an overview of all bookings and the possibility of filtering, sorting and searching through these bookings.

## D System Document

123

# Developing a Booking System for Voll gård System Document

**Version 1.4**

## Revision History

| Dato | Version | Describing | Author |
|---|---|---|---|
| 22/04/2023 | 1.0 | Starting with architecture sketch, security, installation and running, Documentation, and CI. | M. Mousa |
| 24/04/2023 | 1.1 | Updating architecture sketch. | M. Mousa |
| 02/05/2023 | 1.2 | Updating server-client, CI, and Documentation. | M. Mousa |
| 10/05/2023 | 1.3 | Class diagram, updating testing, and documentation. | M. Mousa |
| 19/05/2023 | 1.4 | Adding project structure and the final adjustment. | M. Mousa, C. Ossletten |

# Table of Contents

**1 Introduction** **4**
**2 Architecture Sketch** **4**
**3 Project Structure** **5**
**4 Class Diagram** **8**
**5 Database Model** **9**
**6 Server-Client** **11**
**7 Security** **12**
**8 Installation and Running** **13**
8.1 External dependencies 13
8.2 Running the Application 13
**9 Documentation of Source Code** **14**
10.1 Continuous Integration 15
10.2 Testing 17
10.2.1 Test Coverage 17
**11 References** **19**

## 1 Introduction

The system documentation is written as a part of the computer engineering program at NTNU and the 3rd year course IDATT2900 Bachelor's Thesis (part 2 of 2). The bachelor's thesis title is "Developing a booking system for Voll Gård," with Voll Gård as the contractor.

The system documentation has been written through the semester and developing process. It is technical documentation describing a web application´s architecture, design, and features. The team members use this document to convey the design and features of the developed booking system to all stakeholders and ensure the system fulfills the requirements of users and stakeholders.

## 2 Architecture Sketch

The figure below shows a sketch of the system's architecture, followed by descriptions of each part.

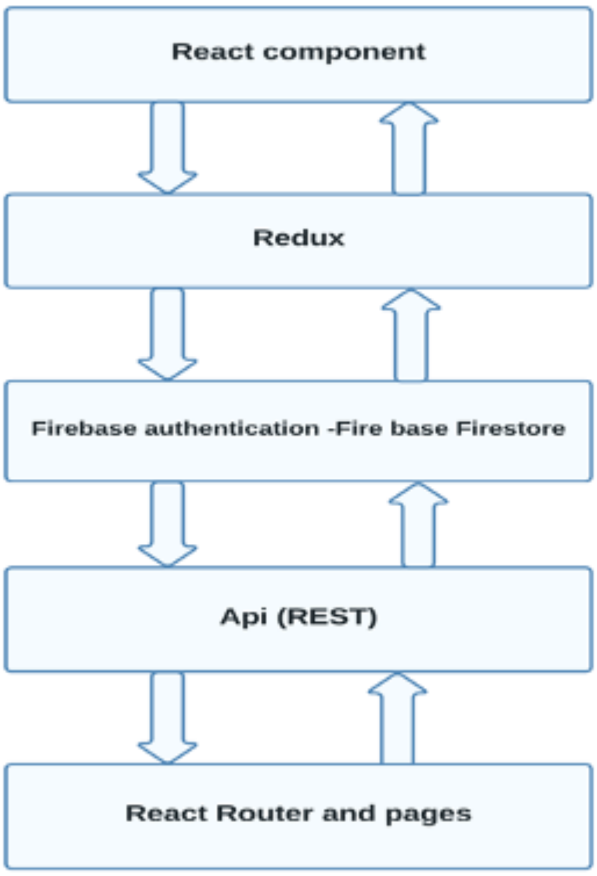


Figure 1: Architecture Sketch.

**React components:** Represent the web application's view layer, meaning the user interface. These are built using React and display data and functionality to the user.

**Redux:** Used as a state library to handle the application's state. Redux allows for a centralized store of data that any component in the application can access. This library has managed complex states and shared data between React components. The data and information will be sent between different views without complexity using Redux. And through the web-based booking system, the data should be sent through several views before POST to the backend.

**Firebase Authentication:** Used to authenticate admin users. Firebase Authentication provides secure authentication using email and password. All user data will be in store before sending this data into user collection in the Firebase authentication service.

**Firestore:** Used as a database layer for the web application. Firestore is a cloud-hosted NoSQL database that stores data as documents in collections. After the user fills in the event forms in the User interface and submits them, all data will be held in the Redux store. The data after that will be sent from this store to the cloud Firestore.

**API layer:** The API layer can be implemented using REST methods like POST, GET, DELETE, and UPDATE to communicate with the backend. The API layer handles the communication between the react user interface and the Firebase end, allowing for the exchange of data and information. Because the Firestore is a cloud service, it uses the react-router with REST methods to REST data and communication between the backend and the user interface.

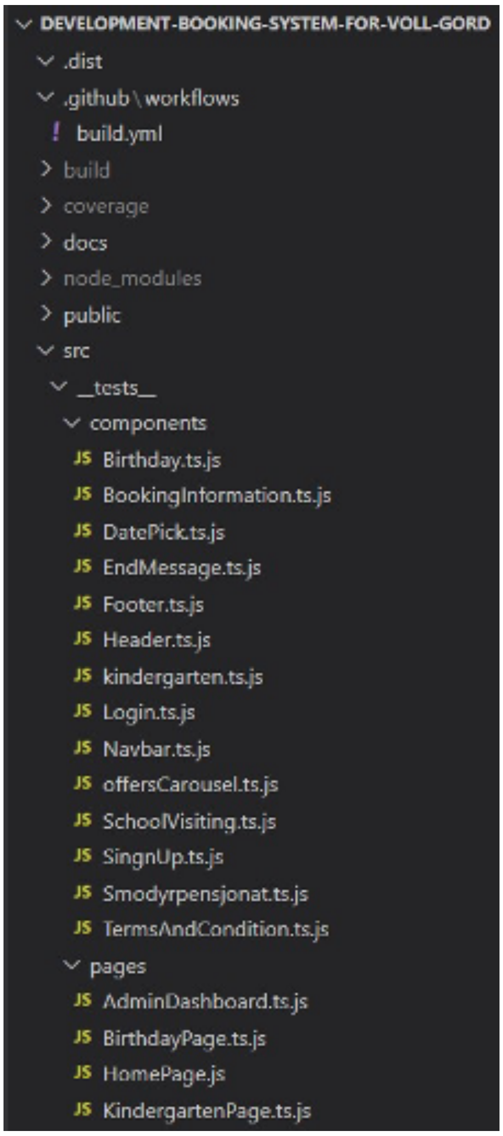
**React Router and pages**: Handles the views. React Router is a routing library that allows for routing in a React application. Pages represent different views in the application, and React Router handles navigation between them.

# 3 Project Structure

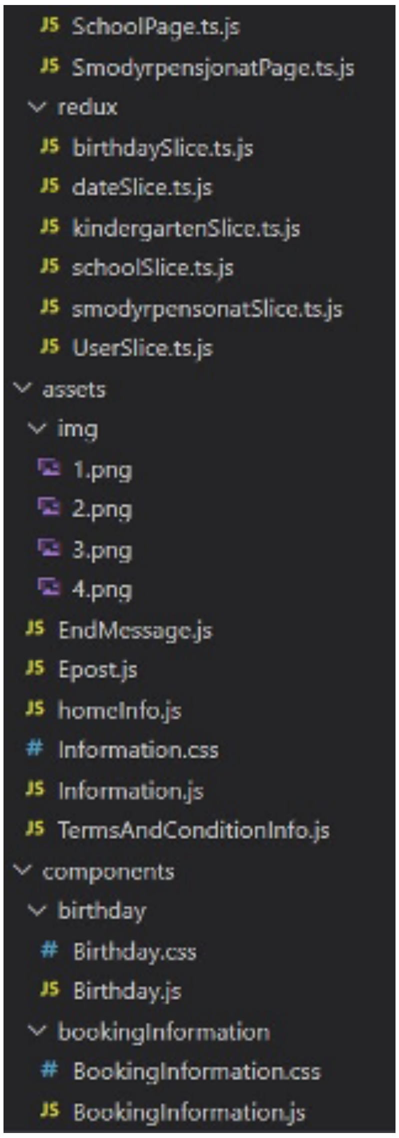
The project root contains several folders: the github\workflow folder, __test__ folder, component folder, hooks folder, page folder, and redux folder. More details about folders are as follows:

- The workflow folder contains one file, "build.yml".
- __test__ folder contains all the testing files, which have the name of the components files.
- Components folder contains all reusable components with their styles.
- Redux folder, which contains all created slices and stores for the project.
- Hooks folder, which contains all custom hooks such as useLogin and useLogout.
- Assets folder, which contains objects and images.
- On the project's root, there are several files for configurations and Docker files.

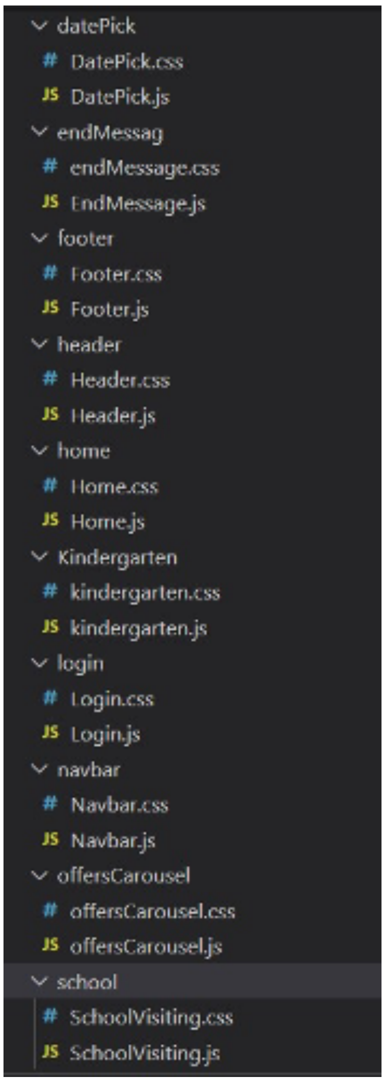
The following screenshots show the structure of the project:



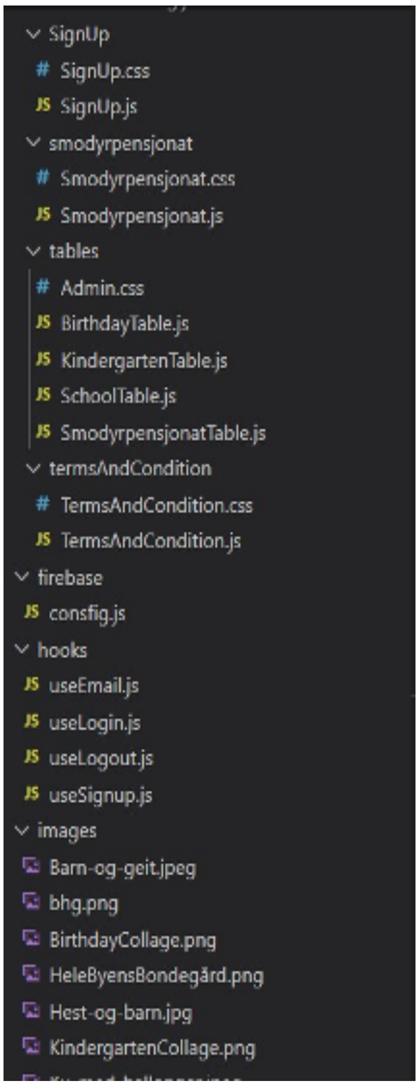
```
∨ DEVELOPMENT-BOOKING-SYSTEM-FOR-VOLL-GORD
  ∨ .dist
  ∨ .github \ workflows
     ! build.yml
  > build
  > coverage
  > docs
  > node_modules
  > public
  ∨ src
    ∨ __tests__
      ∨ components
        JS Birthday.ts.js
        JS BookingInformation.ts.js
        JS DatePick.ts.js
        JS EndMessage.ts.js
        JS Footer.ts.js
        JS Header.ts.js
        JS kindergarten.ts.js
        JS Login.ts.js
        JS Navbar.ts.js
        JS offersCarousel.ts.js
        JS SchoolVisiting.ts.js
        JS SingnUp.ts.js
        JS Smodyrpensjonat.ts.js
        JS TermsAndCondition.ts.js
      ∨ pages
        JS AdminDashboard.ts.js
        JS BirthdayPage.ts.js
        JS HomePage.js
        JS KindergartenPage.ts.js
```



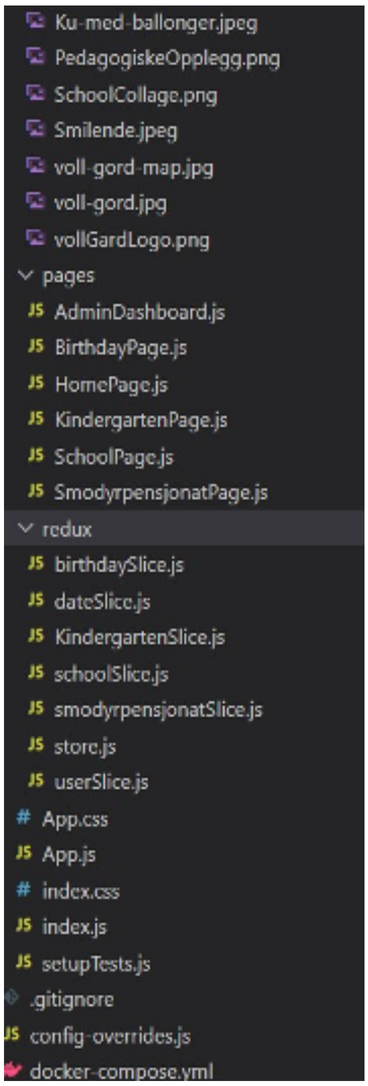
```
      JS SchoolPage.ts.js
      JS SmodyrpensjonatPage.ts.js
    ∨ redux
      JS birthdaySlice.ts.js
      JS dateSlice.ts.js
      JS kindergartenSlice.ts.js
      JS schoolSlice.ts.js
      JS smodyrpensonatSlice.ts.js
      JS UserSlice.ts.js
  ∨ assets
    ∨ img
      1.png
      2.png
      3.png
      4.png
    JS EndMessage.js
    JS Epost.js
    JS homeInfo.js
    # Information.css
    JS Information.js
    JS TermsAndConditionInfo.js
  ∨ components
    ∨ birthday
      # Birthday.css
      JS Birthday.js
    ∨ bookingInformation
      # BookingInformation.css
      JS BookingInformation.js
```



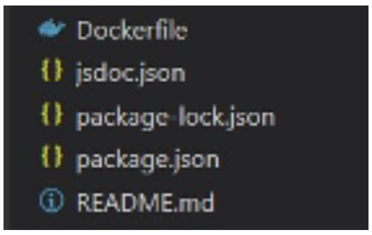
```
datePick
  DatePick.css
  DatePick.js
endMessag
  endMessage.css
  EndMessage.js
footer
  Footer.css
  Footer.js
header
  Header.css
  Header.js
home
  Home.css
  Home.js
Kindergarten
  kindergarten.css
  kindergarten.js
login
  Login.css
  Login.js
navbar
  Navbar.css
  Navbar.js
offersCarousel
  offersCarousel.css
  offersCarousel.js
school
  SchoolVisiting.css
  SchoolVisiting.js
```



```
  SignUp
    SignUp.css
    SignUp.js
  smodyrpensjonat
    Smodyrpensjonat.css
    Smodyrpensjonat.js
  tables
    Admin.css
    BirthdayTable.js
    KindergartenTable.js
    SchoolTable.js
    SmodyrpensjonatTable.js
  termsAndCondition
    TermsAndCondition.css
    TermsAndCondition.js
firebase
  config.js
hooks
  useEmail.js
  useLogin.js
  useLogout.js
  useSignup.js
images
  Barn-og-geit.jpeg
  bhg.png
  BirthdayCollage.png
  HeleByensBondegård.png
  Hest-og-barn.jpg
  KindergartenCollage.png
```



```
  Ku-med-ballonger.jpeg
  PedagogiskeOpplegg.png
  SchoolCollage.png
  Smilende.jpeg
  voll-gord-map.jpg
  voll-gord.jpg
  vollGardLogo.png
pages
  AdminDashboard.js
  BirthdayPage.js
  HomePage.js
  KindergartenPage.js
  SchoolPage.js
  SmodyrpensjonatPage.js
redux
  birthdaySlice.js
  dateSlice.js
  KindergartenSlice.js
  schoolSlice.js
  smodyrpensjonatSlice.js
  store.js
  userSlice.js
App.css
App.js
index.css
index.js
setupTests.js
.gitignore
config-overrides.js
docker-compose.yml
```



```
Dockerfile
jsdoc.json
package-lock.json
package.json
README.md
```

Figure 2, 3, 4, 5, and 6: Shows the structure of the project.

# 4 Class Diagram

The project uses React and Firebase. Therefore, the class diagram shown in Figure 7 will be just in the front end. The project contains several functional components that are treated as classes.

The "App" component is the top-level component responsible for rendering the entire app. The user includes two types of users - authenticated users and unauthenticated users. The component "DatePick" shows all information associated with the booking event, and the user first chooses the date. The "Birthday", "Smodyrpensjonat", "School", and "Kindergarten" components are responsible for filling in all the required data. The "termsAndConditions" component is a reusable component used to show input data, accept conditions and submit birthday and animal boarding input data into Firebase Firestore. The component "BookingInformation" is the reusable component responsible for showing and submitting the school and kindergarten input data. The Login component is a component that is responsible for checking the user data before navigating to the "AdminDashboard" component. The "AdminDashboard" component is responsible for retrieving, updating, and deleting the data from Firebase Firestore.

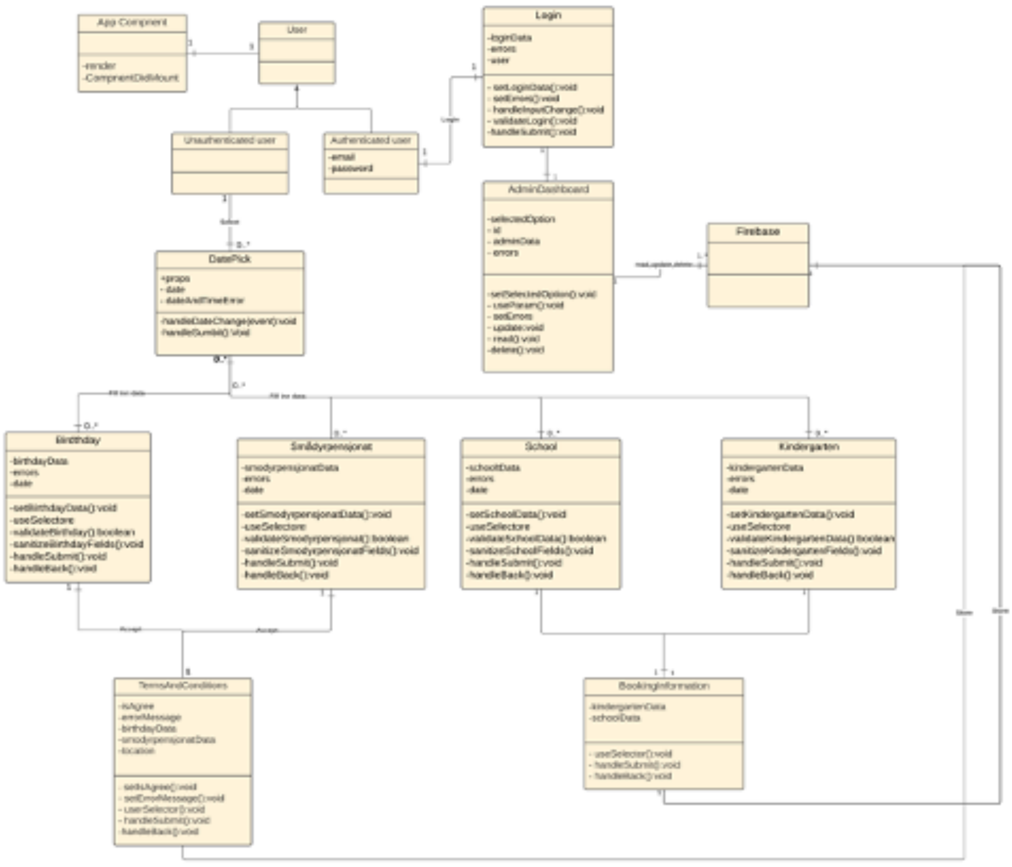


Figure 7: Class Diagram.

# 5 Database Model

Firebase Firestore, which stores the data from the booking system, is a document-based NoSQL database. It includes four collections (birthday, smodyrpensjonat, school, kindergarten), and each collection contains documents in a JSON-like format. Each document includes several fields.

**collection:’ birthday’**

- Document: ’birthday-id’
- Fields:
    - ‘name’: string
    - ‘timeSlot’: string
    - ‘email’: string
    - ‘tlf’: string
    - ‘bookFireHouse’: boolean
    - horsActivity: boolean
    - ‘horseOptions’: string
    - ‘numberOfChildren’: string
    - ‘age’: string
    - ‘comment’: string

**collection: ‘smodyrpensjonat’**

- Document: ‘smodyrpensjonat-id’
- Fields:
    - ‘name’: string
    - ‘email’: string
    - ‘tlf’: string
    - ‘typeOfAnimal’: string
    - ‘nameOfAnimal’: string
    - ‘genderOptions’: string
    - ‘cage’: boolean
    - ‘clawClip’: boolean
    - ‘neutered’: boolean
    - ‘lengthOfStay’: string
    - ‘date’: string
    - ‘comment’: string

**Collection: ‘school’**

- Document: ‘school-id’
- Fields:
    - ‘name’: string
    - ‘email’: string
    - ‘tlf’: string
    - ‘subject’: string
    - ‘level’: string
    - ‘date’: string
    - ‘nameOfSchool’: string
    - ‘numberOfAdults’: string
    - ‘numberOfChildren’: string

    - ‘comment’: string

**Collection: ‘kindergarten’**

- Document: ‘kindergarten-id’
- Fields:
    - ‘name’: string
    - ‘email’: string
    - ‘tlf’: string
    - ‘date’: string
    - ‘nameOfKindergarten’: string
    - ‘numberOfAdults’: string
    - ‘numberOfChildren’: string
    - ‘section’: string
    - ‘subject’: string
    - ‘kindergartenActivity’: string
    - ‘comment’: string

**Collection: ‘User’**

- Document: ‘user-id’
- Fields:
    - ‘email’: string
    - ‘password’: string

This model allows one to easily add and update data for birthday, smodypensjonat, school, and kindergarten in the Voll Gård booking system using the Firebase Firestore API.

The authentication service contains the user collection, which requires two inputs, email and password.

# 6 Server-Client

The team used Firebase as the backend ,and navigation by using the Router library. The REST resources are as follows:

**'\conditions'**: This resource represents the birthday or animal hostel submitted. It allows for POST (to create a new birthday document or a new animal hostel document).

**'\bookinginfo':** This resource represents the school or animal hostel submitted. It allows for POST (to create a new school document or a new kindergarten document).

**'\admin\{id}\ birthdayinfo'**: This resource represents the collection of all birthday documents in Firebase Firestore. It allows for GET (to retrieve all birthday documents), UPDATE (to update the birthday documents), and DELETE (to delete the birthday documents).

**'\admin\{id}\ smodyrpensjonatinfo'**: This resource represents the collection of all animal hostel documents in Firebase Firestore. It allows for GET (to retrieve all animal hostel documents), UPDATE (to update animal hostel documents), and DELETE (to delete the animal hostel documents).

**'\admin\{id}\ schoolinfo'**: This resource represents the collection of all school documents in Firebase Firestore. It allows for GET (to retrieve all school documents), UPDATE (to update the school document), and DELETE (to delete the school document).

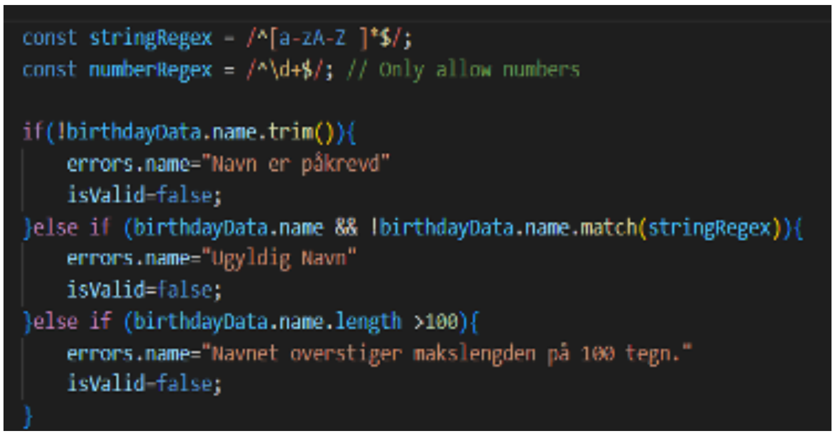
**'\admin\{id}\ kindergarteninfo'**: This resource represents the collection of all kindergarten documents in Firebase Firestore. It allows for GET (to retrieve all kindergarten documents), UPDATE (to update the kindergarten documents), and DELETE (to delete the kindergarten documents).

## 7 Security

The team chose the Firebase cloud platform as the project’s backend. Firebase provides a secure architecture and tools to manage the security of the application. Firebase is hosted on SSL (Secure Sockets Layer), which typically encrypts the connection between client and server, preventing data theft or manipulation at the network layer [1, p. 170].

The team writes security rules to control the access of unauthenticated and authenticated users.
The authenticated user is just the administrator who has the right to read, update and delete documents from several collections. The admin is authenticated by email and password.

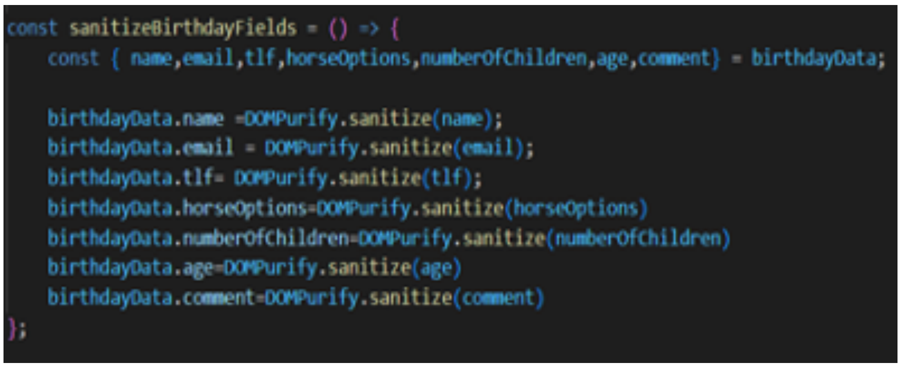
Because the booking is not restricted by authentication and all users have the right to create a document in cloud Firestore, it is essential to check all input data to prevent known attacks. Although the database in Firebase is NoSQL, the team members write coding to check that all input data do not contain any SQL coding before sending data to be stored. Figure 8 is an example from coding showing how input is checked to ensure that the user enters string data.



```
const stringRegex = /^[a-zA-Z ]*$/;
const numberRegex = /^\d+$/; // Only allow numbers

if(!birthdayData.name.trim()){
    errors.name="Navn er påkrevd"
    isValid=false;
}else if (birthdayData.name && !birthdayData.name.match(stringRegex)){
    errors.name="Ugyldig Navn"
    isValid=false;
}else if (birthdayData.name.length >100){
    errors.name="Navnet overstiger makslengden på 100 tegn."
    isValid=false;
}
```

Figure 8

The input data is also checked against Cross-site Scripting (XSS) attacks. The team members decided to use the “DOMPurify” library. It is a JavaScript that can sanitize HTML and prevent XSS attacks. Figure 9 is an example showing how the input data has been checked before submission.



```
const sanitizeBirthdayFields = () => {
    const { name,email,tlf,horseOptions,numberOfChildren,age,comment} = birthdayData;

    birthdayData.name =DOMPurify.sanitize(name);
    birthdayData.email = DOMPurify.sanitize(email);
    birthdayData.tlf= DOMPurify.sanitize(tlf);
    birthdayData.horseOptions=DOMPurify.sanitize(horseOptions)
    birthdayData.numberOfChildren=DOMPurify.sanitize(numberOfChildren)
    birthdayData.age=DOMPurify.sanitize(age)
    birthdayData.comment=DOMPurify.sanitize(comment)
};
```

Figure 9

# 8 Installation and Running

## 8.1 External dependencies

**1 Firebase v9** is a development platform developed by Google that provides file storage, hosting, database, authentication, and analytics. Firebase is free, provides an SSL certificate by default, and offers impressive speed across multiple regions [2].

Terminal command:
*npm install firebase* (to install the package on the React project)

**2 Redux/toolkit** is a predictable state container for JavaScript apps. It helps manage the state of an application and makes it easier to develop and maintain complex applications. It includes tools for creating reducer functions, creating and managing slices of state, and configuring the store.

Terminal commands:
*npm install @reduxjs/toolkit* (to install the package)

*npm install react-redux* (to bind with React)

**3 React Router v6** is a popular library for implementing routing functionality in React applications. It allows the application to define routes for different URLs and render various components depending on the active route.

Terminal commands:
*npm install react-router-dom* (to install the package)

## 8.2 Running the Application

Terminal commands:
*npm start* (to run the application in development mode)

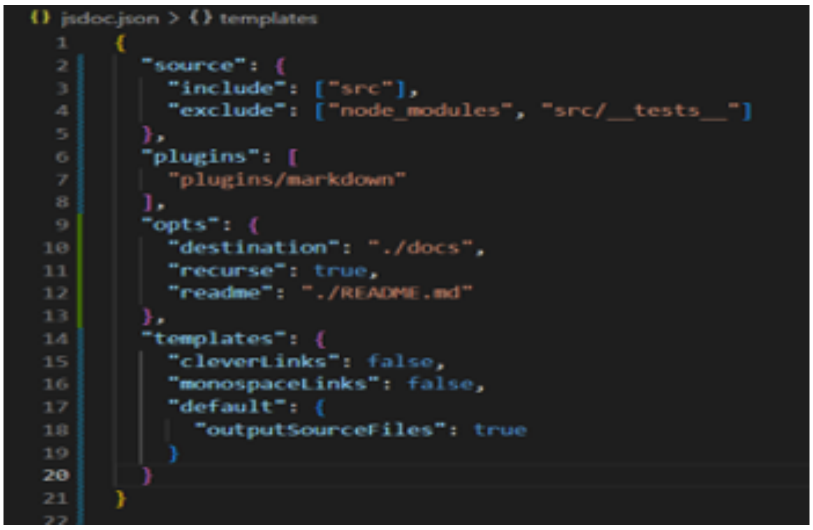
*npm test* (to launch the test runner in the interactive watch mode)

# 9 Documentation of Source Code

JSDoc is used to generate documentation. It is a markup language used to annotate JavaScript source code files. Until to use this library, these steps are followed:

1. Installing the library by using the terminal command "npm install -g jsdoc."

2. Creating a configuration file "jsdoc.json" in the root of the project to include just the "src" directory and generate the documentation in the directory "./docs."



```
{} jsdoc.json > {} templates
{
  "source": {
    "include": ["src"],
    "exclude": ["node_modules", "src/__tests__"]
  },
  "plugins": [
    "plugins/markdown"
  ],
  "opts": {
    "destination": "./docs",
    "recurse": true,
    "readme": "./README.md"
  },
  "templates": {
    "cleverLinks": false,
    "monospaceLinks": false,
    "default": {
      "outputSourceFiles": true
    }
  }
}
```

Figure 10

3. Writing documentation of source code.

4. Generate documentation by using the command line "jsdoc -c jsdoc.json"

To open the generated code in the browser, open the docs folder in the project's directory, then open "index.html" in the local host on port 5500.

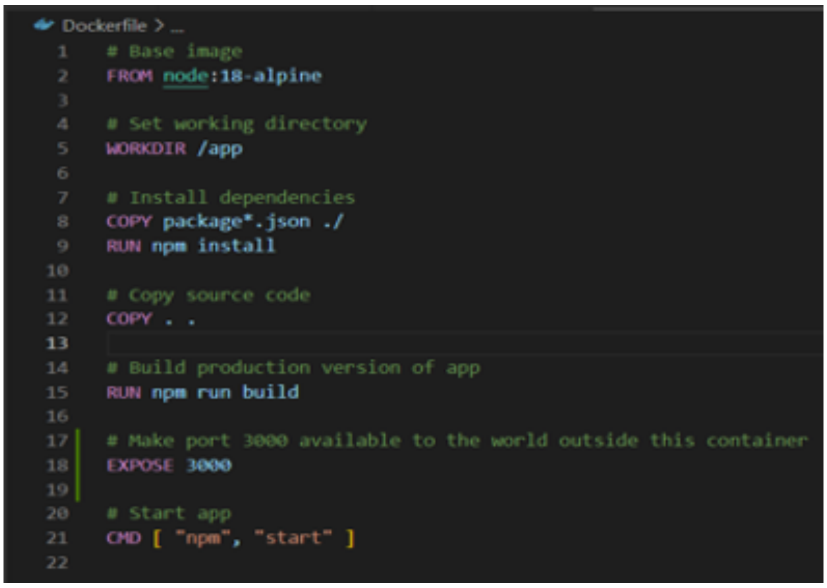
URL: http://127.0.0.1:5500/docs/index.html

# 10 Continuous Integration and Testing

## 10.1 Continuous Integration

The team members decided to set up continuous integration using Docker and GitHub Actions. The following steps have been taken to do this:

1. Create a "Dockerfile" in the project's root, which will be used to create a docker image. And then push the file to the GitHub repository.

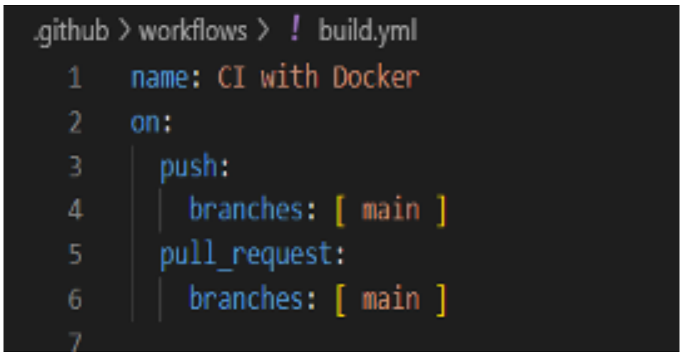


```
Dockerfile > ...
# Base image
FROM node:18-alpine

# Set working directory
WORKDIR /app

# Install dependencies
COPY package*.json ./
RUN npm install

# Copy source code
COPY . .

# Build production version of app
RUN npm run build

# Make port 3000 available to the world outside this container
EXPOSE 3000

# Start app
CMD [ "npm", "start" ]
```

Figure 11

2. A directory named ". github/workflows" containing the "build. yaml" file was created in the project's root.

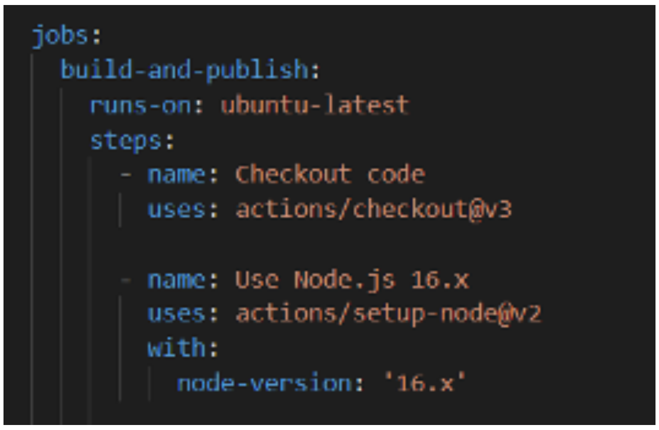
   a. The name of the workflow is "CI with Docker.". The workflow is triggered by a push to the "main" branch or when pulled from the "main" branch.



```
.github > workflows > ! build.yml
name: CI with Docker
on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]
```

Figure 12

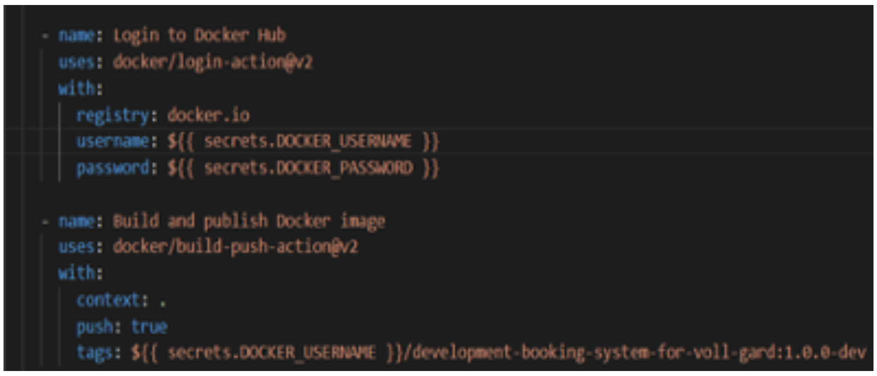
b. This workflow contains one job which runs on the "Ubuntu" operating system. This job includes the Checkout code using the "actions/checkout" action. Set up Node.js version 16. x.



```
jobs:
  build-and-publish:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Use Node.js 16.x
        uses: actions/setup-node@v2
        with:
          node-version: '16.x'
```

Figure 13

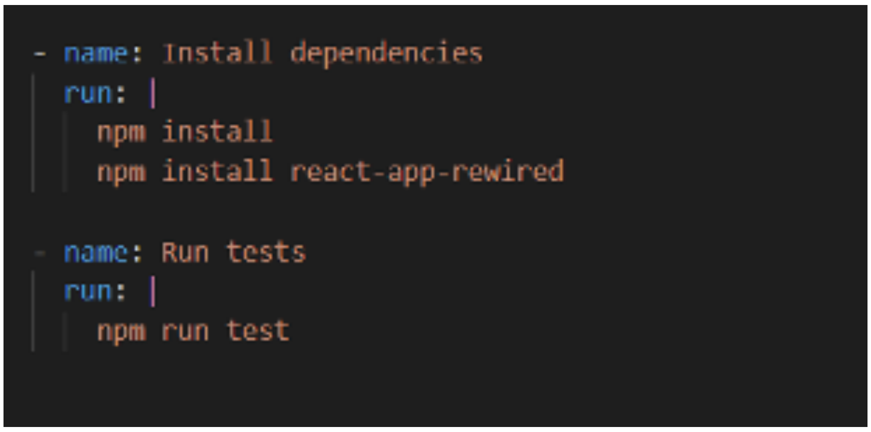
c. Login to Docker Hub using the ‘docker/login action. Then build the Docker image using the "build-push-action@v2".



```
- name: Login to Docker Hub
  uses: docker/login-action@v2
  with:
    registry: docker.io
    username: ${{ secrets.DOCKER_USERNAME }}
    password: ${{ secrets.DOCKER_PASSWORD }}

- name: Build and publish Docker image
  uses: docker/build-push-action@v2
  with:
    context: .
    push: true
    tags: ${{ secrets.DOCKER_USERNAME }}/development-booking-system-for-voll-gard:1.0.0-dev
```

Figure 14

d. Install the dependency and run an "npm run test" to automate building and testing.



```
- name: Install dependencies
  run: |
    npm install
    npm install react-app-rewired

- name: Run tests
  run: |
    npm run test
```

Figure 15

## 10.2 Testing

The team members used several libraries to test code:

a. '@testing-library/react' library to test React components by simulating how the user interacts with the component.

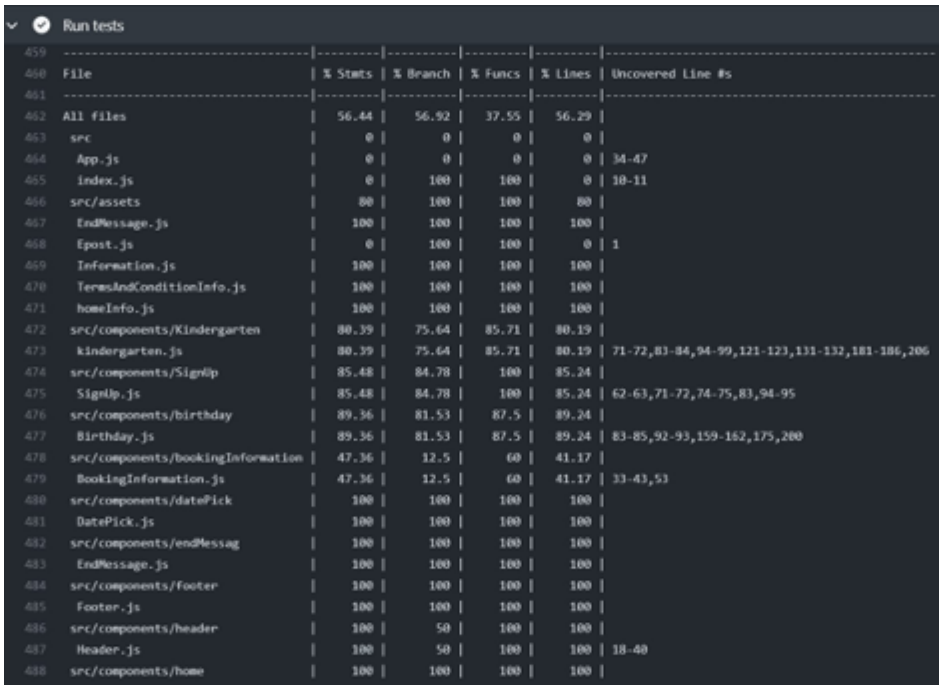
b. “redux-mock-store” library to test Redux logic without setting up an entire Redux store.

c. “Jest” library is a JavaScript testing framework designed to ensure the correctness of any JavaScript codebase [3].

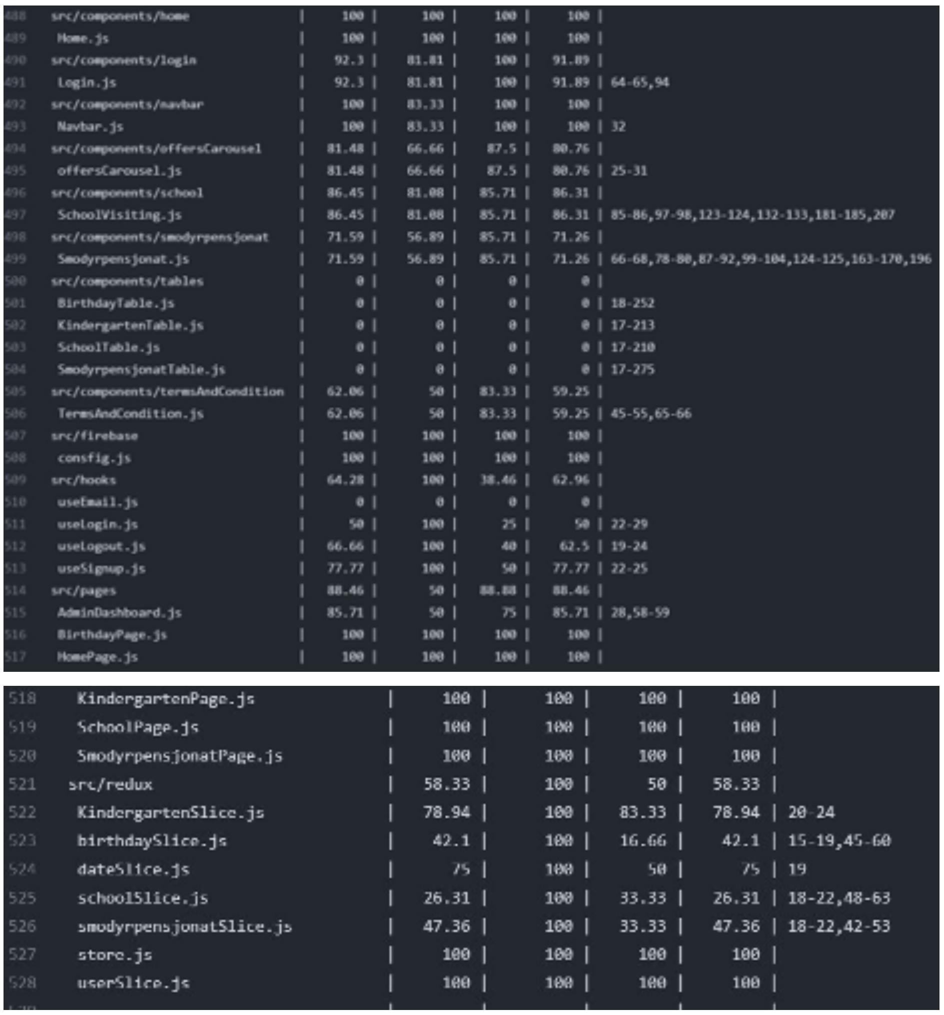
Terminal command:
*npm test* (to run the testing)

### 10.2.1 Test Coverage

The test coverage is about 56.4% of all SRC folders.



```
----------------------------------|---------|----------|---------|---------|------------------------------------------
File                              | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------------------|---------|----------|---------|---------|------------------------------------------
All files                         |   56.44 |    56.92 |   37.55 |   56.29 |
 src                              |       0 |        0 |       0 |       0 |
  App.js                          |       0 |        0 |       0 |       0 | 34-47
  index.js                        |       0 |      100 |     100 |       0 | 10-11
 src/assets                       |      80 |      100 |     100 |      80 |
  EndMessage.js                   |     100 |      100 |     100 |     100 |
  Epost.js                        |       0 |      100 |     100 |       0 | 1
  Information.js                  |     100 |      100 |     100 |     100 |
  TermsAndConditionInfo.js        |     100 |      100 |     100 |     100 |
  homeInfo.js                     |     100 |      100 |     100 |     100 |
 src/components/Kindergarten      |   80.39 |    75.64 |   85.71 |   80.19 |
  kindergarten.js                 |   80.39 |    75.64 |   85.71 |   80.19 | 71-72,83-84,94-99,121-123,131-132,181-186,206
 src/components/SignUp            |   85.48 |    84.78 |     100 |   85.24 |
  SignUp.js                       |   85.48 |    84.78 |     100 |   85.24 | 62-63,71-72,74-75,83,94-95
 src/components/birthday          |   89.36 |    81.53 |    87.5 |   89.24 |
  Birthday.js                     |   89.36 |    81.53 |    87.5 |   89.24 | 83-85,92-93,159-162,175,200
 src/components/bookingInformation |   47.36 |     12.5 |      60 |   41.17 |
  BookingInformation.js           |   47.36 |     12.5 |      60 |   41.17 | 33-43,53
 src/components/datePick          |     100 |      100 |     100 |     100 |
  DatePick.js                     |     100 |      100 |     100 |     100 |
 src/components/endMessag         |     100 |      100 |     100 |     100 |
  EndMessage.js                   |     100 |      100 |     100 |     100 |
 src/components/footer            |     100 |      100 |     100 |     100 |
  Footer.js                       |     100 |      100 |     100 |     100 |
 src/components/header            |     100 |       50 |     100 |     100 |
  Header.js                       |     100 |       50 |     100 |     100 | 18-40
 src/components/home              |     100 |      100 |     100 |     100 |
```



```
 src/components/home                |     100 |      100 |     100 |     100 |
  Home.js                           |     100 |      100 |     100 |     100 |
 src/components/login               |    92.3 |    81.81 |     100 |   91.89 |
  Login.js                          |    92.3 |    81.81 |     100 |   91.89 | 64-65,94
 src/components/navbar              |     100 |    83.33 |     100 |     100 |
  Navbar.js                         |     100 |    83.33 |     100 |     100 | 32
 src/components/offersCarousel      |   81.48 |    66.66 |    87.5 |   80.76 |
  offersCarousel.js                 |   81.48 |    66.66 |    87.5 |   80.76 | 25-31
 src/components/school              |   86.45 |    81.08 |   85.71 |   86.31 |
  SchoolVisiting.js                 |   86.45 |    81.08 |   85.71 |   86.31 | 85-86,97-98,123-124,132-133,181-185,207
 src/components/smodyrpensjonat     |   71.59 |    56.89 |   85.71 |   71.26 |
  Smodyrpensjonat.js                |   71.59 |    56.89 |   85.71 |   71.26 | 66-68,78-80,87-92,99-104,124-125,163-170,196
 src/components/tables              |       0 |        0 |       0 |       0 |
  BirthdayTable.js                  |       0 |        0 |       0 |       0 | 18-252
  KindergartenTable.js              |       0 |        0 |       0 |       0 | 17-213
  SchoolTable.js                    |       0 |        0 |       0 |       0 | 17-210
  SmodyrpensjonatTable.js           |       0 |        0 |       0 |       0 | 17-275
 src/components/termsAndCondition   |   62.06 |       50 |   83.33 |   59.25 |
  TermsAndCondition.js              |   62.06 |       50 |   83.33 |   59.25 | 45-55,65-66
 src/firebase                       |     100 |      100 |     100 |     100 |
  consfig.js                        |     100 |      100 |     100 |     100 |
 src/hooks                          |   64.28 |      100 |   38.46 |   62.96 |
  useEmail.js                       |       0 |        0 |       0 |       0 |
  useLogin.js                       |      50 |      100 |      25 |      50 | 22-29
  useLogout.js                      |   66.66 |      100 |      40 |    62.5 | 19-24
  useSignup.js                      |   77.77 |      100 |      50 |   77.77 | 22-25
 src/pages                          |   88.46 |       50 |   88.88 |   88.46 |
  AdminDashboard.js                 |   85.71 |       50 |      75 |   85.71 | 28,58-59
  BirthdayPage.js                   |     100 |      100 |     100 |     100 |
  HomePage.js                       |     100 |      100 |     100 |     100 |
```

```
  KindergartenPage.js       |     100 |      100 |     100 |     100 |
  SchoolPage.js             |     100 |      100 |     100 |     100 |
  SmodyrpensjonatPage.js    |     100 |      100 |     100 |     100 |
 src/redux                  |   58.33 |      100 |      50 |   58.33 |
  KindergartenSlice.js      |   78.94 |      100 |   83.33 |   78.94 | 20-24
  birthdaySlice.js          |    42.1 |      100 |   16.66 |    42.1 | 15-19,45-60
  dateSlice.js              |      75 |      100 |      50 |      75 | 19
  schoolSlice.js            |   26.31 |      100 |   33.33 |   26.31 | 18-22,48-63
  smodyrpensjonatSlice.js   |   47.36 |      100 |   33.33 |   47.36 | 18-22,42-53
  store.js                  |     100 |      100 |     100 |     100 |
  userSlice.js              |     100 |      100 |     100 |     100 |
```

Figure 16: Test coverage.

The folders have been tested as follows:

- The component folders containing the most reusable components such as Birthday, Kindergarten, Smodyrpensjonat, termsAndCondition, and other components.

- Page folder, which includes AdminDashboard, BirthdayPage, and others.

- Redux folder, which includes all slices such as userSlice, birthdaySlice, smodyrpensjonatSlice, dateSlice,

- The source code that still needs to be tested is all the code relating to the Firebase backend platform, which was challenging for the team for several reasons. Firebase does real-time synchronization; the testing did not stop properly. Additionally, the database in Firebase is a live database that cannot be mocked through testing. The team uses two services of the Firebase platform, authentication, and Firebase Firestore. Both need different libraries and tools to be tested. Still, the team needs more time or experience to deal with those complexities.

# 11 References

[1] “Serverless Web Applications with React and Firebase : Develop Real-time Applications for Web and Mobile Platforms.” https://web.s.ebscohost.com/ehost/ebookviewer/ebook/bmxlYmtfXzE3ODk0ODlfX0FO0?sid=a9f83129-715e-483f-86ae-513509383c4d@redis&vid=0&format=EB&lpid=lp_168&rid=0 (accessed Apr. 23, 2023).

[2] “Automate deployment of React applications to Firebase,” *CircleCI*, Sep. 06, 2022. https://circleci.com/blog/deploy-react-apps-to-firebase/ (accessed May 02, 2023).

[3] “Jest.” https://jestjs.io/ (accessed May 10, 2023).

[4] “Dockerizing a React App.” https://mherman.org/blog/dockerizing-a-react-app/ (accessed Apr. 23, 2023).

[5] “Workflow syntax for GitHub Actions,” *GitHub Docs*. https://ghdocs-prod.azurewebsites.net/_next/data/bGD1141Cu6hDBSEEWOMUh/en/free-pro-team@latest/actions/using-workflows/workflow-syntax-for-github-actions.json?versionId=free-pro-team%40latest&productId=actions&restPage=using-workflows&restPage=workflow-syntax-for-github-actions (accessed Apr. 23, 2023).

[6] “React Testing Library | Testing Library,” Aug. 09, 2022. https://testing-library.com/docs/react-testing-library/intro/ (accessed Apr. 23, 2023).

[7] “Getting Started | Redux Toolkit,” Oct. 30, 2022. https://redux-toolkit.js.org/introduction/getting-started (accessed May 02, 2023).

## E Usability Testing

123

# Utvikling av et bookingsystem for Voll gård
## Brukertesting

VOLL GÅRD
HELE BYENS BONDEGÅRD

NTNU
Kunnskap for en bedre verden

# Innholdsfortegnelse

**Informasjon om testingen** **2**
**Spørsmål til testpersoner** **3**
**Brukertest 1 - Smådyrpensjonat** **4**
Før test 4
Under test 4
Etter test 5
**Brukertest 2 - Bursdagsfeiring** **6**
Før test 6
Under test 6
Etter test 7
**Brukertest 3 - Skolebesøk** **8**
Før test 8
Under test 8
Etter test 9

## Informasjon om testingen

Hensikten med å gjennomføre denne brukertestingen er å se om systemet er designet på en tilfredsstillende måte; at det er ryddig, intuitivt, funksjonelt og brukervennlig for potensielle kunder. Tilbakemeldingene fra testpersonene vil kunne være til stor nytte for videre utvikling, da de ser ting med nye øyne sammenlignet med oss som sitter og jobber med det dag inn og dag ut.

Systemet som skal testet er et bookingsystem utviklet som en webapplikasjon for Stiftelsen Voll gård. I første omgang skal testingen gå ut på å gjennomføre en booking av de ulike tjenestene som tilbys.

Testpersonene er potensielle kunder av de ulike opplevelsene og tjenestene som tilbys via bookingsystemet per dags dato. Disse tjenestene er bursdagsfeiring, smådyrpensjonat og skole- og barnehagebesøk. Relevante testpersoner kan da være foreldre til barn i barneskolealder, dyreeiere, førskolelærere og lærere.

Utstyret som trengs for å kunne gjennomføre testene er en PC og kildekoden som kjøres gjennom Visual Studio Code og localhost. Utviklerens egen PC skal benyttes. Testene gjennomføres fysisk såfremt det er mulig, hvis ikke kan det gjennomføres en digital fjerntesting av wireframes. Det estimeres at hver test tar ca. 15-20 minutter, avhengig av hvor mye tilbakemelding testpersonen kommer med.

Under testingen bes testpersonen om å tenke høyt, samtidig som utvikleren som utfører testen tar opptak eller notater av det som blir sagt. I tillegg er det formulert spørsmål som skal stilles både før og etter gjennomføringen av testen. Disse spørsmålene er presentert i neste seksjon.

## Spørsmål til testpersoner

Følgende spørsmål stilles til testpersonen før gjennomføring av testen:

1. Fortell litt om deg selv og hvorfor du er en relevant testperson.

2. Har du tidligere booket en opplevelse/tjeneste hos Voll gård? Hvis ja:
    a. Hvilken opplevelse/tjeneste?
    b. Var det noe som utmerket seg positivt i bookingprosessen den gangen?
    c. Var det noe som utmerket seg negativt i bookingprosessen den gangen?

3. Har du booket lignende opplevelser/tjenester andre steder? Hvis ja:
    a. Hvor og hvilken opplevelse/tjeneste?
    b. Var det noe som utmerket seg positivt i bookingprosessen den gangen?
    c. Var det noe som utmerket seg negativt i bookingprosessen den gangen?

4. Hva er viktig for deg når det gjelder denne typen booking?

5. Hva er viktig for deg når det gjelder booking generelt?

6. Hva skal til for at du heller velger å bruke en web-basert løsning fremfor e-postkommunikasjon eller en telefonsamtale?

Følgende spørsmål stilles til testpersonen etter gjennomføring av testen:

1. Hva er førsteinntrykket ditt av bookingsystemet?

2. Var det noe som utmerket seg positivt?

3. Var det noe som utmerket seg negativt?

4. Er det noe du synes burde vært gjort annerledes?

5. Er det noe du synes mangler?

6. Hva tenker du om å bruke en tilsvarende webapplikasjon i motsetning til dagens system over e-post og telefonsamtaler?

7. Andre kommentarer?

# Brukertest 1 - Smådyrpensjonat

Testen ble gjennomført fysisk med en dame i midten av 40-årene, som også er dyreeier og en aktuell kandidat til å benytte seg av smådyrpensjonatet på gården.

## Før test

1. Fortell litt om deg selv og hvorfor du er en relevant testperson.
   Jeg har en kanin som trenger et sted å være dersom jeg ønsker å reise bort. Det er vanskelig å finne noen som har mulighet og rette forutsetninger for å kunne passe henne, og dette har derfor vært en begrensning for meg.

2. Har du tidligere booket en opplevelse/tjeneste hos Voll gård?
   Nei, jeg har kun vært på gården i forbindelse med konfirmasjon i utleielokalene.

3. Har du booket lignende opplevelser/tjenester andre steder?
   Nei, jeg visste ikke om tilbudet til Voll gård, og det har ikke falt meg inn at det fantes slike tilbud i Trondheim.

4. Hva er viktig for deg når det gjelder denne typen booking?
   Det er viktig for meg at nettsiden inneholder all viktig informasjon. For meg er dette priser, informasjon om hva jeg må ta med og informasjon om hvordan kaninen vil bli tatt hånd om. Det er definitivt en fordel at tjenesten befinner seg i nærheten av sentrum på et sted med god bussforbindelse, da jeg ikke har bil.

5. Hva er viktig for deg når det gjelder booking generelt?
   At det går kjapt og smidig og at nettsiden er lett å forstå seg på. Et stort pluss er et moderne og ryddig design, men det er jo veldig individuelt hva man definerer som det.

6. Hva skal til for at du heller velger å bruke en web-basert løsning fremfor e-postkommunikasjon eller en telefonsamtale?
   Relativt lite, da jeg foretrekker web-baserte løsninger. Dersom opplegget ser seriøst ut og nødvendig informasjon oppgis, er jeg fornøyd.

## Under test

Testingen starter på forsiden av webapplikasjonen. Denne er foreløpig tom. Testpersonen blir så bedt om å utføre en booking av et opphold på smådyrpensjonatet.

Notater fra gjennomføringen:

- Ser en tom forside. Lurer på om det er meningen, eller om det er noe feil.
- Får beskjed om å booke opphold på smådyrpensjonat. Ser menyen i headeren og velger "smådyrpensjonat".
- "Oi, der ja!". Kommer til informasjonssiden for smådyrpensjonat. Leser gjennom alt. Lurer på hvordan det er dersom dyret blir sykt mens det er på gården.
- Blir bedt om å velge dato. Velger dato og forventer at det skal dukke opp en ny datovelger der man velger dato for henting. Det skjer ikke.

- Får beskjed om å klikke seg videre. Kommer til siden for utfylling av informasjon om dyr og eier. "Åå, går det an å låne bur? Hvilken størrelse er disse, da? Koster det noe?"
- Får beskjed om å fylle ut informasjon og klikke seg videre. Trykker automatisk på knappen til høyre uten å se at det står "tilbake", og blir sendt til infosiden. Velger dato og trykker seg tilbake igjen. "Må jeg fylle ut alt på nytt nå?".
- Fyller ut all informasjonen igjen og klikker seg videre, denne gangen på knappen til venstre der det står "gå videre". Mumler at knappene er plassert i feil rekkefølge.
- På neste side kommer en oppsummering av utfylt informasjon. Brukeren bekrefter bookingen og blir sendt rett til førstesiden. "Hva skjedde nå? Ble bookingen gjennomført, eller må jeg gjøre noe på nytt? Får jeg noe bekreftelse?"

## Etter test

1. Hva er førsteinntrykket ditt av bookingsystemet?
   Ryddig og enkelt å forstå seg på!

2. Var det noe som utmerket seg positivt?
   Likte godt informasjonen om tjenesten - synes den var utfyllende og jeg følte ikke at jeg manglet noe informasjon for å "trygt" kunne legge inn en booking.

3. Var det noe som utmerket seg negativt?
   Knappene "tilbake" og "gå videre" var plassert i motsatt rekkefølge til det jeg er vant til. Problematisk at all informasjonen man har skrevet inn blir borte, så kanskje lurt å endre rekkefølge. Det var også rart at man ikke måtte velge hvor langt oppholdet skal være, og å ikke få bekreftelse på om bestillingen ble mottatt.

4. Er det noe du synes burde vært gjort annerledes?
   Designet er noe enkelt, men fikk opplyst på starten at dette bare er første versjon av systemet, så det er forståelig.

5. Er det noe du synes mangler?
   Det hadde vært kjekt å få oppgitt prisen på oppholdet i oppsummeringen på slutten, basert på antall netter og eventuelle tilleggstjenester. Jeg prøvde i hvert fall å regne ut hvor mye det dreide seg om, og det er sikkert flere i min situasjon.

6. Hva tenker du om å bruke en tilsvarende webapplikasjon i motsetning til dagens system over e-post og telefonsamtaler?
   Jeg er overhodet ikke glad i å prate i telefon, så for meg er en nettbasert løsning det aller beste. Det går greit over e-post også, men da burde det i så fall være enkelt å finne informasjon om tjenesten på Facebook eller hjemmesiden, slik at jeg slipper å spørre om alt jeg lurer på før jeg setter dyret mitt bort til dem.

7. Andre kommentarer?
   Nei.

# Brukertest 2 - Bursdagsfeiring

Testen ble gjennomført fysisk med en mann på rundt 40 år. Han er far til barn i barneskolealder, og har vært gjest i bursdagsfeiring på gården.

## Før test

1. Fortell litt om deg selv og hvorfor du er en relevant testperson.
   Jeg har to barn på barneskolen, som begge har vært i flere bursdagsfeiringer til klassekamerater på Voll gård, og selv maser om å få feire der.

2. Har du tidligere booket en opplevelse/tjeneste hos Voll gård?
   Nei, men jeg har vært på gården i bursdagsfeiring og vi benytter oss flittig av åpen gård på søndager.

3. Har du booket lignende opplevelser/tjenester andre steder? Ja.
   a. Hvor og hvilken opplevelse/tjeneste?
   Barna har feiret bursdager på Leos lekeland og Rush trampolinepark.

   b. Var det noe som utmerket seg positivt i bookingprosessen den gangen?
   Veldig kjekt å kunne sjekke hvilke datoer som er ledig å bestille med én gang!

   c. Var det noe som utmerket seg negativt i bookingprosessen den gangen?
   Spesielt Leos lekeland har en glorete og rotete nettside. Det var vanskelig å navigere og alt for mye informasjon og mange valg.

4. Hva er viktig for deg når det gjelder denne typen booking?
   Jeg synes det er viktig at prisene presenteres tidlig i prosessen, og at informasjon om oppmøte, opplegg og vilkår i tilfelle sykdom og avbestilling legges fram.

5. Hva er viktig for deg når det gjelder booking generelt?
   At det ikke er for mye (unødvendig) informasjon og at det går raskt.

6. Hva skal til for at du heller velger å bruke en web-basert løsning fremfor e-postkommunikasjon eller en telefonsamtale?
   Jeg synes det er ganske lettvint å ta en telefon til stedet for å legge inn en booking eller få svar på spørsmål, fremfor å måtte lete etter svar på nettsiden. Så det er i hvert fall viktig for meg at jeg finner det jeg trenger av informasjon, hvis ikke er jeg rask til å ringe.

## Under test

Testingen starter på forsiden av webapplikasjonen. Denne er foreløpig tom. Testpersonen blir så bedt om å utføre en booking av bursdagsfeiring.

- Får beskjed om å klikke seg inn på siden for bursdagsfeiring. Ser en tom forside, sier ikke noe på det. Velger "bursdagsfeiring" fra menyen i toppen. Kommer inn på infosiden om bursdagsfeiring. Leser gjennom infoen og velger dato. "Er alle disse ledig?". Trykker på en dato og lurer på om det ikke er mulig å velge klokkeslett. Klikker seg videre.

- Kommer til siden for å fylle ut informasjon. Fyller ut informasjon og klikker seg videre. Bemerker at knappene er plassert i rar rekkefølge. Kommer til oppsummeringen og ser at e-postadressen er feil, klikker seg tilbake for å endre og ser at all input er borte. Virker litt oppgitt over å måtte taste inn alt på nytt. Bekrefter bookingen og får beskjed om at bookingen er bekreftet. Lurer på om bekreftelsen sendes på mail i tillegg, står ingenting om det.

## Etter test

1. Hva er førsteinntrykket ditt av bookingsystemet?
   Ser ryddig ut og inneholder den viktigste informasjonen. Lett å finne fram.

2. Var det noe som utmerket seg positivt?
   God informasjon om opplegget, likte at prisliste og tidsskjema var inkludert.

3. Var det noe som utmerket seg negativt?
   Rart å ikke skulle velge klokkeslett eller få noe informasjon om hvilke klokkeslett som er mulige.

4. Er det noe du synes burde vært gjort annerledes?
   Forvirrende plassering av knapper.

5. Er det noe du synes mangler?
   Det bør være mulig å se ledige klokkeslett på ledige datoer. Nå står alle datoer som tilgjengelig, og det er ikke mulig å velge noe klokkeslett. Hadde ønsket bekreftelse på mail eller at det var mulig å laste ned. Hadde også vært fint med illustrasjoner!

6. Hva tenker du om å bruke en tilsvarende webapplikasjon i motsetning til dagens system over e-post og telefonsamtaler?
   Såfremt alt kommer i orden, var dette et veldig enkelt og greit bookingsystem som jeg godt kunne brukt i stedet for telefonsamtaler.

7. Andre kommentarer?
   Nei.

## Brukertest 3 - Skolebesøk

Testen ble gjennomført fysisk med en dame på rundt 30 år. Hun jobber på barneskole.

### Før test

1. Fortell litt om deg selv og hvorfor du er en relevant testperson.
   Jeg jobber på barneskole og har vært på gårdsbesøk med pedagogisk opplegg sammen med skoleklasser flere ganger tidligere. Jeg har i tillegg selv vært med som elev for mange år siden.

2. Har du tidligere booket en opplevelse/tjeneste hos Voll gård? Ja.
   a. Hvilken opplevelse/tjeneste?
   Skolebesøk.

   b. Var det noe som utmerket seg positivt i bookingprosessen den gangen?
   Rask respons over mail, veldig kjekt å bli fulgt opp "personlig" og få svar på alt av spørsmål, til tross for at det blir litt kommunikasjon før man får spikret noe. Vi følte oss godt ivaretatt både før og under besøket!

   c. Var det noe som utmerket seg negativt i bookingprosessen den gangen?
   Det tok litt tid å finne ledig dato, spesielt siden trinnet er stort og vi måtte fordele oss over flere datoer.

3. Har du booket lignende opplevelser/tjenester andre steder?
   Nei.

4. Hva er viktig for deg når det gjelder denne typen booking?
   Det er viktig å få en bekreftelse som kan videresendes til resten av lærerne på trinnet.

5. Hva er viktig for deg når det gjelder booking generelt?
   Det viktigste for meg er at systemet er enkelt og at jeg ikke føler jeg sitter med noen spørsmål som må besvares før besøket.

6. Hva skal til for at du heller velger å bruke en web-basert løsning fremfor e-postkommunikasjon eller en telefonsamtale?
   Jeg bruker gjerne en web-basert løsning dersom kriteriene i punkt 4. og 5. er innfridd. Jeg synes i grunn det er veldig behagelig å gjøre bookinger over nett, kontra over f.eks. telefon.

### Under test

Testingen starter på forsiden av webapplikasjonen. Denne er foreløpig tom. Testpersonen blir så bedt om å utføre en booking av skolebesøk.

- Ser en tom forside og lurer på om systemet har stoppet. Blir bedt om å klikke seg videre for å booke skolebesøk. Ser menyen i headeren og velger "skolebesøk".

- Kommer til infosiden og leser gjennom. Ønsker en oversikt over aktuelle pedagogiske tema. Velger dato og klikker seg videre.
- Blir bedt om å fylle inn informasjon. Ser at det er en dropdown-meny der de ulike pedagogiske temaene er presentert, "oia, der ja!".
- Trykker på et tidspunkt på enter for å komme til neste felt, blir sendt ut av siden. Må klikke seg tilbake og skrive alt på nytt.
- Skal gå videre, trykker på feil knapp og kommer til informasjonssiden igjen. Må fylle ut alt på nytt.
- Klikker seg videre og leser gjennom oppsummeringen. Klikker for å bekrefte bookingen og får opp på skjermen at bookingen er bekreftet. "Kommer infoen på mail også?"

## Etter test

1. Hva er førsteinntrykket ditt av bookingsystemet?
   Veldig intuitivt, måtte ikke lete eller tenke masse for å finne fram.

2. Var det noe som utmerket seg positivt?
   God informasjon på første side.

3. Var det noe som utmerket seg negativt?
   Kjipt å måtte fylle inn informasjonen flere ganger.

4. Er det noe du synes burde vært gjort annerledes?
   Hadde vært kjekt om informasjonen ble lagret selv om man trykker seg fram og tilbake.

5. Er det noe du synes mangler?
   En presentasjon av de pedagogiske oppleggene på informasjonssiden hadde vært fint. Bilder er også bra.

6. Hva tenker du om å bruke en tilsvarende webapplikasjon i motsetning til dagens system over e-post og telefonsamtaler?
   Det stiller jeg meg positiv til!

7. Andre kommentarer?
   Nei.

---

*Testingen ble utført 17.april på en av de tidligste versjonene av webapplikasjonen. På dette tidspunktet var applikasjonen relativ enkel og lik wireframes (som vist i kravdokumentet). Det var ikke gjort særlig mye med designet, og det var heller ikke lagt til illustrasjoner. Flere ting har blitt rettet opp i og endret etter tilbakemeldingene fra testpersonene. Planen var å rekke flere brukertester på et senere tidspunkt, men dette var ikke gjennomførbart.*

# F Views

123

# Utvikling av et bookingsystem for Voll gård
# User Interface - Views

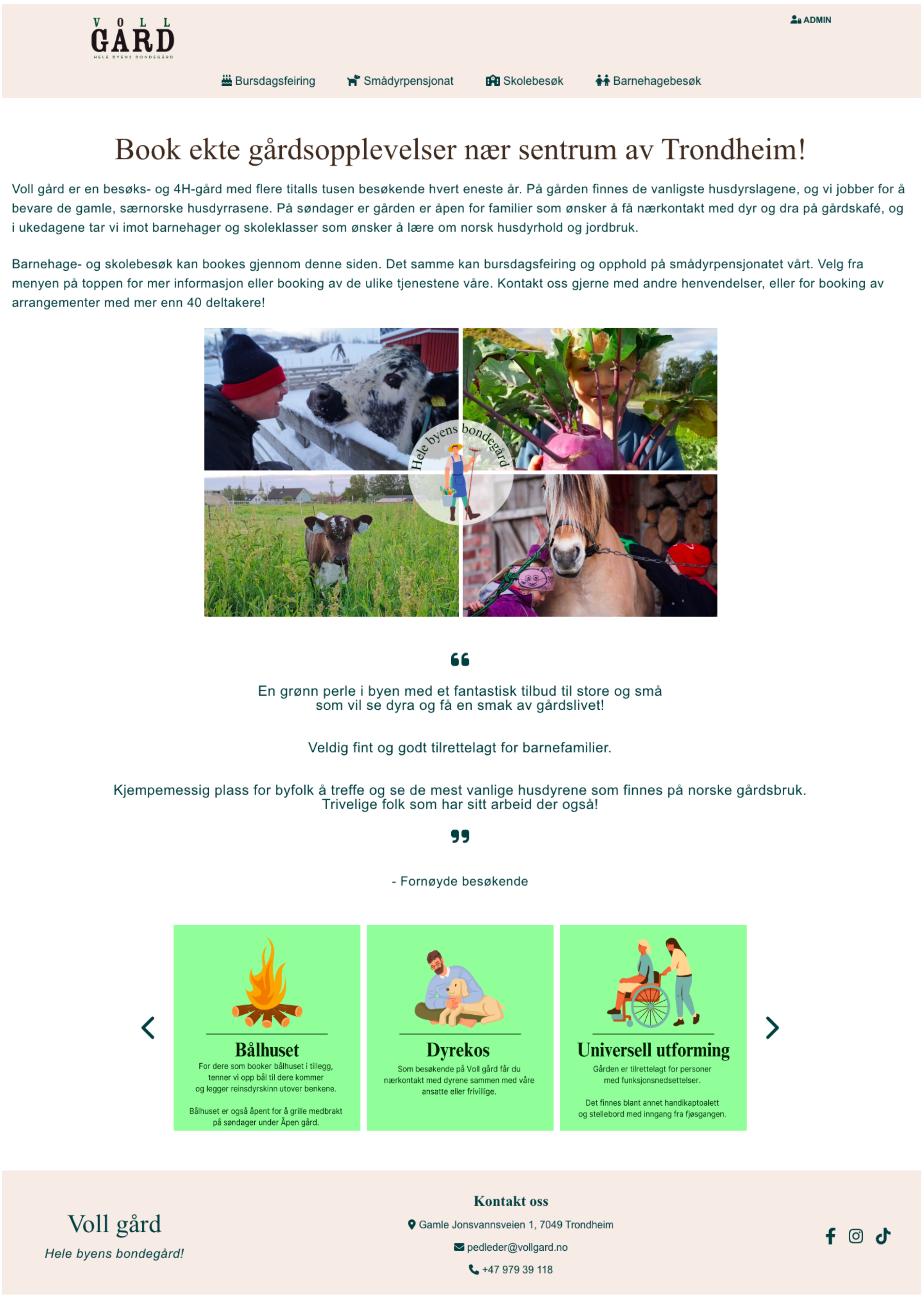
VOLL GÅRD
HELE BYENS BONDEGÅRD

NTNU
Kunnskap for en bedre verden

# Innholdsfortegnelse

Home Page 3
Birthday Celebration 5
School Visits 12
Kindergarten Visits 16
Confirmation View 19

## Home Page



Book ekte gårdsopplevelser nær sentrum av Trondheim!

Voll gård er en besøks- og 4H-gård med flere titalls tusen besøkende hvert eneste år. På gården finnes de vanligste husdyrslagene, og vi jobber for å bevare de gamle, særnorske husdyrrasene. På søndager er gården er åpen for familier som ønsker å få nærkontakt med dyr og dra på gårdskafé, og i ukedagene tar vi imot barnehager og skoleklasser som ønsker å lære om norsk husdyrhold og jordbruk.

Barnehage- og skolebesøk kan bookes gjennom denne siden. Det samme kan bursdagsfeiring og opphold på smådyrpensjonatet vårt. Velg fra menyen på toppen for mer informasjon eller booking av de ulike tjenestene våre. Kontakt oss gjerne med andre henvendelser, eller for booking av arrangementer med mer enn 40 deltakere!

En grønn perle i byen med et fantastisk tilbud til store og små som vil se dyra og få en smak av gårdslivet!

Veldig fint og godt tilrettelagt for barnefamilier.

Kjempemessig plass for byfolk å treffe og se de mest vanlige husdyrene som finnes på norske gårdsbruk. Trivelige folk som har sitt arbeid der også!

- Fornøyde besøkende

Bålhuset

For dere som booker bålhuset i tillegg, tenner vi opp bål til dere kommer og legger reinsdyrskinn utover benkene.

Bålhuset er også åpent for å grille medbrakt på søndager under Åpen gård.

Dyrekos

Som besøkende på Voll gård får du nærkontakt med dyrene sammen med våre ansatte eller frivillige.

Universell utforming

Gården er tilrettelagt for personer med funksjonsnedsettelser.

Det finnes blant annet handikaptoalett og stellebord med inngang fra fjøsgangen.

Voll gård

Hele byens bondegård!

Kontakt oss

Gamle Jonsvannsveien 1, 7049 Trondheim

pedleder@vollgard.no

+47 979 39 118

Figure 1: Home page of the application.

## Admin Pages

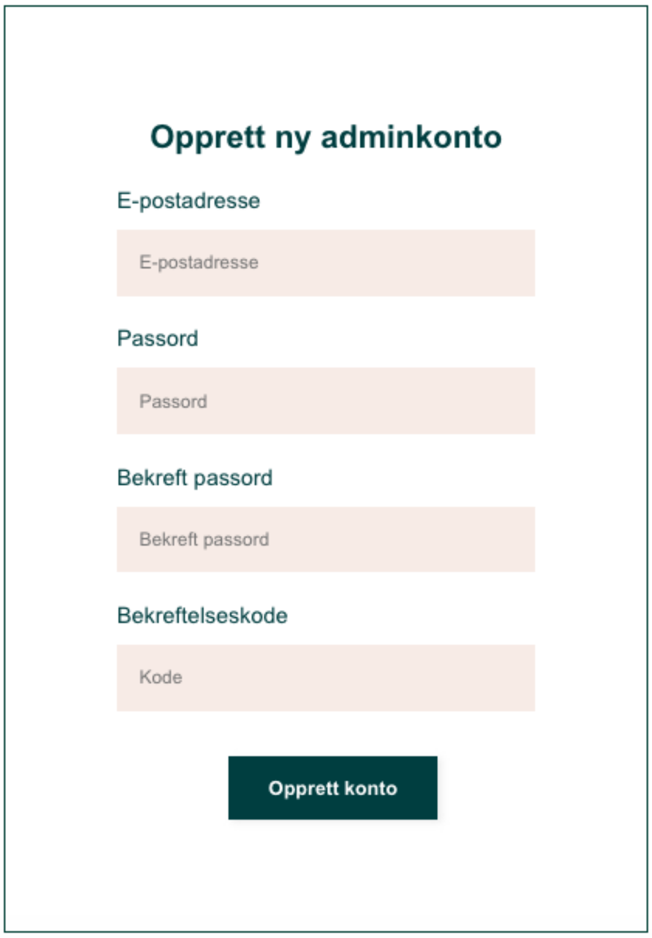


Figure 2: Signup page for new admin users.

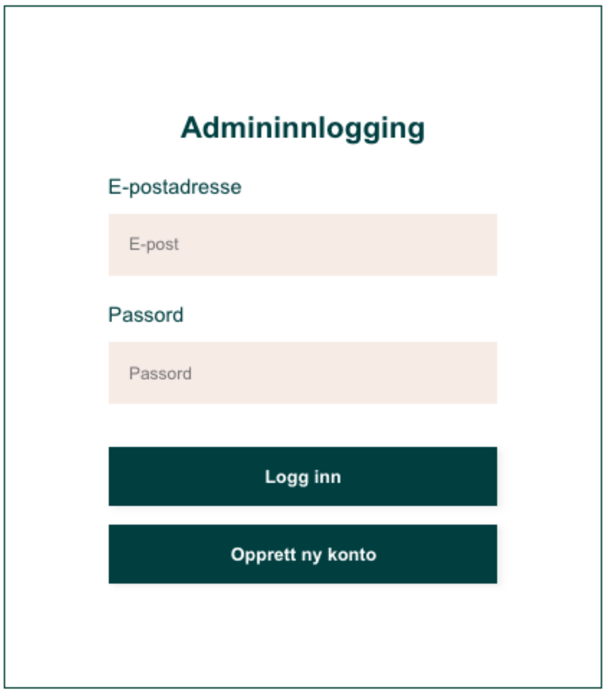
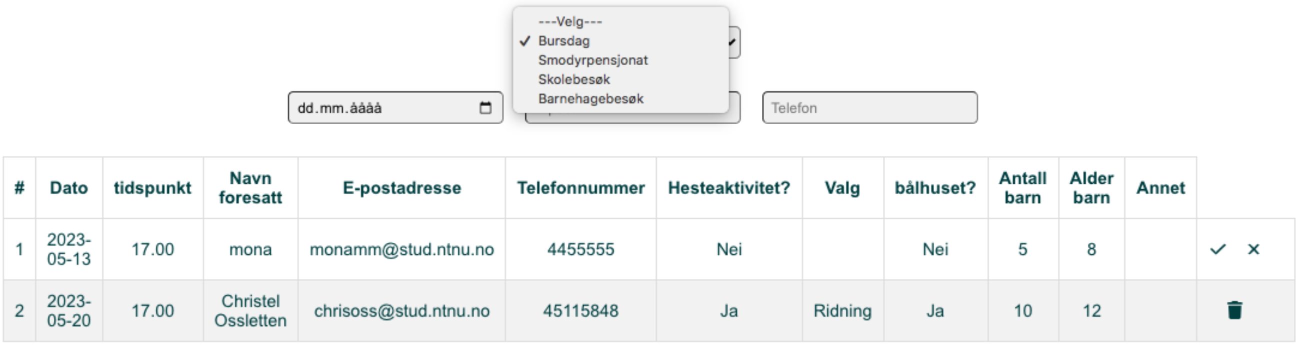


Figure 3: Login page for admin users.

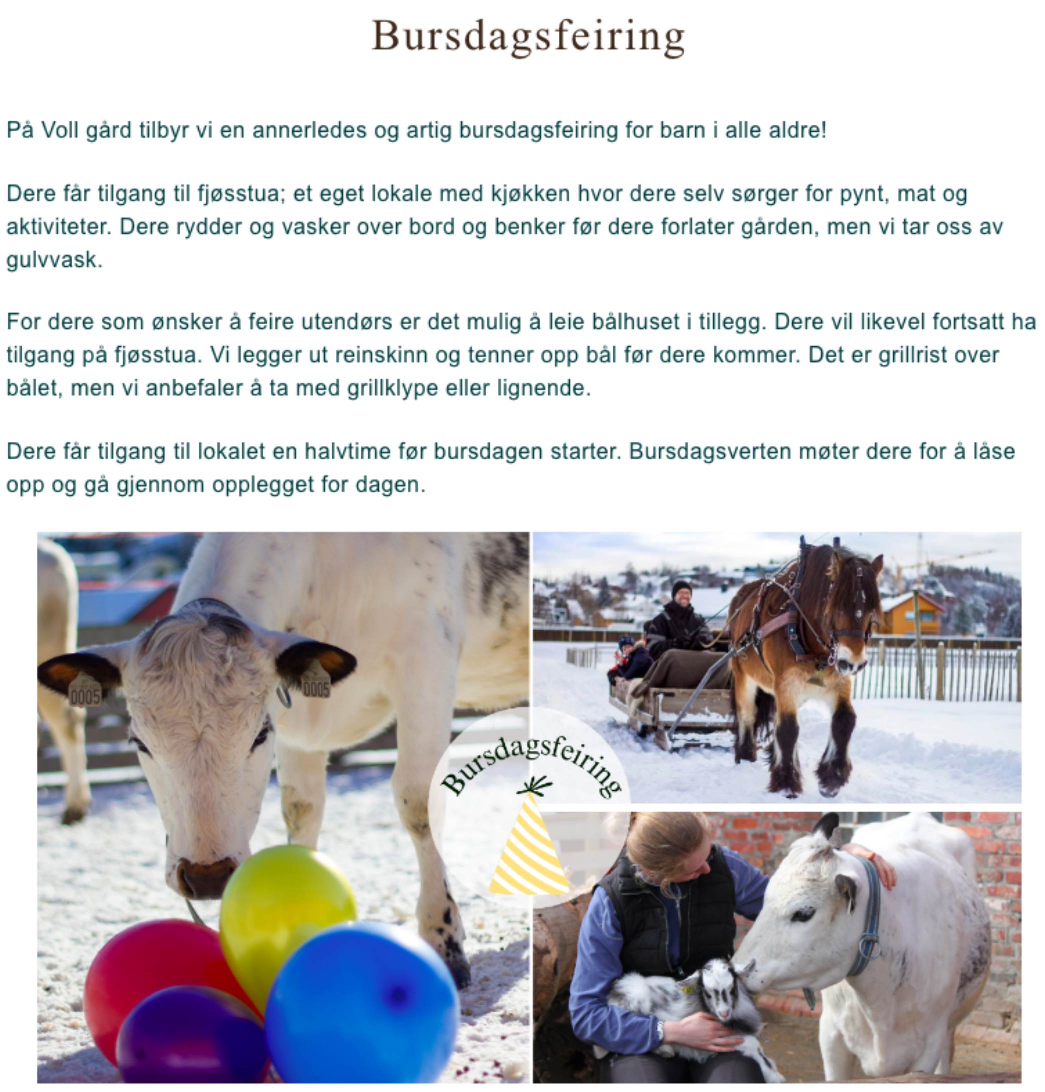


| # | Dato | tidspunkt | Navn foresatt | E-postadresse | Telefonnummer | Hesteaktivitet? | Valg | bålhuset? | Antall barn | Alder barn | Annet | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2023-05-13 | 17.00 | mona | monamm@stud.ntnu.no | 4455555 | Nei | | Nei | 5 | 8 | | ✓ ✕ |
| 2 | 2023-05-20 | 17.00 | Christel Ossletten | chrisoss@stud.ntnu.no | 45115848 | Ja | Ridning | Ja | 10 | 12 | | 🗑 |

Figure 4: Admin dashboard where the administrators can view, modify and cancel bookings. Here filtered by service on birthday celebrations.

## Birthday Celebration

*Note: header and footer components are present on all the following pages.*



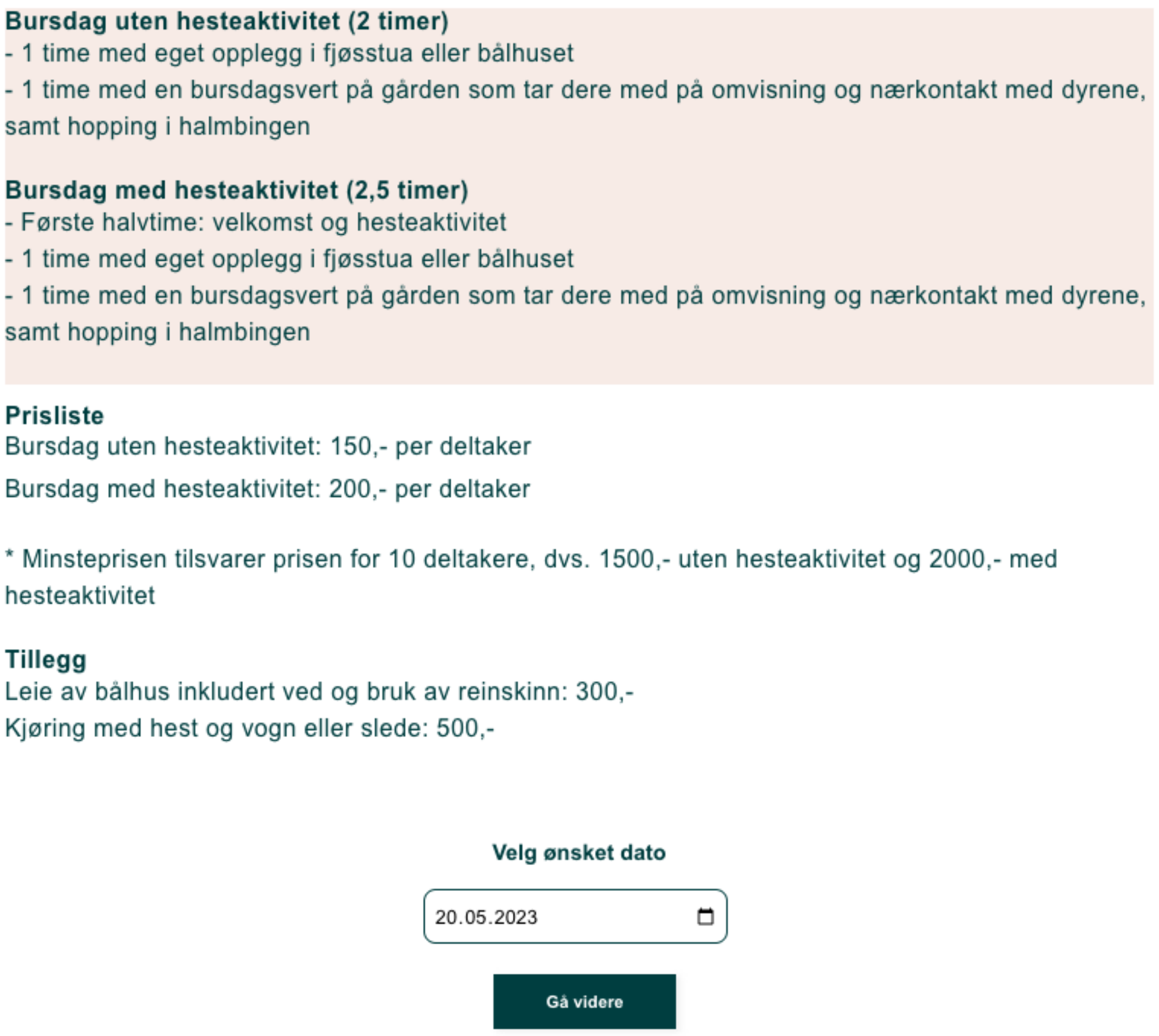
Bursdagsfeiring

På Voll gård tilbyr vi en annerledes og artig bursdagsfeiring for barn i alle aldre!

Dere får tilgang til fjøsstua; et eget lokale med kjøkken hvor dere selv sørger for pynt, mat og aktiviteter. Dere rydder og vasker over bord og benker før dere forlater gården, men vi tar oss av gulvvask.

For dere som ønsker å feire utendørs er det mulig å leie bålhuset i tillegg. Dere vil likevel fortsatt ha tilgang på fjøsstua. Vi legger ut reinskinn og tenner opp bål før dere kommer. Det er grillrist over bålet, men vi anbefaler å ta med grillklype eller lignende.

Dere får tilgang til lokalet en halvtime før bursdagen starter. Bursdagsverten møter dere for å låse opp og gå gjennom opplegget for dagen.



**Bursdag uten hesteaktivitet (2 timer)**
- 1 time med eget opplegg i fjøsstua eller bålhuset
- 1 time med en bursdagsvert på gården som tar dere med på omvisning og nærkontakt med dyrene, samt hopping i halmbingen

**Bursdag med hesteaktivitet (2,5 timer)**
- Første halvtime: velkomst og hesteaktivitet
- 1 time med eget opplegg i fjøsstua eller bålhuset
- 1 time med en bursdagsvert på gården som tar dere med på omvisning og nærkontakt med dyrene, samt hopping i halmbingen

**Prisliste**
Bursdag uten hesteaktivitet: 150,- per deltaker
Bursdag med hesteaktivitet: 200,- per deltaker

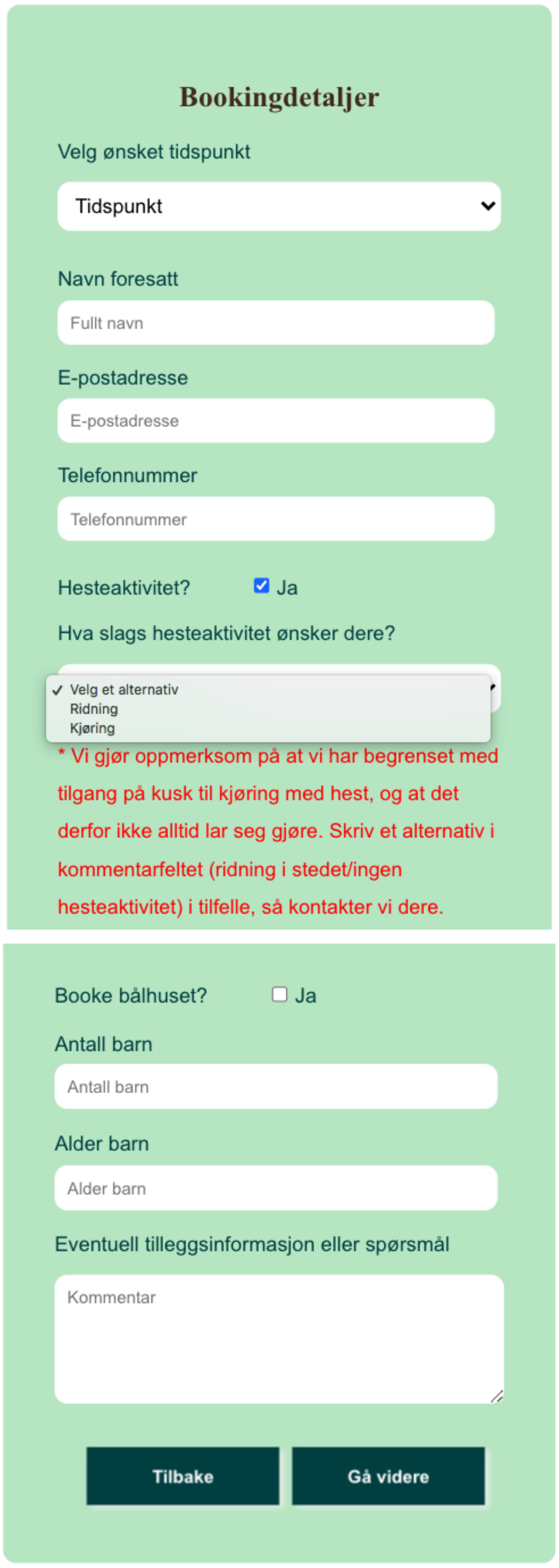
* Minsteprisen tilsvarer prisen for 10 deltakere, dvs. 1500,- uten hesteaktivitet og 2000,- med hesteaktivitet

**Tillegg**
Leie av bålhus inkludert ved og bruk av reinskinn: 300,-
Kjøring med hest og vogn eller slede: 500,-

**Velg ønsket dato**

20.05.2023

Figure 5: First page of birthday celebration bookings. Information and date picker calendar.


Bookingdetaljer

Velg ønsket tidspunkt

Tidspunkt

Navn foresatt

Fullt navn

E-postadresse

E-postadresse

Telefonnummer

Telefonnummer

Hesteaktivitet? [✓ Ja]

Hva slags hesteaktivitet ønsker dere?

✓ Velg et alternativ
Ridning
Kjøring

* Vi gjør oppmerksom på at vi har begrenset med tilgang på kusk til kjøring med hest, og at det derfor ikke alltid lar seg gjøre. Skriv et alternativ i kommentarfeltet (ridning i stedet/ingen hesteaktivitet) i tilfelle, så kontakter vi dere.

Booke bålhuset? Ja

Antall barn

Antall barn

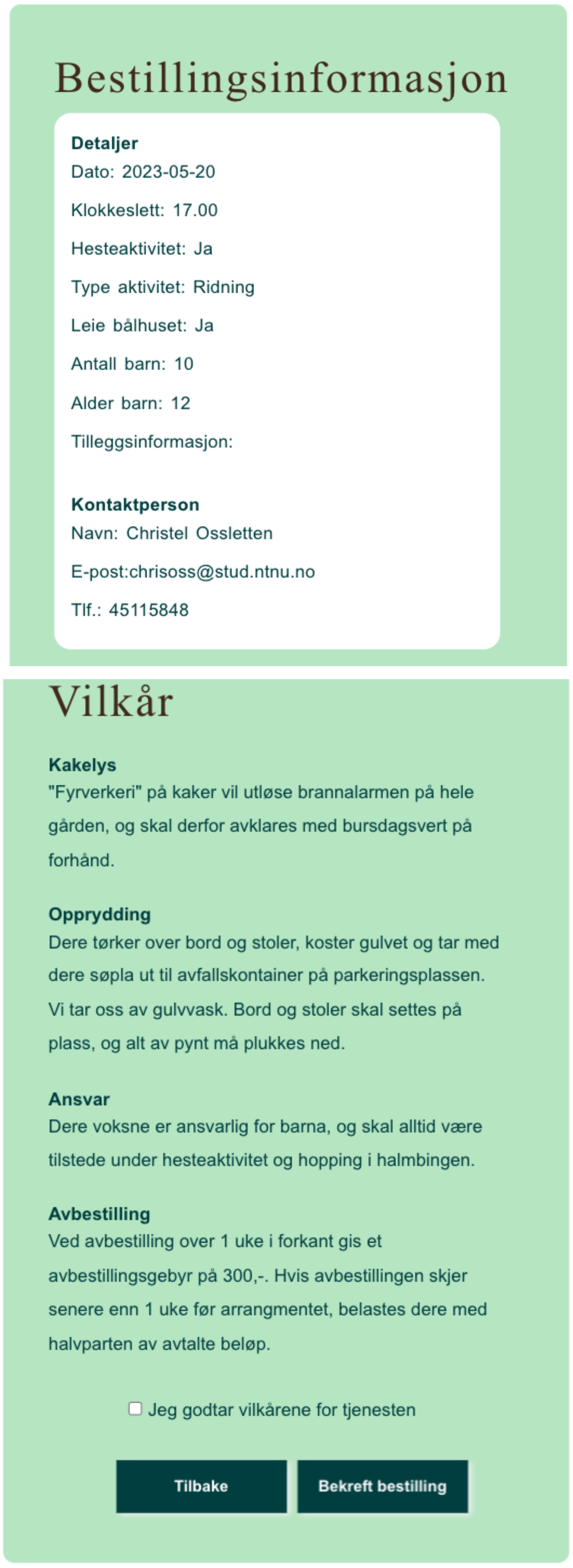
Alder barn

Alder barn

Eventuell tilleggsinformasjon eller spørsmål

Kommentar

Figure 6: Birthday booking form.



Bestillingsinformasjon

**Detaljer**

Dato: 2023-05-20

Klokkeslett: 17.00

Hesteaktivitet: Ja

Type aktivitet: Ridning

Leie bålhuset: Ja

Antall barn: 10

Alder barn: 12

Tilleggsinformasjon:

**Kontaktperson**

Navn: Christel Ossletten

E-post:chrisoss@stud.ntnu.no

Tlf.: 45115848

Vilkår

**Kakelys**

"Fyrverkeri" på kaker vil utløse brannalarmen på hele gården, og skal derfor avklares med bursdagsvert på forhånd.

**Opprydding**

Dere tørker over bord og stoler, koster gulvet og tar med dere søpla ut til avfallskontainer på parkeringsplassen. Vi tar oss av gulvvask. Bord og stoler skal settes på plass, og alt av pynt må plukkes ned.

**Ansvar**

Dere voksne er ansvarlig for barna, og skal alltid være tilstede under hesteaktivitet og hopping i halmbingen.

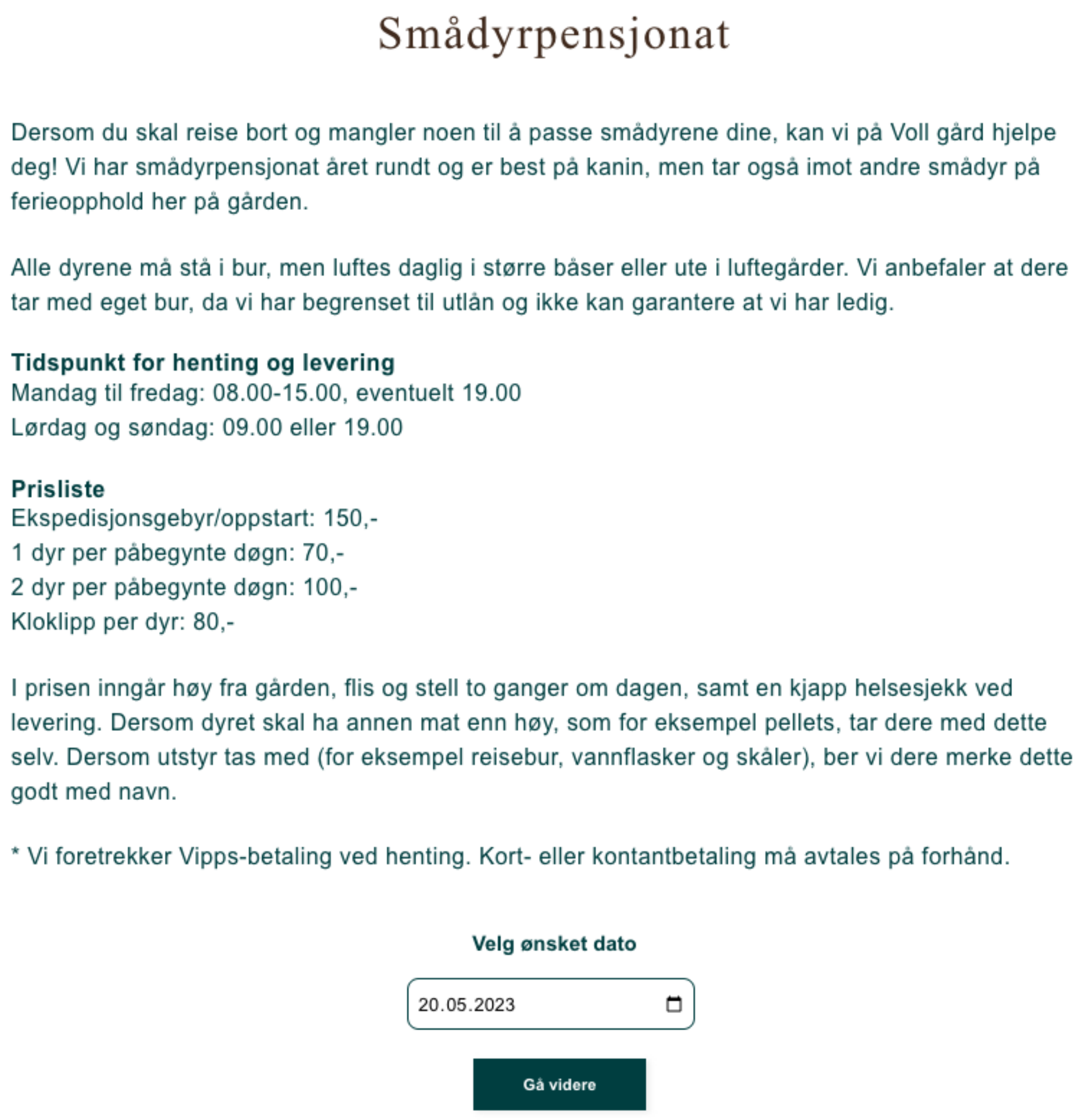
**Avbestilling**

Ved avbestilling over 1 uke i forkant gis et avbestillingsgebyr på 300,-. Hvis avbestillingen skjer senere enn 1 uke før arrangmentet, belastes dere med halvparten av avtalte beløp.

☐ Jeg godtar vilkårene for tjenesten

Figure 7: Booking details summarized together with terms and conditions.

## Animal Boarding



Smådyrpensjonat

Dersom du skal reise bort og mangler noen til å passe smådyrene dine, kan vi på Voll gård hjelpe deg! Vi har smådyrpensjonat året rundt og er best på kanin, men tar også imot andre smådyr på ferieopphold her på gården.

Alle dyrene må stå i bur, men luftes daglig i større båser eller ute i luftegårder. Vi anbefaler at dere tar med eget bur, da vi har begrenset til utlån og ikke kan garantere at vi har ledig.

**Tidspunkt for henting og levering**
Mandag til fredag: 08.00-15.00, eventuelt 19.00
Lørdag og søndag: 09.00 eller 19.00

**Prisliste**
Ekspedisjonsgebyr/oppstart: 150,-
1 dyr per påbegynte døgn: 70,-
2 dyr per påbegynte døgn: 100,-
Kloklipp per dyr: 80,-

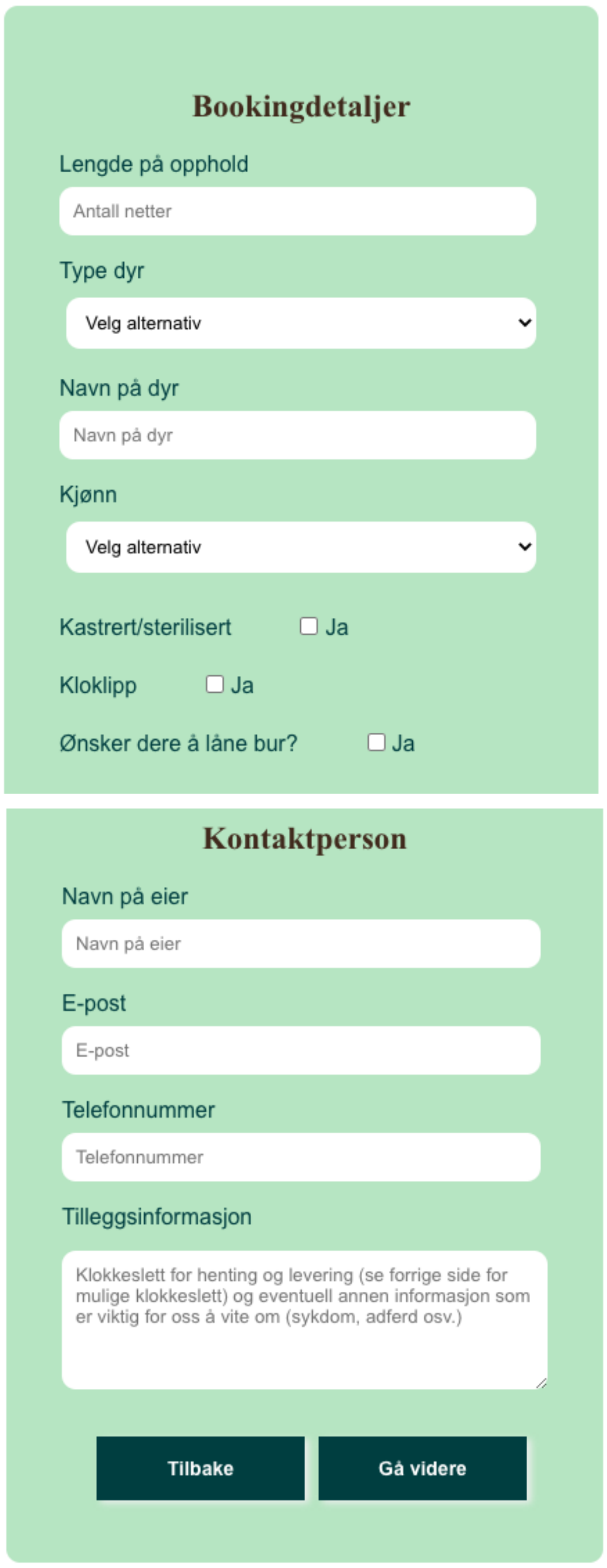
I prisen inngår høy fra gården, flis og stell to ganger om dagen, samt en kjapp helsesjekk ved levering. Dersom dyret skal ha annen mat enn høy, som for eksempel pellets, tar dere med dette selv. Dersom utstyr tas med (for eksempel reisebur, vannflasker og skåler), ber vi dere merke dette godt med navn.

* Vi foretrekker Vipps-betaling ved henting. Kort- eller kontantbetaling må avtales på forhånd.

**Velg ønsket dato**

20.05.2023

Figure 8: Animal boarding booking page. Information is presented and the animal owner must pick a suitable date to bring the animal.


Bookingdetaljer

Lengde på opphold

Antall netter

Type dyr

Velg alternativ

Navn på dyr

Navn på dyr

Kjønn

Velg alternativ

Kastrert/sterilisert ☐ Ja

Kloklipp ☐ Ja

Ønsker dere å låne bur? ☐ Ja

Kontaktperson

Navn på eier

Navn på eier

E-post

E-post

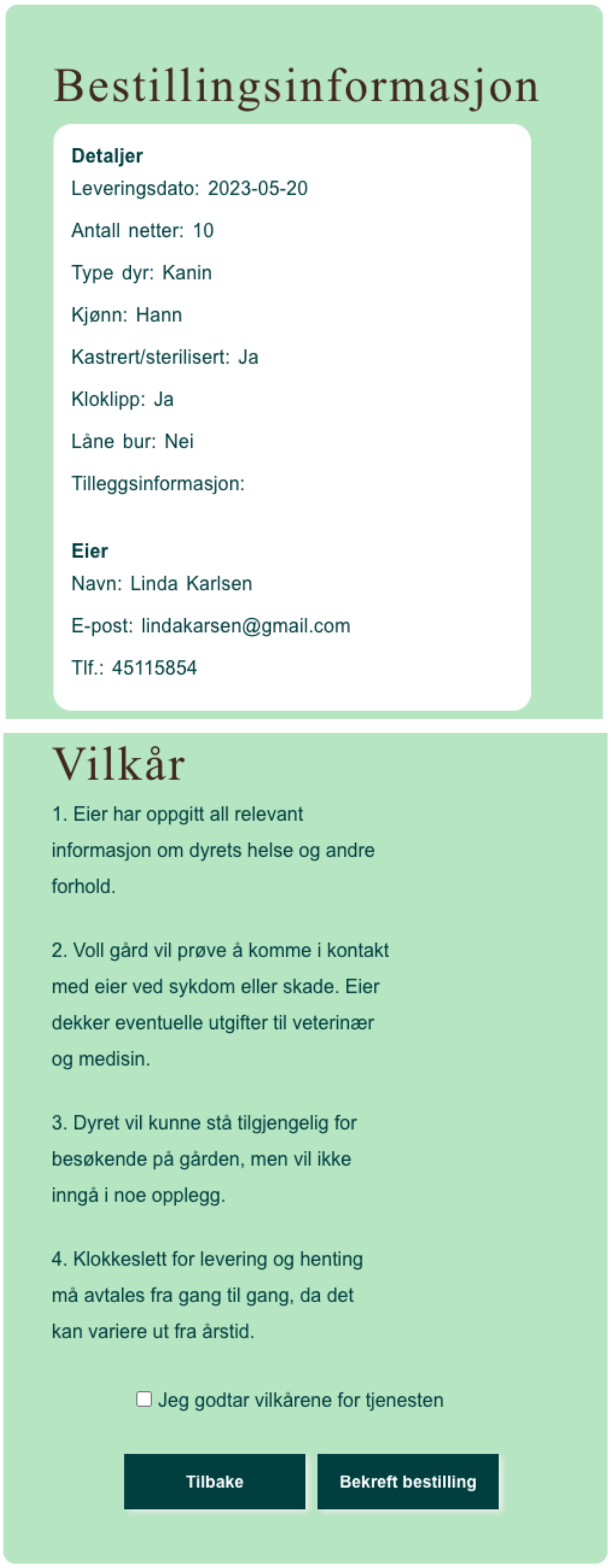
Telefonnummer

Telefonnummer

Tilleggsinformasjon

Klokkeslett for henting og levering (se forrige side for mulige klokkeslett) og eventuell annen informasjon som er viktig for oss å vite om (sykdom, adferd osv.)

Figure 9: Animal boarding booking form.


Bestillingsinformasjon

**Detaljer**

Leveringsdato: 2023-05-20

Antall netter: 10

Type dyr: Kanin

Kjønn: Hann

Kastrert/sterilisert: Ja

Kloklipp: Ja

Låne bur: Nei

Tilleggsinformasjon:

**Eier**

Navn: Linda Karlsen

E-post: lindakarsen@gmail.com

Tlf.: 45115854

Vilkår

1. Eier har oppgitt all relevant informasjon om dyrets helse og andre forhold.

2. Voll gård vil prøve å komme i kontakt med eier ved sykdom eller skade. Eier dekker eventuelle utgifter til veterinær og medisin.

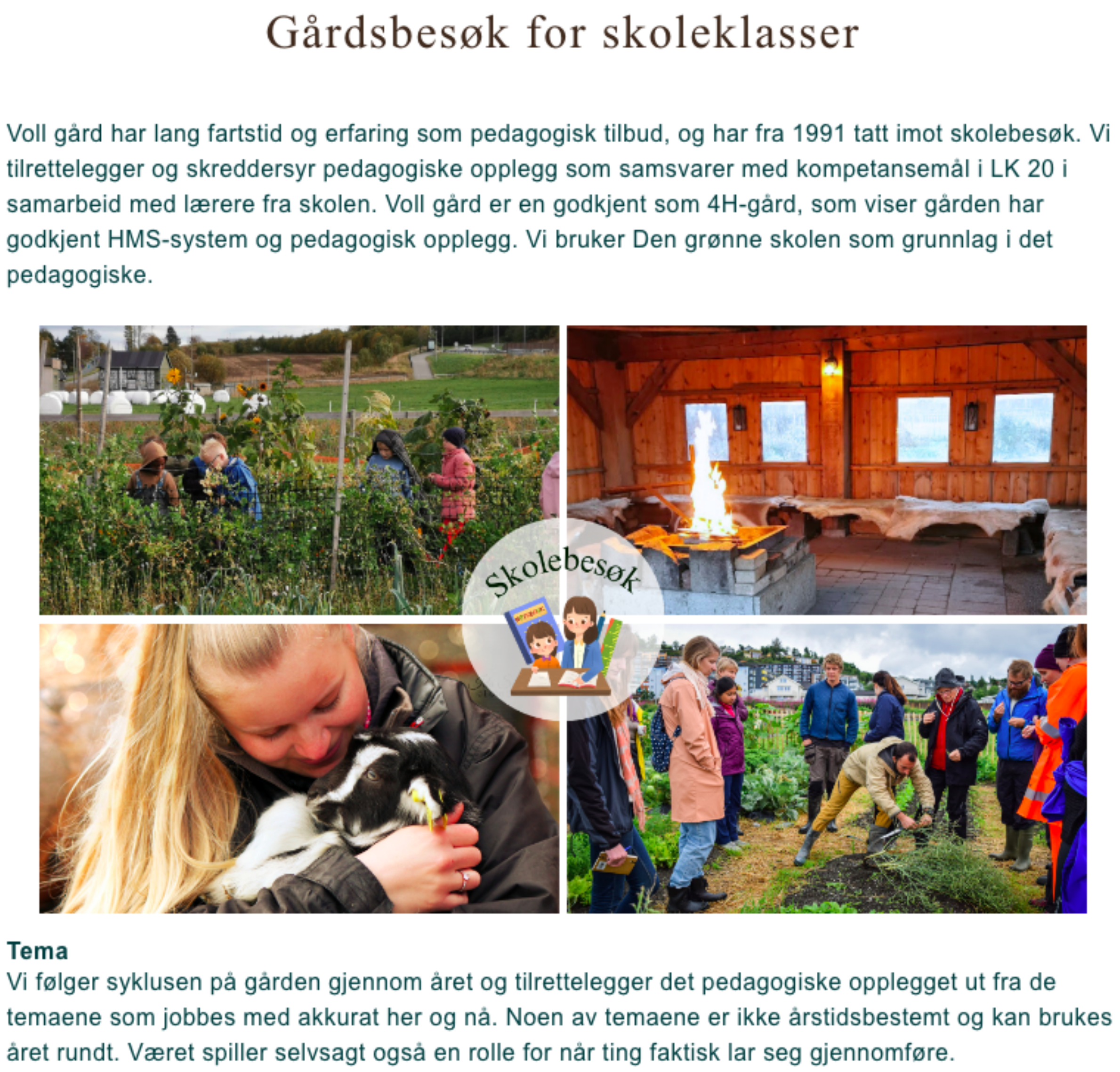
3. Dyret vil kunne stå tilgjengelig for besøkende på gården, men vil ikke inngå i noe opplegg.

4. Klokkeslett for levering og henting må avtales fra gang til gang, da det kan variere ut fra årstid.

☐ Jeg godtar vilkårene for tjenesten

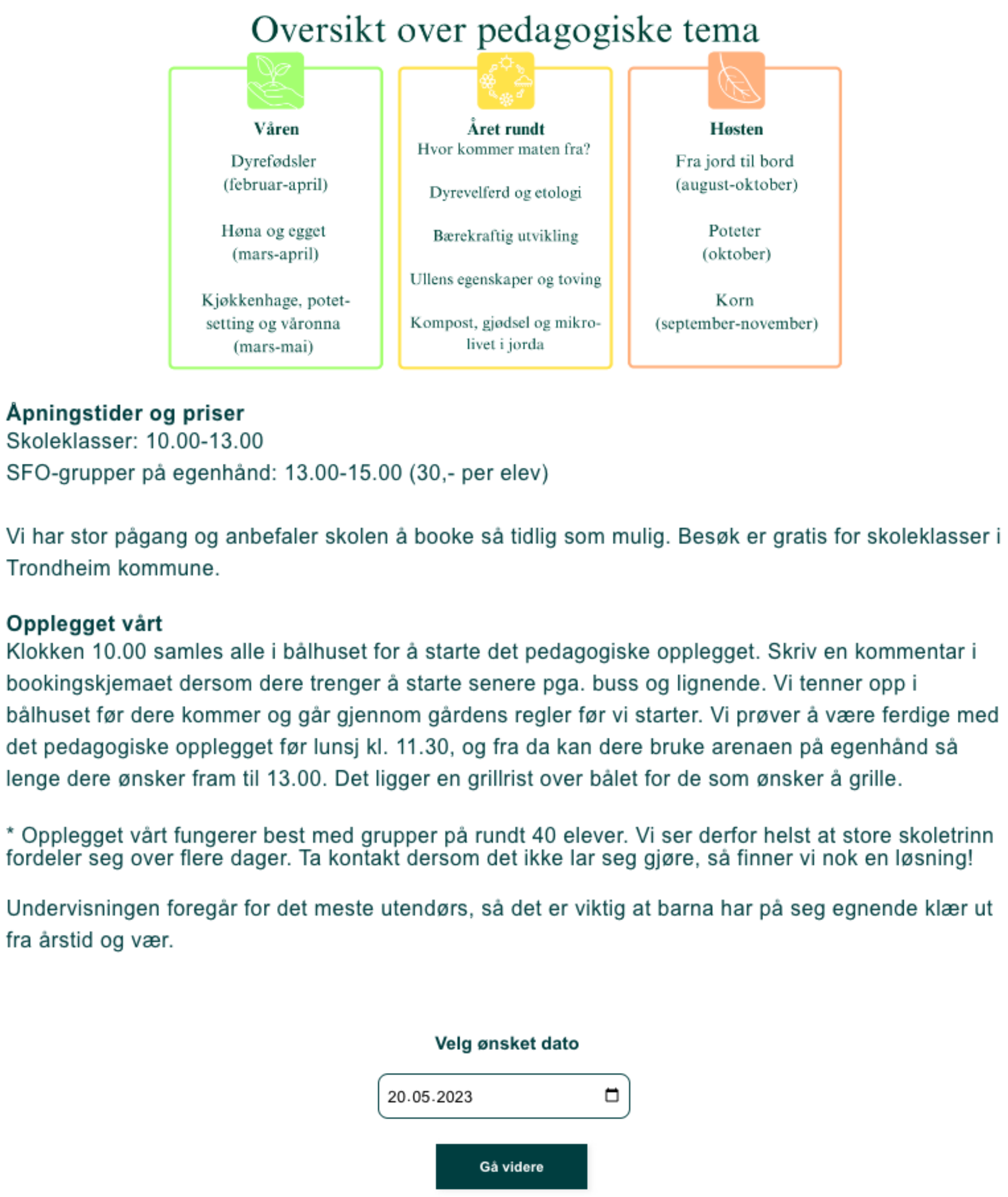
Figure 10: Booking details summarized together with terms and conditions.

## School Visits



Voll gård har lang fartstid og erfaring som pedagogisk tilbud, og har fra 1991 tatt imot skolebesøk. Vi tilrettelegger og skreddersyr pedagogiske opplegg som samsvarer med kompetansemål i LK 20 i samarbeid med lærere fra skolen. Voll gård er en godkjent som 4H-gård, som viser gården har godkjent HMS-system og pedagogisk opplegg. Vi bruker Den grønne skolen som grunnlag i det pedagogiske.

**Tema**
Vi følger syklusen på gården gjennom året og tilrettelegger det pedagogiske opplegget ut fra de temaene som jobbes med akkurat her og nå. Noen av temaene er ikke årstidsbestemt og kan brukes året rundt. Været spiller selvsagt også en rolle for når ting faktisk lar seg gjennomføre.


## Oversikt over pedagogiske tema

| Våren | Året rundt | Høsten |
|---|---|---|
| Dyrefødsler (februar-april) | Hvor kommer maten fra? | Fra jord til bord (august-oktober) |
| Høna og egget (mars-april) | Dyrevelferd og etologi | Poteter (oktober) |
| Kjøkkenhage, potet-setting og våronna (mars-mai) | Bærekraftig utvikling | Korn (september-november) |
| | Ullens egenskaper og toving | |
| | Kompost, gjødsel og mikro-livet i jorda | |

**Åpningstider og priser**
Skoleklasser: 10.00-13.00
SFO-grupper på egenhånd: 13.00-15.00 (30,- per elev)

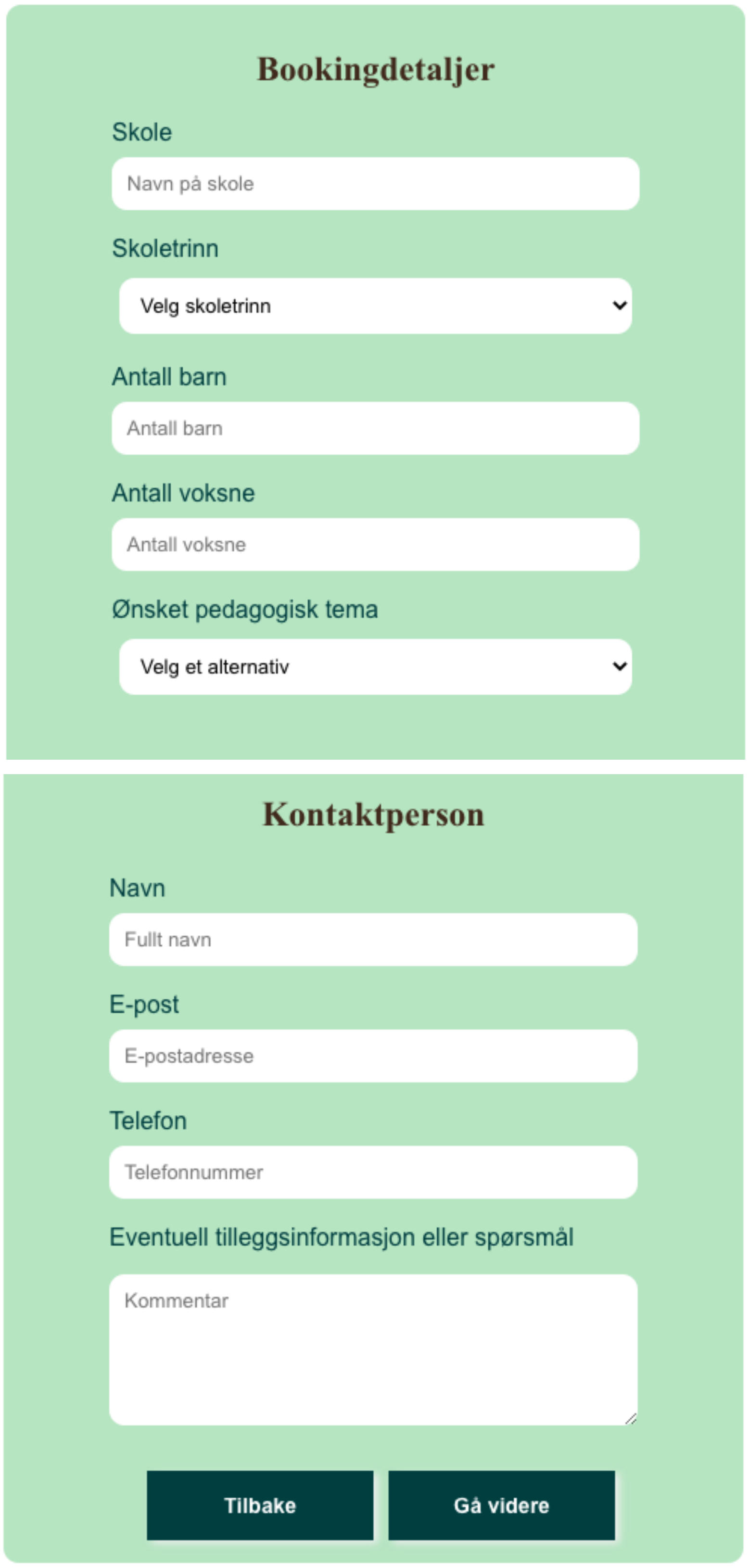
Vi har stor pågang og anbefaler skolen å booke så tidlig som mulig. Besøk er gratis for skoleklasser i Trondheim kommune.

**Opplegget vårt**
Klokken 10.00 samles alle i bålhuset for å starte det pedagogiske opplegget. Skriv en kommentar i bookingskjemaet dersom dere trenger å starte senere pga. buss og lignende. Vi tenner opp i bålhuset før dere kommer og går gjennom gårdens regler før vi starter. Vi prøver å være ferdige med det pedagogiske opplegget før lunsj kl. 11.30, og fra da kan dere bruke arenaen på egenhånd så lenge dere ønsker fram til 13.00. Det ligger en grillrist over bålet for de som ønsker å grille.

* Opplegget vårt fungerer best med grupper på rundt 40 elever. Vi ser derfor helst at store skoletrinn fordeler seg over flere dager. Ta kontakt dersom det ikke lar seg gjøre, så finner vi nok en løsning!

Undervisningen foregår for det meste utendørs, så det er viktig at barna har på seg egnende klær ut fra årstid og vær.

Figure 11: School visit booking page. Information is presented and the teachers must choose a date.



Bookingdetaljer

Skole

Navn på skole

Skoletrinn

Velg skoletrinn

Antall barn

Antall barn

Antall voksne

Antall voksne

Ønsket pedagogisk tema

Velg et alternativ

Kontaktperson

Navn

Fullt navn

E-post

E-postadresse

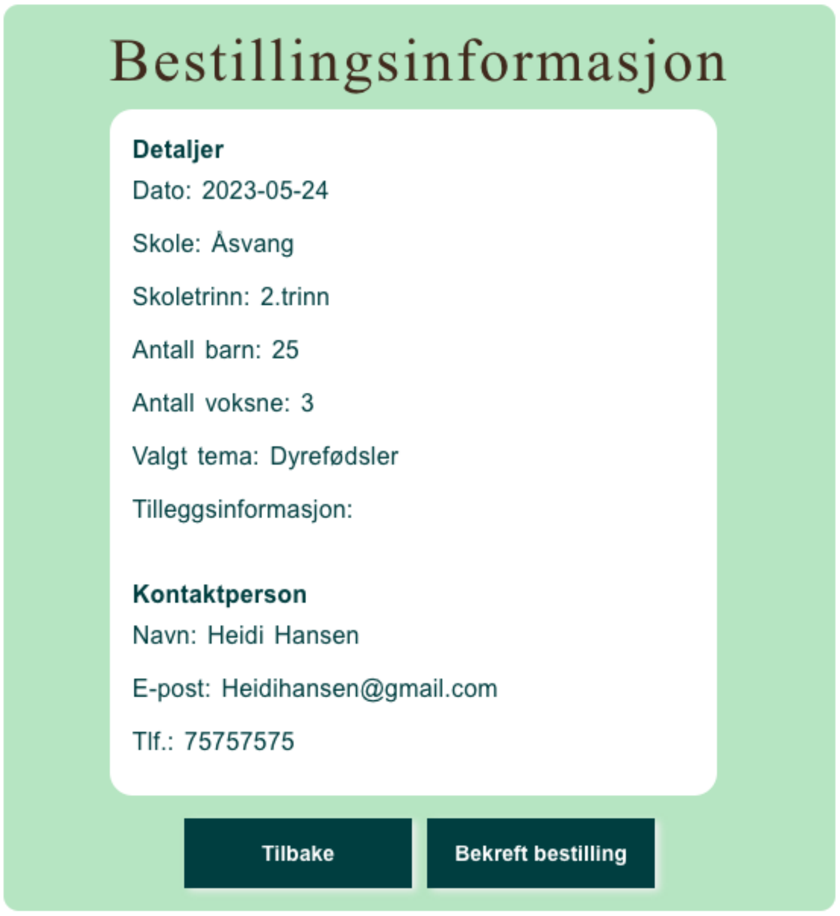
Telefon

Telefonnummer

Eventuell tilleggsinformasjon eller spørsmål

Kommentar

Figure 12: School visit booking form.


Bestillingsinformasjon

**Detaljer**
Dato: 2023-05-24
Skole: Åsvang
Skoletrinn: 2.trinn
Antall barn: 25
Antall voksne: 3
Valgt tema: Dyrefødsler
Tilleggsinformasjon:

**Kontaktperson**
Navn: Heidi Hansen
E-post: Heidihansen@gmail.com
Tlf.: 75757575

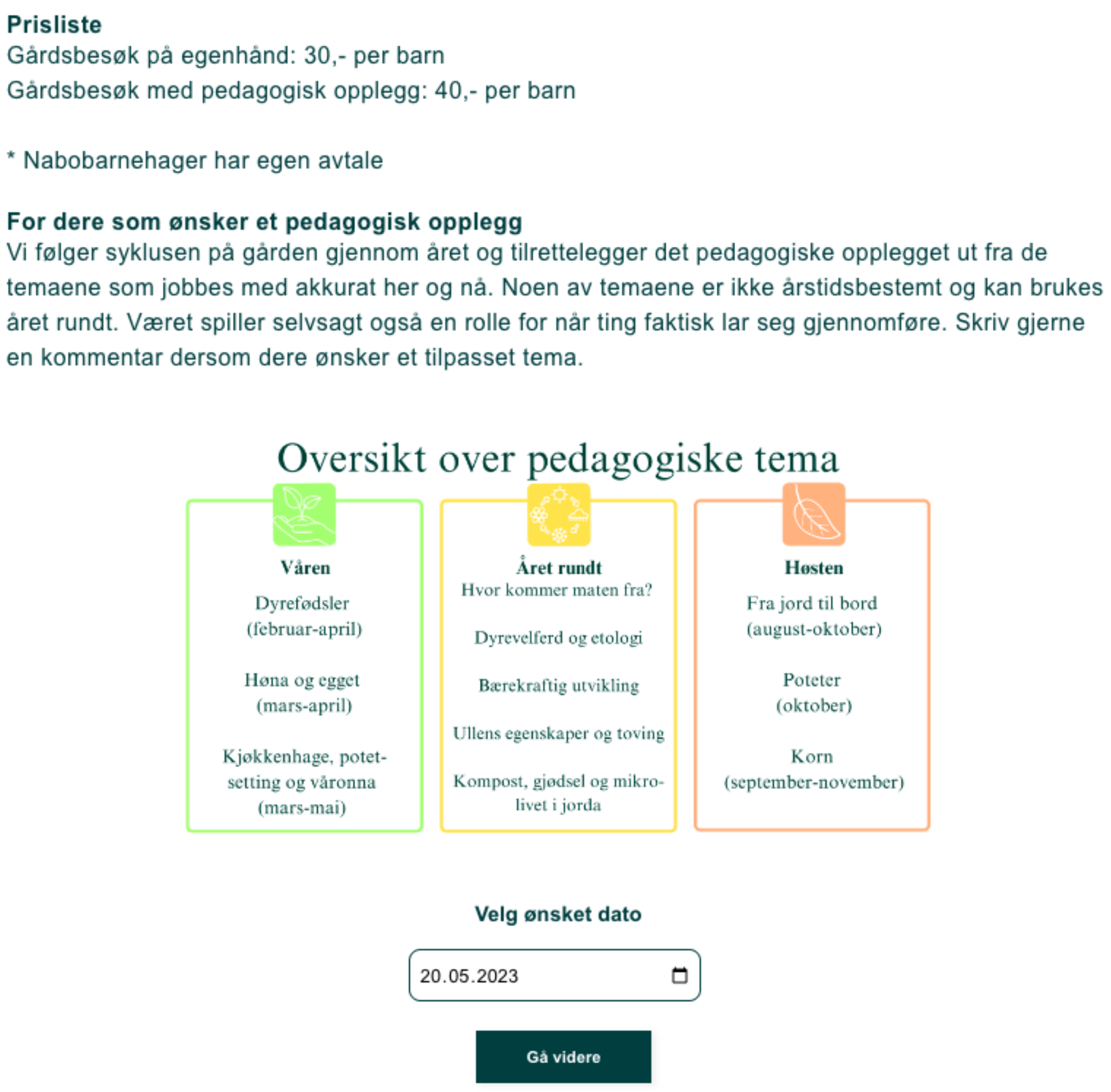
Figure 13: Booking details summarized.

## Kindergarten Visits

### Gårdsbesøk for barnehager

Vi ønsker barnehager velkommen på gårdsbesøk! Vi har åpent for besøk mellom 10.00 og 15.00 i ukedagene. Da kan dere besøke dyrene i fjøset og stallen, hoppe i halmbingen og spise lunsj på gården. Ved ledig kapasitet blir våre ansatte med dere en runde slik at dere får nærkontakt med dyrene. Bålhuset er forbeholdt skoleklasser med pedagogisk opplegg, men ved tidlig forespørsel kan det være ledig. Ellers er det mulig å spise lunsj oppe på låven, på benkene utendørs eller i fjøsstua. Skriv gjerne en kommentar på hva dere ønsker, så hører dere fra oss.

Alle besøk skal bookes på forhånd. Dette gjelder også for barnehager med naboavtale. Vi trenger å vite hvem som bruker arenaen for å kunne tilrettelegge best mulig og føre besøksstatistikk.



**Prisliste**
Gårdsbesøk på egenhånd: 30,- per barn
Gårdsbesøk med pedagogisk opplegg: 40,- per barn

* Nabobarnehager har egen avtale

**For dere som ønsker et pedagogisk opplegg**
Vi følger syklusen på gården gjennom året og tilrettelegger det pedagogiske opplegget ut fra de temaene som jobbes med akkurat her og nå. Noen av temaene er ikke årstidsbestemt og kan brukes året rundt. Været spiller selvsagt også en rolle for når ting faktisk lar seg gjennomføre. Skriv gjerne en kommentar dersom dere ønsker et tilpasset tema.

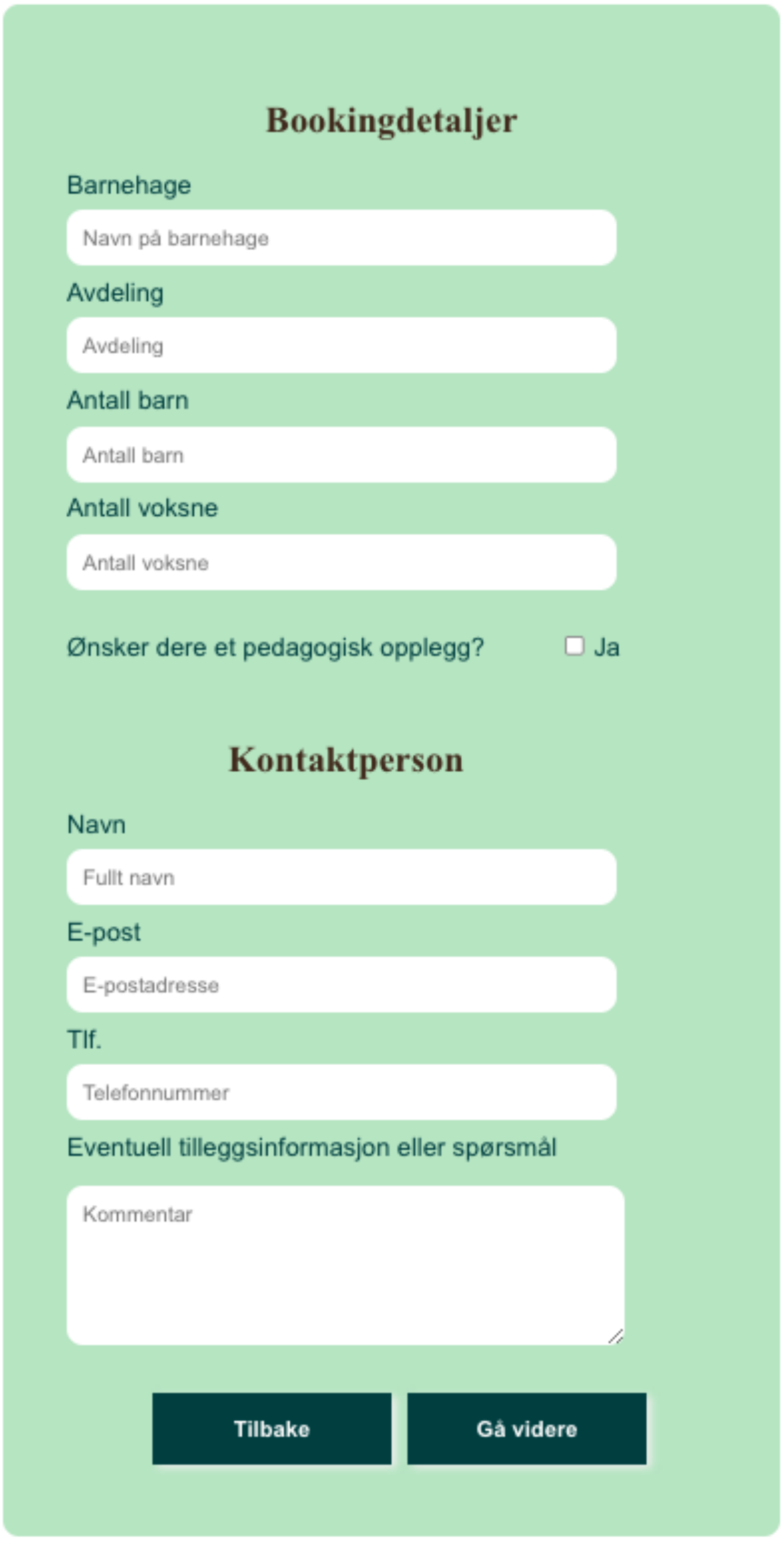
Oversikt over pedagogiske tema

| Våren | Året rundt | Høsten |
|---|---|---|
| Dyrefødsler (februar-april) | Hvor kommer maten fra? | Fra jord til bord (august-oktober) |
| Høna og egget (mars-april) | Dyrevelferd og etologi | Poteter (oktober) |
| Kjøkkenhage, potet-setting og våronna (mars-mai) | Bærekraftig utvikling | Korn (september-november) |
|  | Ullens egenskaper og toving |  |
|  | Kompost, gjødsel og mikro-livet i jorda |  |

Velg ønsket dato

20.05.2023

Figure 14: Kindergarten visit booking page. Information is presented and the teachers must choose a date.



Bookingdetaljer

Barnehage

Navn på barnehage

Avdeling

Avdeling

Antall barn

Antall barn

Antall voksne

Antall voksne

Ønsker dere et pedagogisk opplegg? ☐ Ja

Kontaktperson

Navn

Fullt navn

E-post

E-postadresse

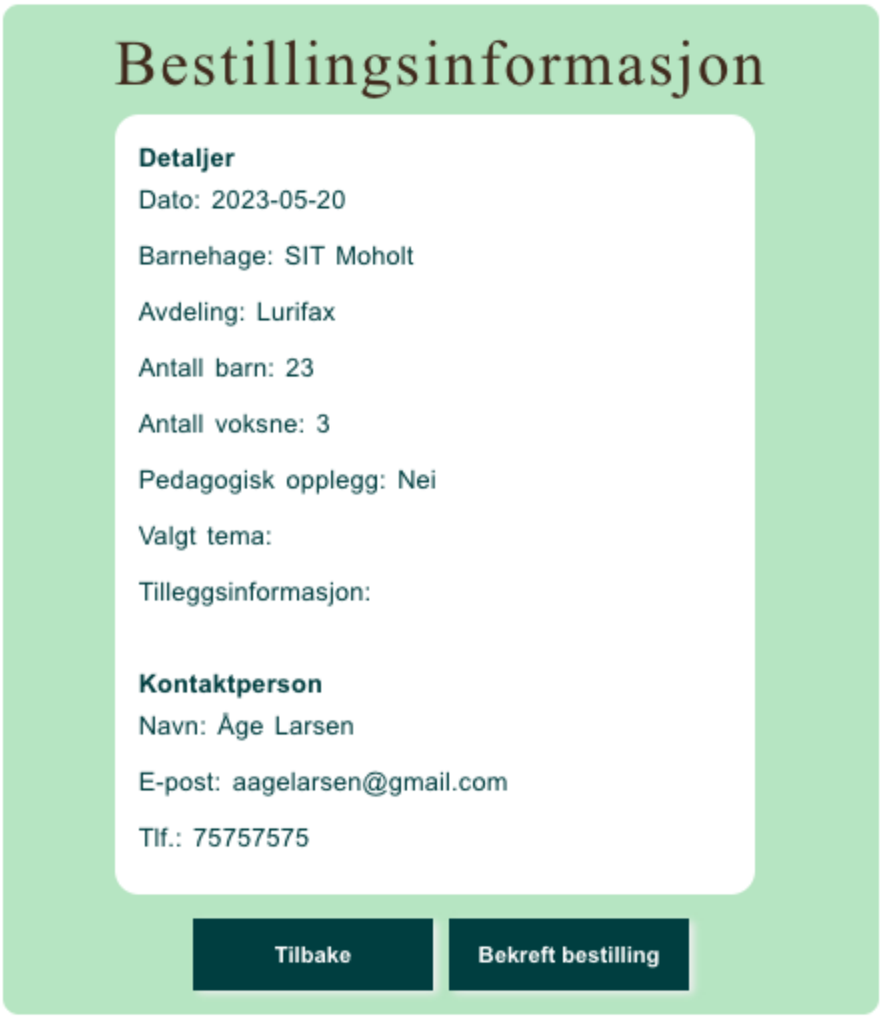
Tlf.

Telefonnummer

Eventuell tilleggsinformasjon eller spørsmål

Kommentar

Figure 15: Kindergarten visit booking form.



Bestillingsinformasjon

**Detaljer**

Dato: 2023-05-20

Barnehage: SIT Moholt

Avdeling: Lurifax

Antall barn: 23

Antall voksne: 3

Pedagogisk opplegg: Nei

Valgt tema:

Tilleggsinformasjon:

**Kontaktperson**

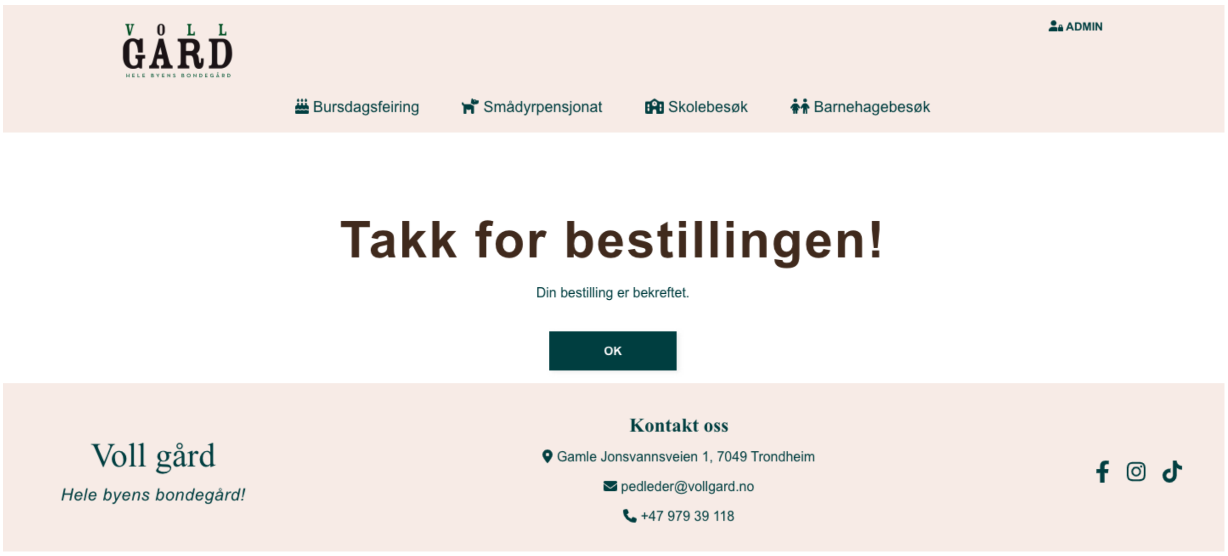
Navn: Åge Larsen

E-post: aagelarsen@gmail.com

Tlf.: 75757575

Figure 16: Booking details summarized.

## Confirmation View



Takk for bestillingen!

Din bestilling er bekreftet.

Voll gård

*Hele byens bondegård!*

Kontakt oss

Gamle Jonsvannsveien 1, 7049 Trondheim

pedleder@vollgard.no

+47 979 39 118

Figure 17: Confirmation message received when submitting bookings.