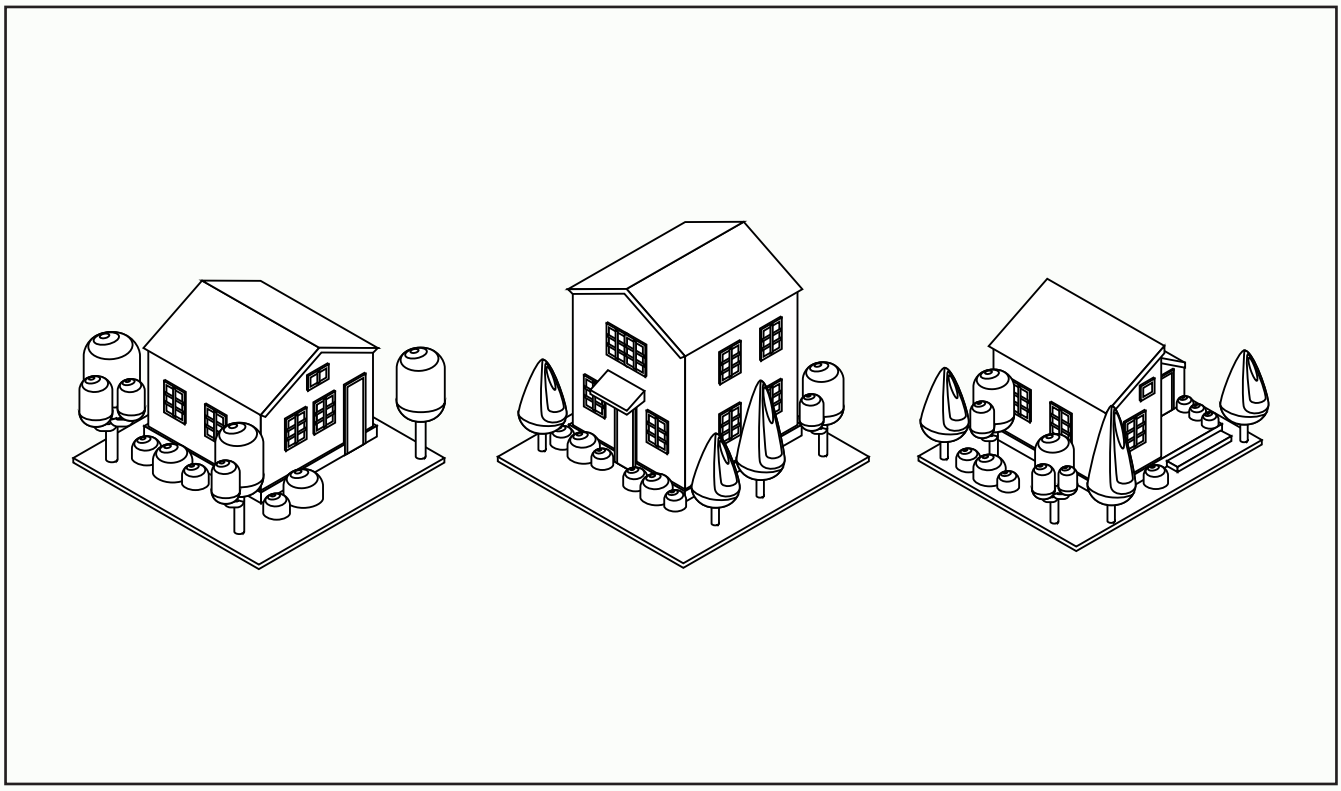
NOVI UNIVERSITY
OF APPLIED SCIENCES

FULL STACK DEVELOPMENT

Final project
vacationhome application

Lara Nettesheim

Preface.



GitHub Project:

https://github.com/lara-ai-dev/vacationhome.git

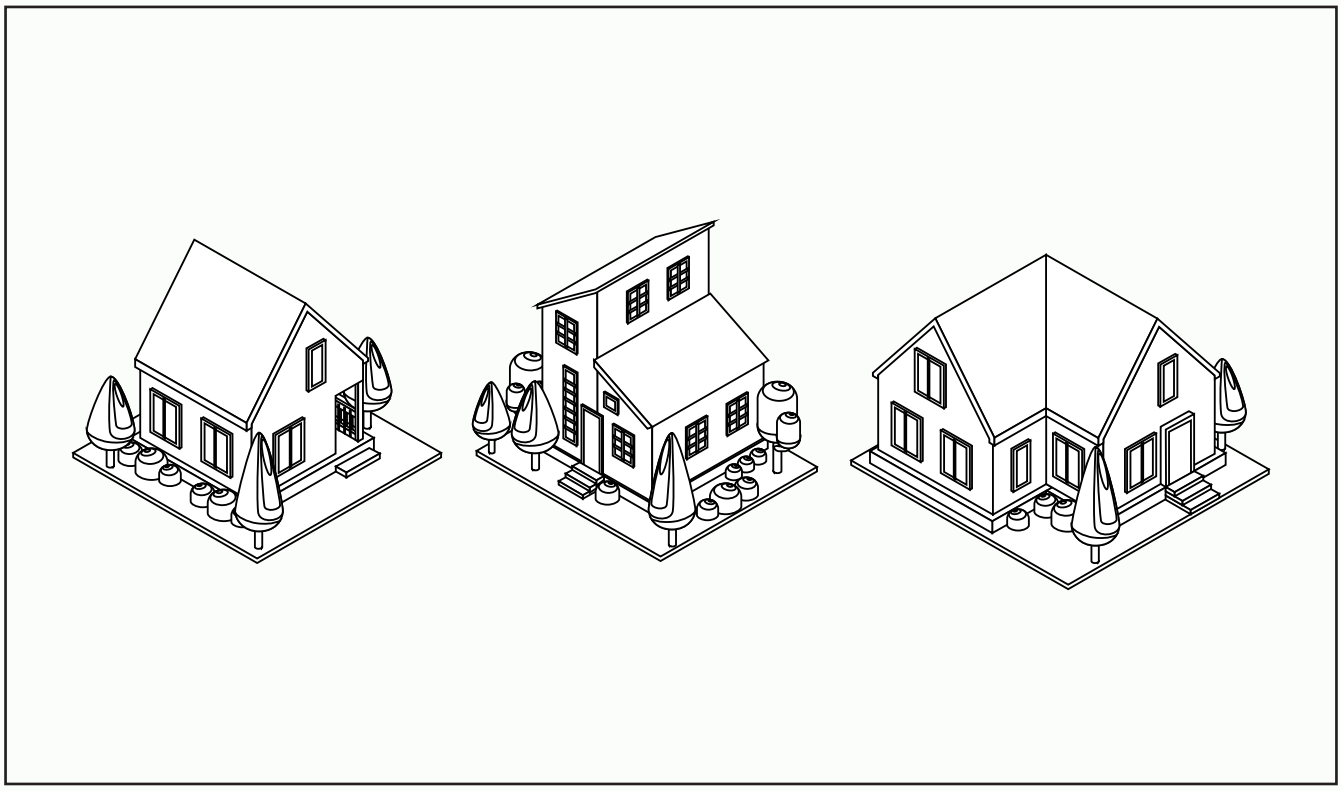
As a part of the Full-Stack-Development program by the university of applied sciences NOVI we were required to compose a report about our final project.

The personal aim of the full-stack development course was to understand and implement the newly required skills of different programming languages into a “real-lifetime” project. This real-lifetime project I chose was an apartment booking (reservation) system.

Furthermore, my main ambition, to join the Full-Stack-Development bootcamp, was to acquire a new skill that I can use in my future career path.
In the future I most likely want to work in the UX research field whereby coding would be a usefull skill to have.

Table of content.

Introduction.



For the final project there are a couple of milestones which should be accomplished by the student. The different categories/ chapter are:

> Preliminary research and setting up requirements (secure software):
The preliminary research includes the analysis about the different conditions that are needed for the application. This analysis contains the interview with the client where the needs and various requirements are discussed in order to understand the different necessities. In this chapter the risk and security analysis is also conducted to avoid potential short comings in the application.

> Design and Prototype:
In the first chapter, the various wishes and needs of the client are realized. In this stage those needs are applied to a functional design and afterwards to a technical design. The technical design includes the use-case diagrams, use diagrams and class diagrams of the application.
This chapter also contains the research on various UX/UI elements. Hereby, user tests were conducted to understand if the webpage is easy-to-use and satisfactory for the end user. Subsequently, a prototype was built to aid as a blueprint for the front-end.

> front-end & back-end
In this chapter the utilized methods for the front-end and back-end are specified. Furthermore, it includes the accountability document and a general instruction on how to download the elements for the application.

Chapter I

Functional Design

Preliminary research

Research
A preliminary analysis was carried out before specific specifications/requirements could be set up. An interview and additional research on similar applications were performed for this preliminary research. By doing this, it is possible to filter out different criteria for the application that overlap with the requirements set by NOVI.

Definition study:
An interview was held with the client in order to set up the criteria of the final project. Due to the corona crisis it was not possible to search for multiple clients or users to interview and test the application. During the interview specific questions were asked regarding the client's wishes on functionality and the outline of the application.

Client :
A relative who owns a vacation home.

Interview:
The client specifically indicated in the interview that he would like to rent his holiday home to prospective visitors.
Hereby, one of the client's wishes is to be able view its apartments. In addition, when applicable, he wants to be able to see the reservations and be able to edit his reservation.
He pointed out that he wants a home feature for the outline of the website where you can see an overview of the apartments and a header with all the menu functions to move between pages.
Some of the questions asked were:
> What are functionalities that can not be missed out in the application ?
> Do you have any clear concept outlines that you want to bring into the website?
> Are there any specific elements / functionalities that you do not want?

Final project idea (in 250 words)

The final project idea is about creating an apartment booking system. The client should be able to login and see an overview of all the reservations made.
Furthermore, the client should be also able to cancel their reservation if needed.

The user is able to reserve an apartment if the apartment is available on the selected dates. After booking, the user should get an email confirmation and be able to leave a review. Moreover, they should be able to upload pictures with the review as well. The user should be able to see an overview of the apartments and the specific location.

Chapter I

Functional Design

Wireframes

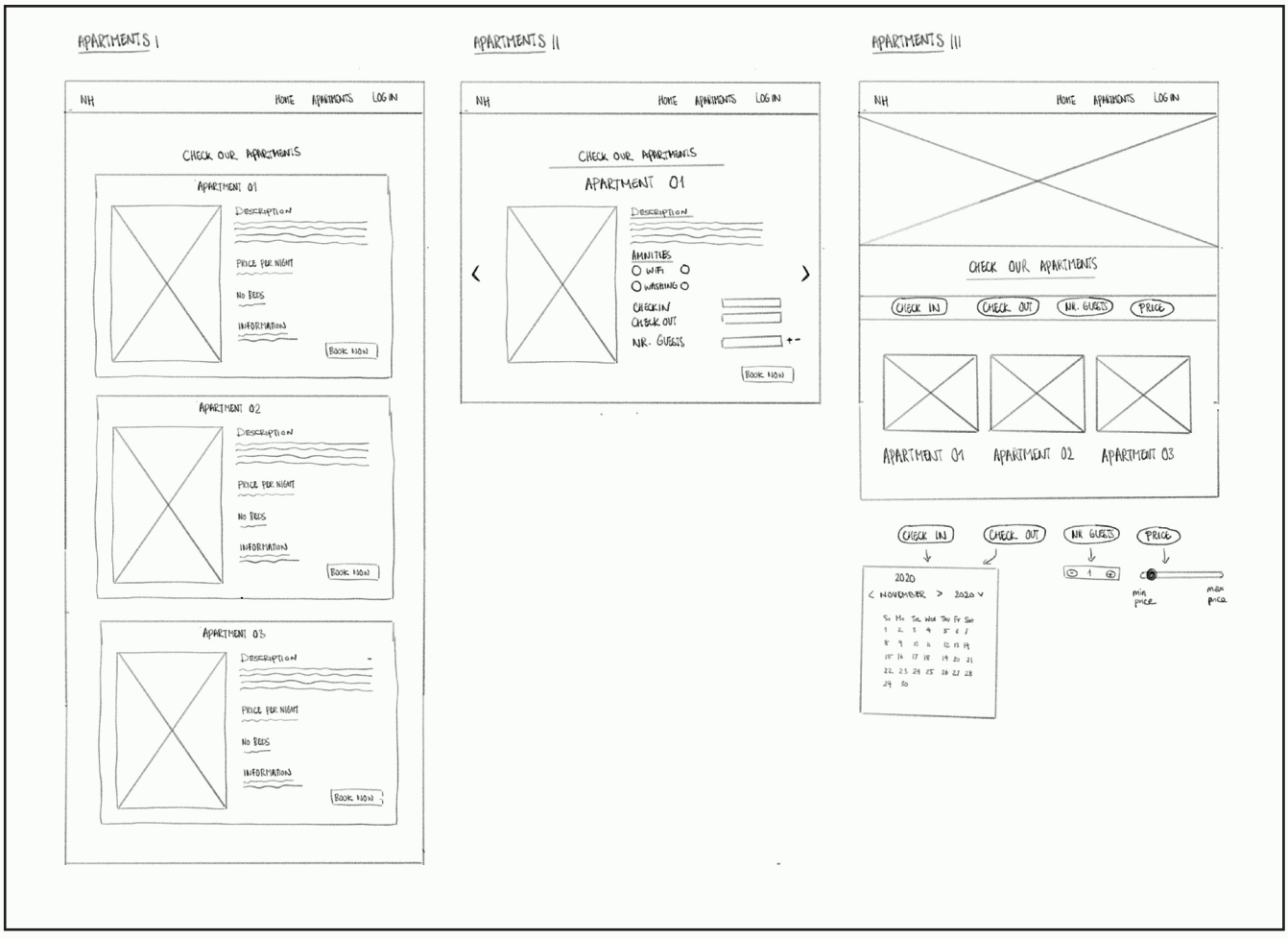
Sketching wireframes

The wireframing approach was used in order to explain the layout of the overall content and page features. Before the wireframe's final design was decided, several iterations were made.

On four key screens, I made iterations :

> Home screen
The user can see the overall elements of the application on the home screen. This home page includes an overview of the apartments, location and a review section.

> *Home screen - User Testing*
For the Apartment section the user should be able to see a list of apartments. This section should either aid as a filter where the user is able to filter their desired apartment out or as a information page for all the apartments.



Chapter I

Functional Design

Wireframes

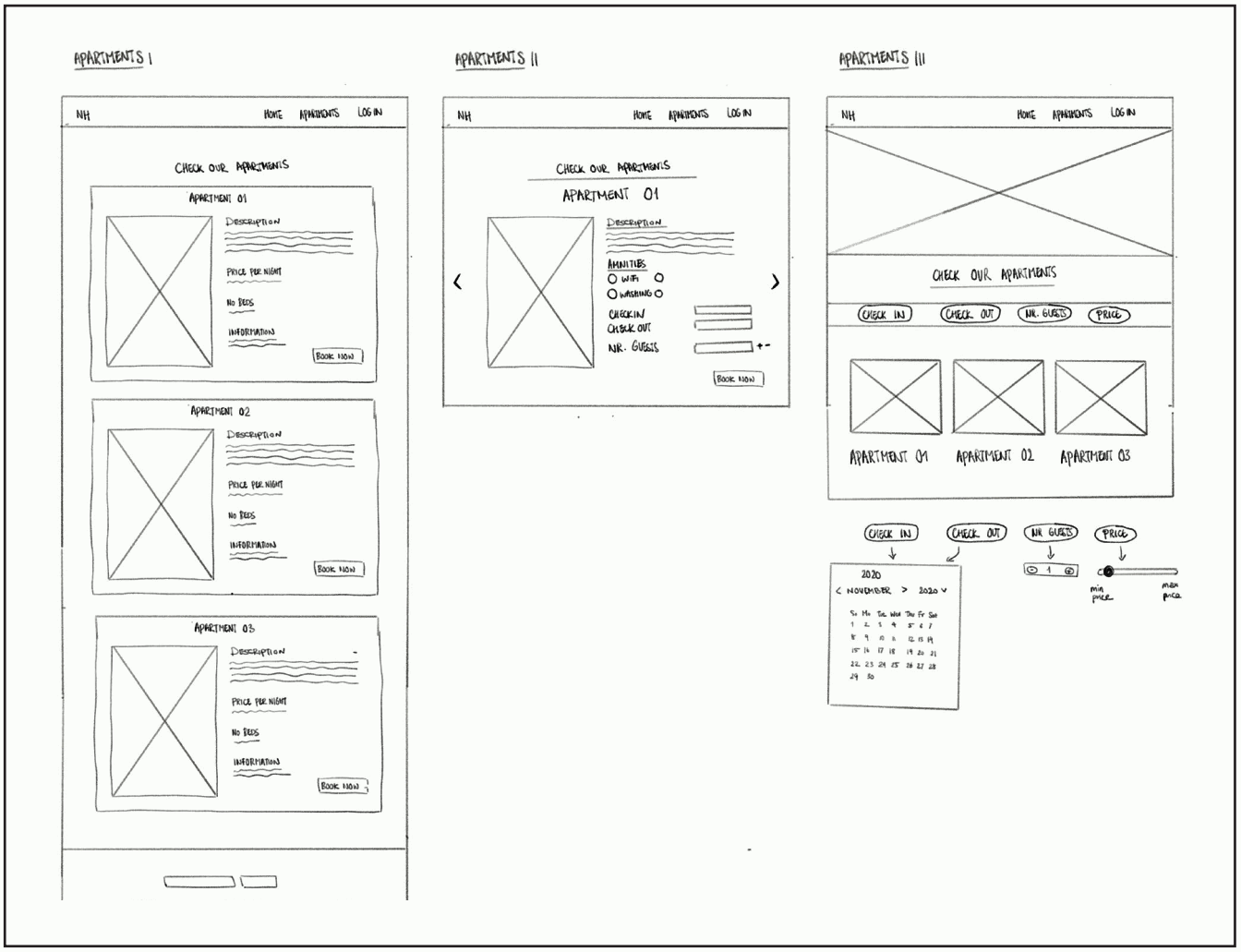
Sketching wireframes

> Apartment screen

For the Apartment section the user should be able to see a list of apartments. This section should either aid as a filter where the user is able to filter their desired apartment out or as an information page for all the apartments.

> Apartment screen - User Testing

For the Apartment section the user should be able to see a list of apartments. This section should either aid as a filter where the user is able to filter their desired apartment out or as a information page for all the apartments.



Chapter I

Functional Design

Wireframes

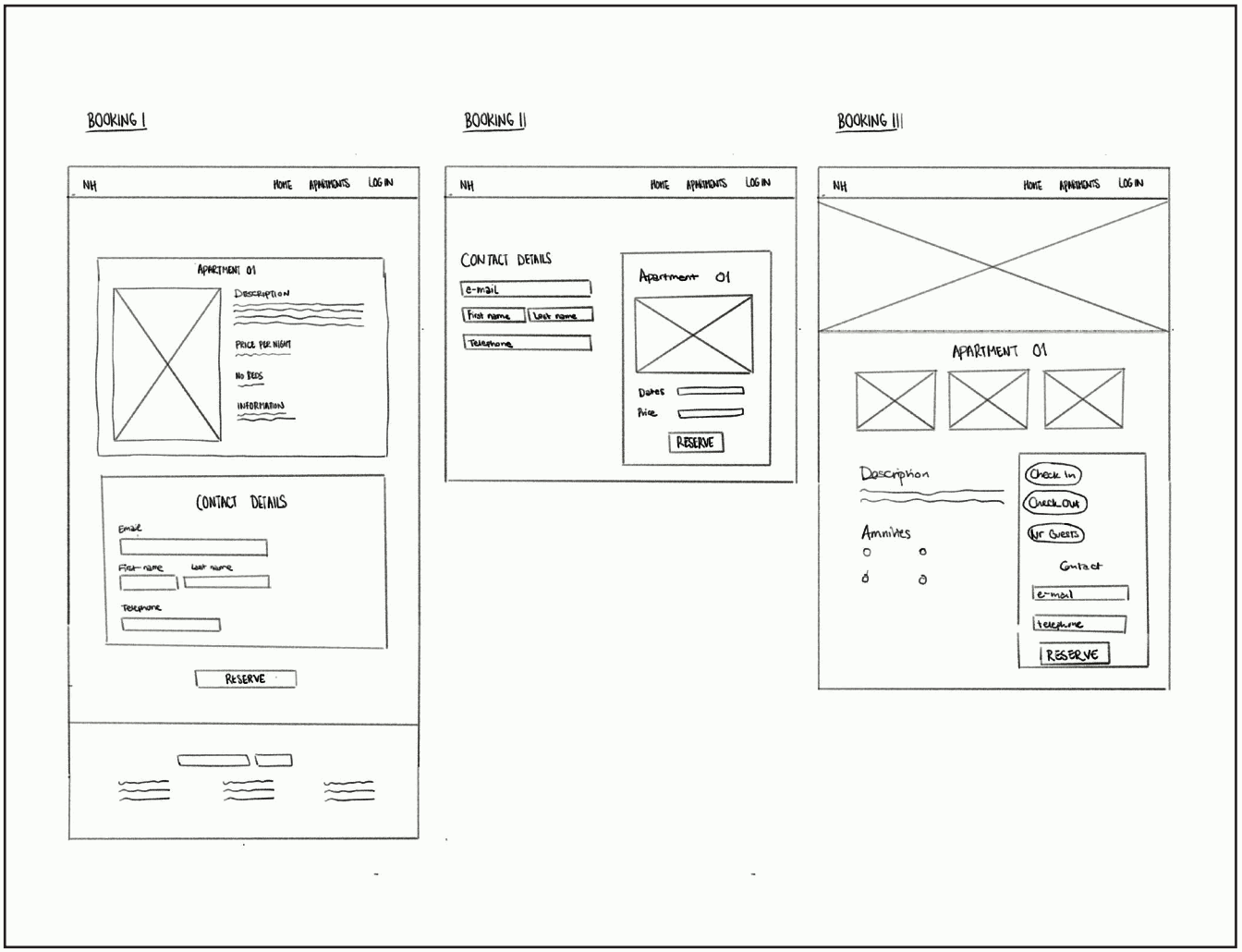
Sketching wireframes

> Booking screen

The Booking page should have the option to fill in the contact details of the user. Furthermore, it is important that the user can see what date and apartment he/she is booking for,

> Booking screen - User Testing

The user should be able to change their personal details even though they are logged in. Hence, after logging in, if the user wants to book a room the user should fill in their email adress again.



Chapter I

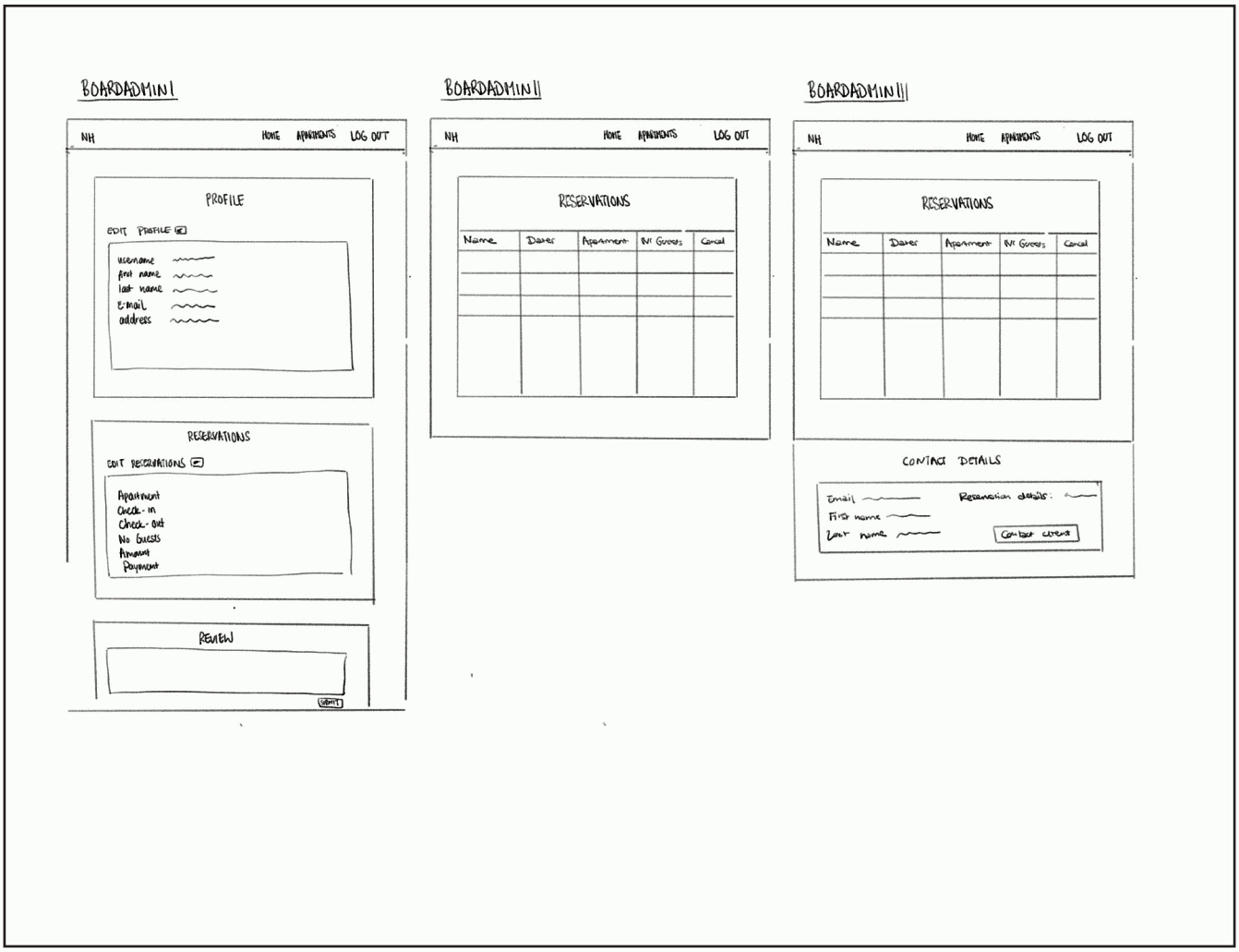
Functional Design

Wireframes

Sketching wireframes

> Board Admin screen
The admin board should only be visible for the admin. On this page the admin can see all the reservations and have also the ability to cancel or contact the clients.

> Booking screen - User Testing
A simple table is convenient for the admin to easily cancel and update reservations.



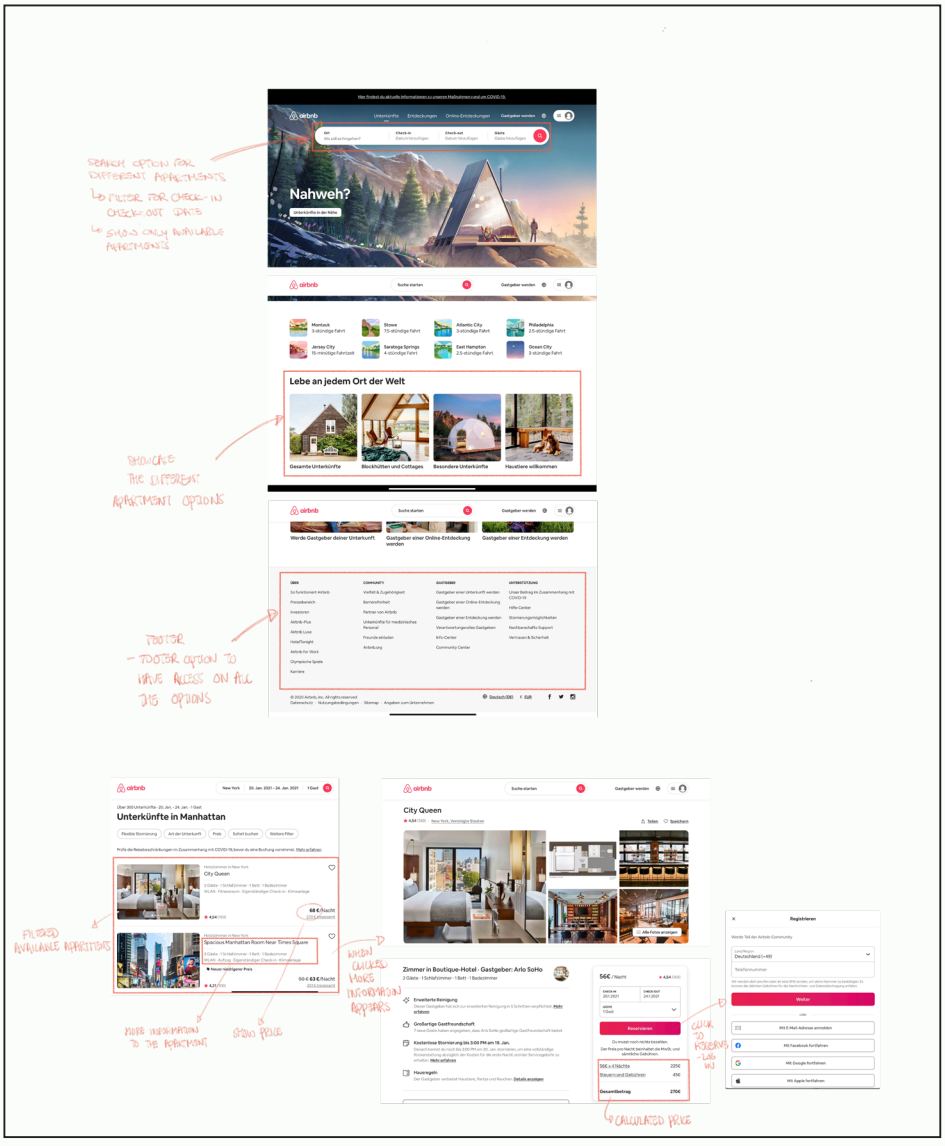
Chapter I

Functional Design

Inspiration

Getting inspirated

Throughout the wireframing process several websites were examined visually. These websites are mainly booking websites for vacationhomes or flights. Hereby, it was possible to understand the basic procedure the user needs to follow through in order to make a reservation.



Airbnb Inspiration
> The date filter option: the user can filter out the available apartments by selecting their checkin and checkout date
> Showcase apartments: the user can see an overview of all the apartments in the beginning
> Apartment list: the user can see an overview of the apartments with its specific descriptions
> Reservation : the user can see the price before reserving the apartment

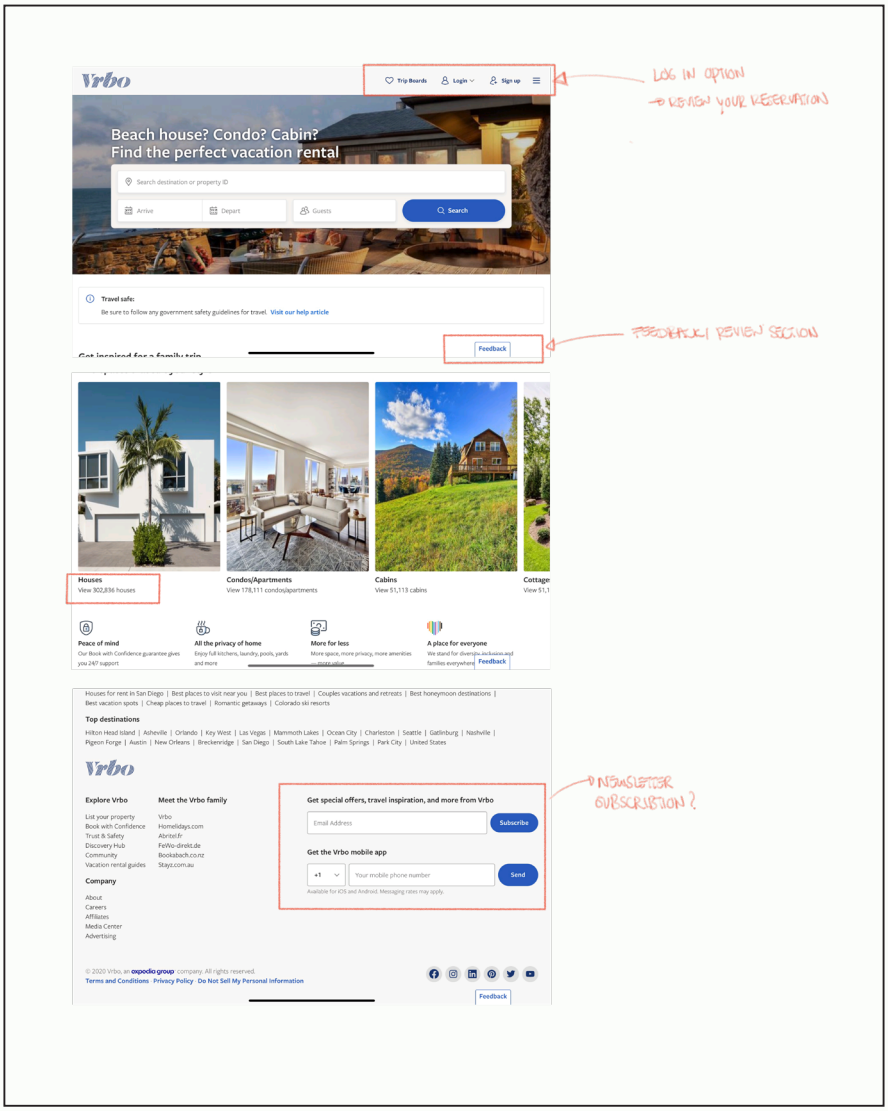
Inspiration by Airbnb

Chapter I

Functional Design

Inspiration

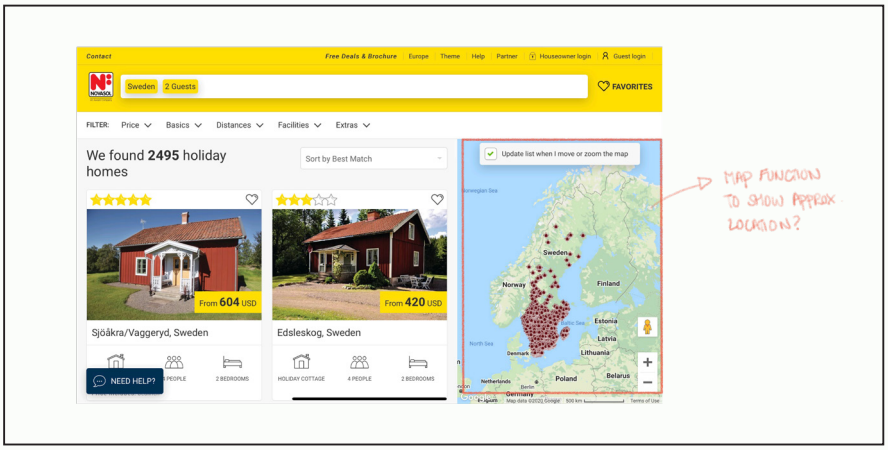
Getting inspirated



Vrbo inspiration
> Header: The header has a Login and Sign up function
> Feedback : The registered user can leave a feedback

Inspiration by Vrbo

Novasol inspiration
> Location map: the user can see the approximate location of the apartment



Inspiration by Novasol

Chapter I

Prototype

Functional Design

Prototyping in Adobe XD

In order to understand and detect which functions should be included in the application, it was necessary to prototype an application in Adobe XD. While prototyping multiple iterations were made and adjusted.

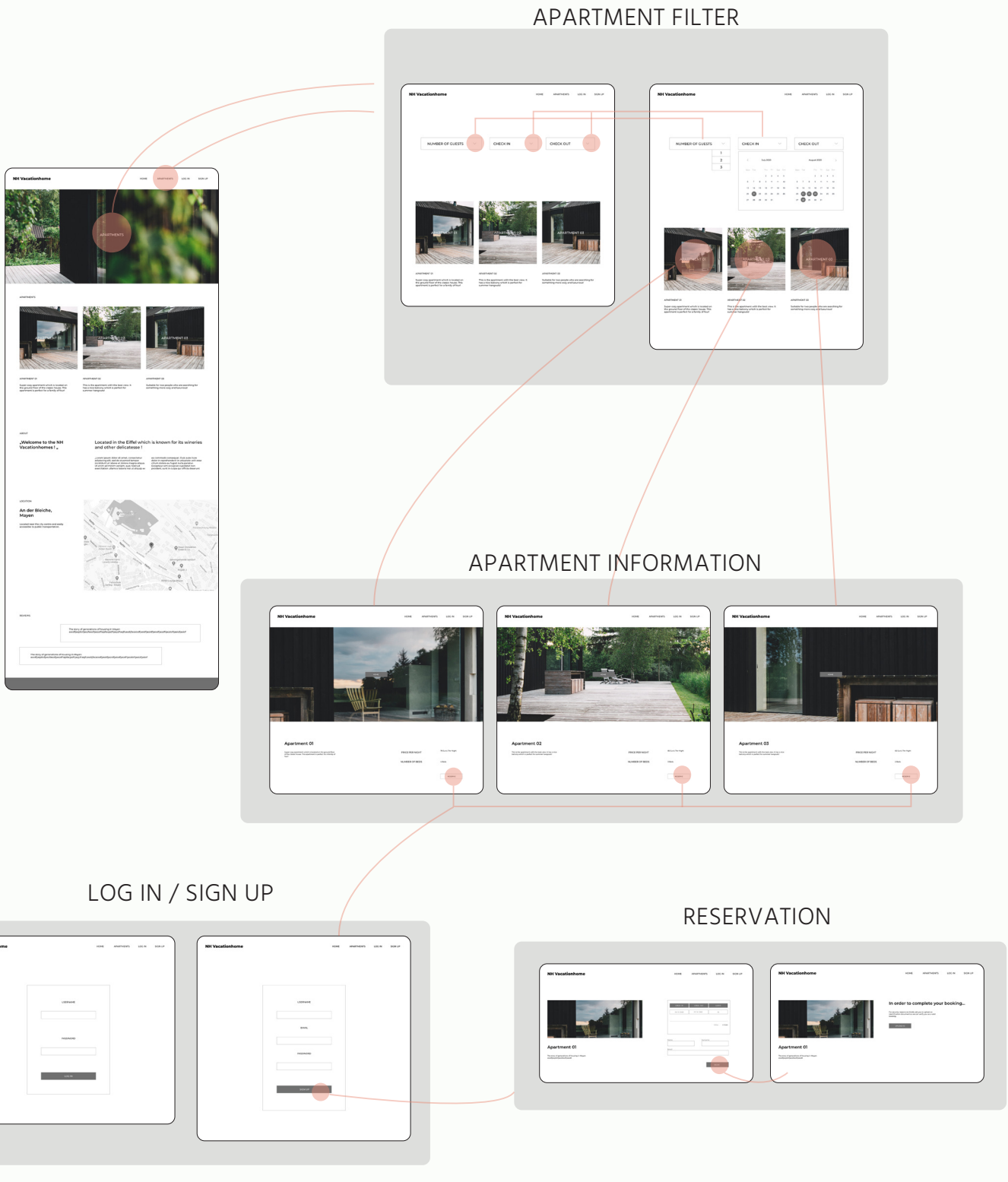
To understand the relations between the screens, colored relation lines were used to make it clearer to the reader. Furthemore, at this stage it was important for me to keep it simple to make it not too difficult in the later phase of coding.

HOME SCREEN

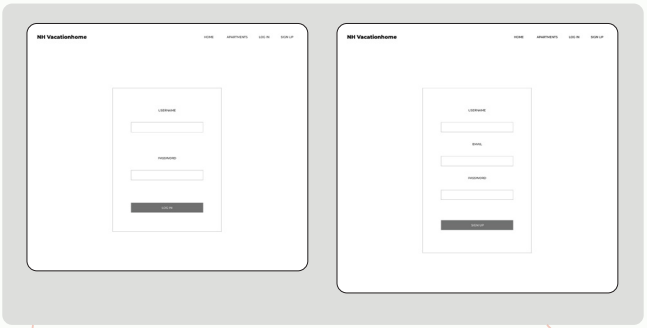
Chapter I

Prototype

Prototyping in Adobe XD

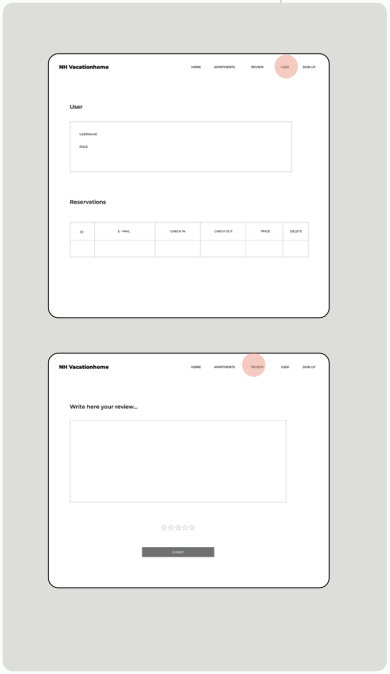


LOG IN / SIGN UP



User

Admin



Chapter I

Functional Design

General requirements

General requirements

With the interview of the client, research on similar websites and the given prerequisites by Novi, the following general requirements could be concluded:

> The application should enable interaction between users:
This could be achieved by adding a review section to engage different users to leave a feedback on their stay. In addition, another feature may be that instructions on how to get into the house can be downloaded by the user.

> The application should have multiple parts with data storage
This could be achieved by enabling reservations and reviews to be saved into a database.

> The application should be able to add and modify data
The user should be able to add a reservation and the admin should be able to modify the reservation if needed.

> The application should be able to upload data files e.g. mp3 or jpeg
The user should be able to upload a review with a picture.

Chapter I

Non / Functional Requirements

Functional Design

Functional requirements

> Sign up Feedback
Before the user is able to sign up, the email, username and password should give a feedback to the user if the email / username / pasword meets certain requirements.

> After Sign Up
The user gets a confirmation message if the sign up was successful

> Reservation Add
A new Reservation can be added after the user is logged in.

> Upload feedback
This could be achieved by enabling reservations and reviews to be saved into a database.

> Usage of WebTokens
In the application the JSON Webtokens are used for authentication and security.

> NavBar
Through the Navigation Bar the user can navigate to different pages.

Non functional requirements

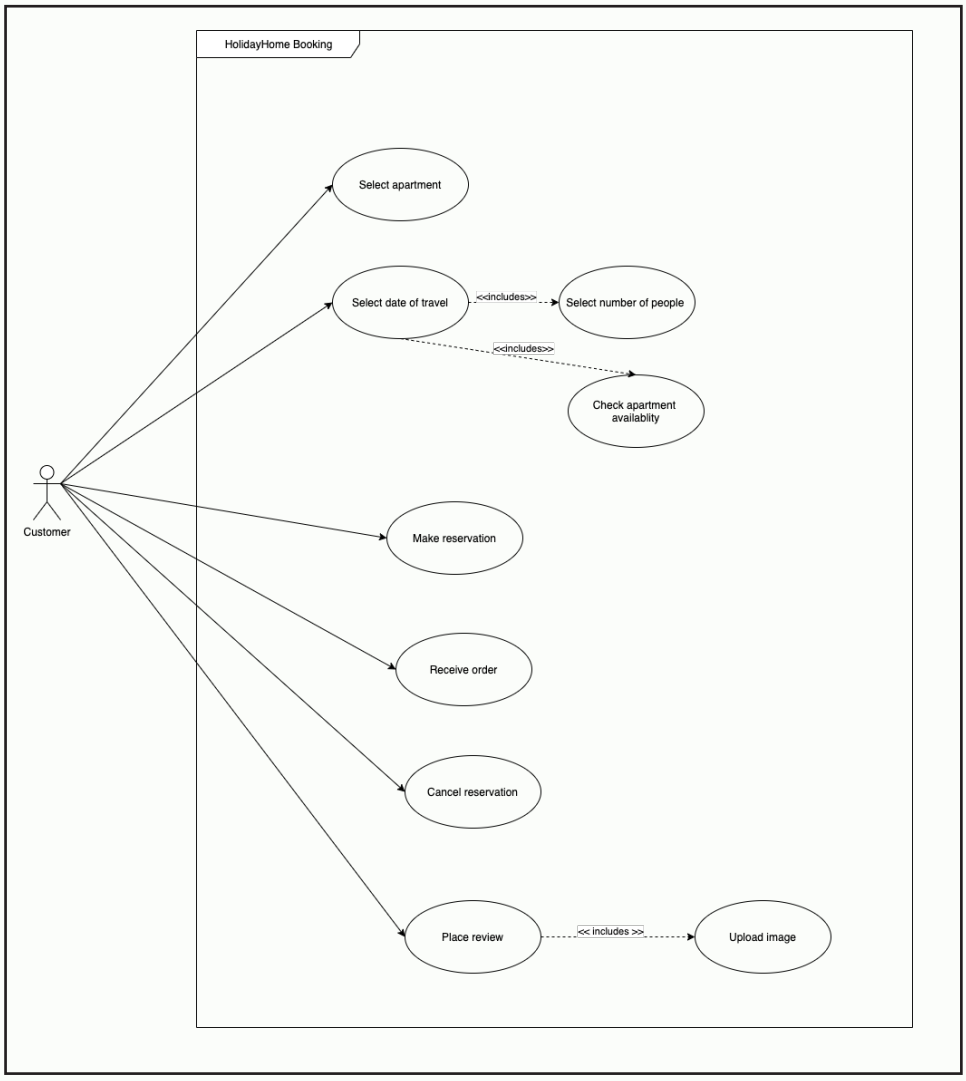
> Data Loading
An indication of data loading should be shown in order to indicate the user that the data is loading

> Design of the interface is simple and minimalistic
In order to keep it simple to the user, the application should have a minimalistic and simple look.

Chapter I

Functional Design

Use cases



Use Case:	**Sign up** *This use case describes how a new user signs up for the vacation home application*
Number:	1
Actor:	New user
Precondition:	-
Description:	New user needs to fill in its username, email and password
Postcondition:	Successful sign up, user can log in with their corresponding username and password

Chapter I

Functional Design

Use cases

Use Case:	**Log In** *This use case describes how the user logs in into the vacation home application*
Number:	1.1
Actor:	User
Precondition:	Existing user - user is required to have an username & password
Description:	- User can log in with using their corresponding username & password - Wrong username and password will result into an error message
Postcondition:	Correct username & password opens the corresponding user details and other functions depending on which role the user has. - Admin has access to all the reservation list - User has access to the review section

Use Case:	**Account details** *This use case describes how the users details are shown depending on the different types of users.*
Number:	2
Actor:	User
Precondition:	Existing user
Description:	After successfully logging in, the user can see its account details such as email, username and it's assigned role
Postcondition:	If the user is not a system admin the user can see the reservations

Use Case:	**Generate review** *This use case describes how the user is able to write a review*
Number:	3
Actor:	User
Precondition:	User needs to be logged in
Description:	The user can write a review and rate its stay
Postcondition:	The submitted review can be seen on the home screen

Chapter I

Functional Design

Use cases

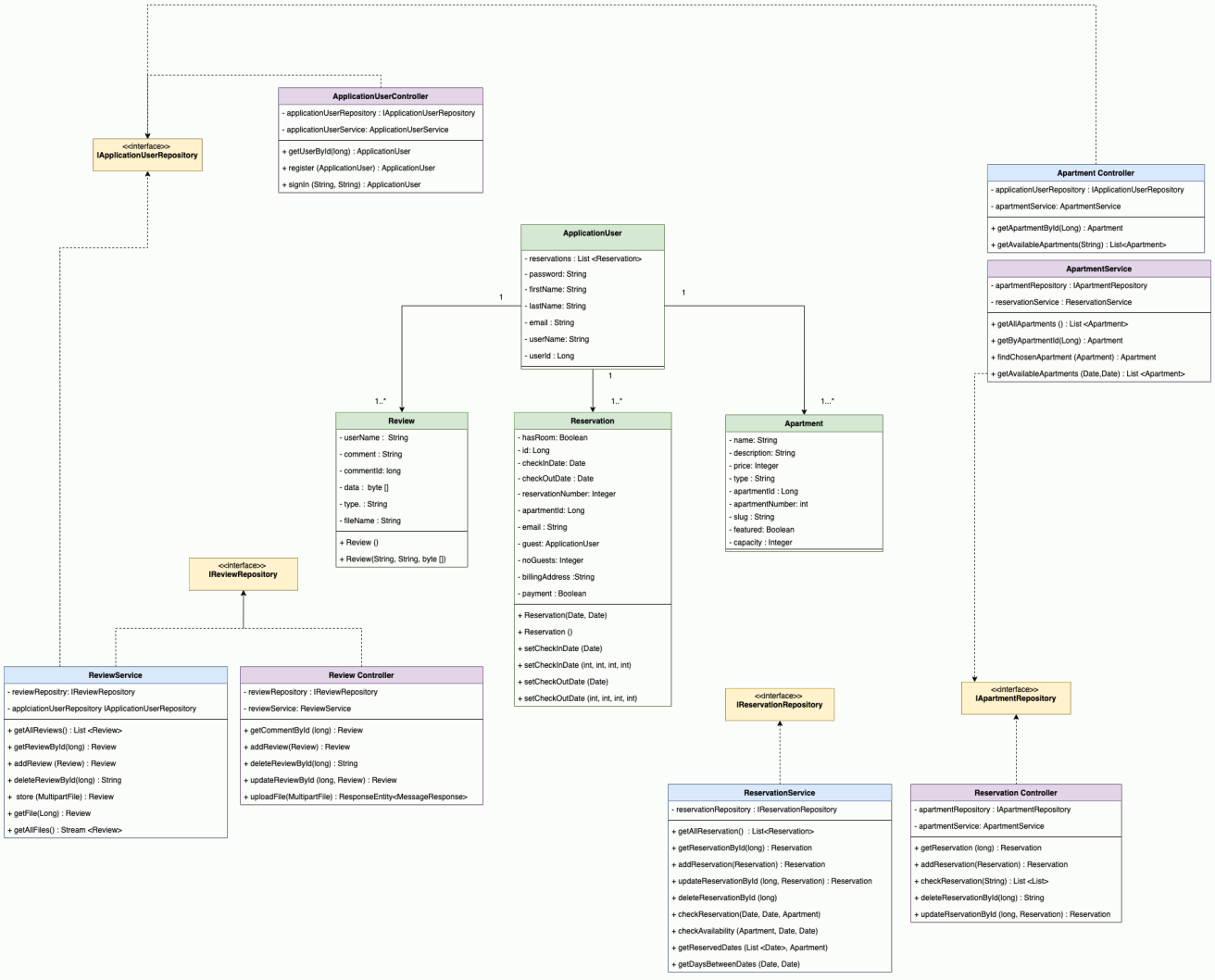
Use Case:	**Add reservation** *This use case describes how user can add a reservation*
Number:	4
Actor:	User
Precondition:	User needs to be logged in
Description:	- The user chooses an apartment - Selects the desired check in, check out dates and number of guests - Clicks on the make reservation button
Postcondition:	Confirmation message after the reservation is added

Use Case:	**Delete reservation** *This use case describes how the system admin can delete the reservation made by an user*
Number:	5
Actor:	User
Precondition:	The system admin needs to be logged in
Description:	- In the reservation list screen the system admin can see all the reservation details including: id, email, check-in, check-out dates and price - The admin can click on the delete button next to each reservation to delete / cancel the reservation
Postcondition:	The reservation does not exist anymore.

Chapter II

Technical Design

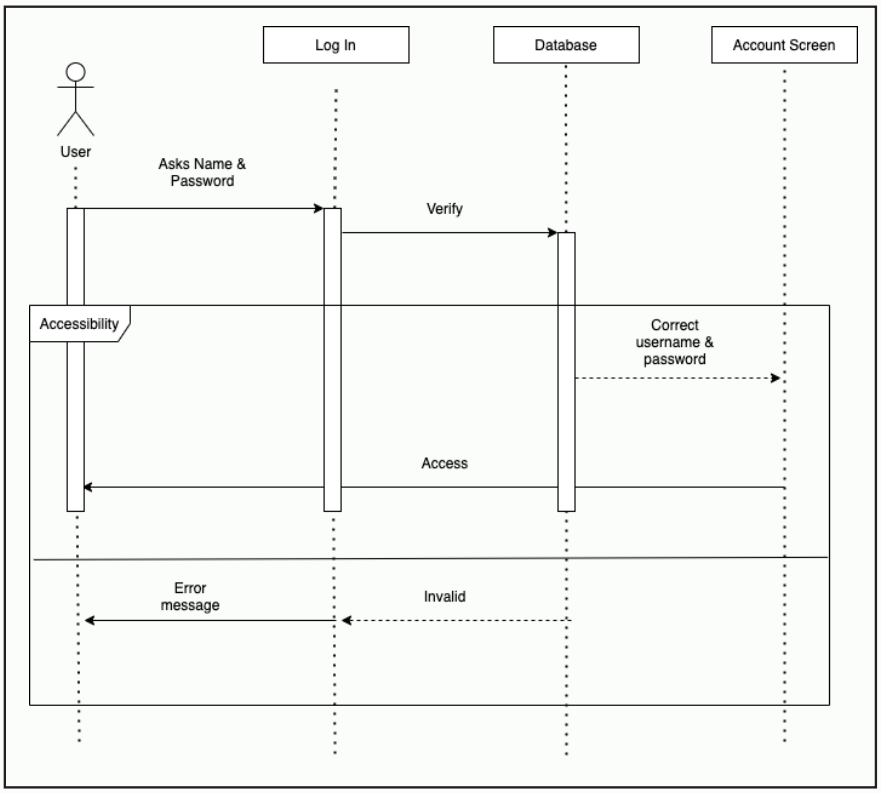
UML / Class diagram



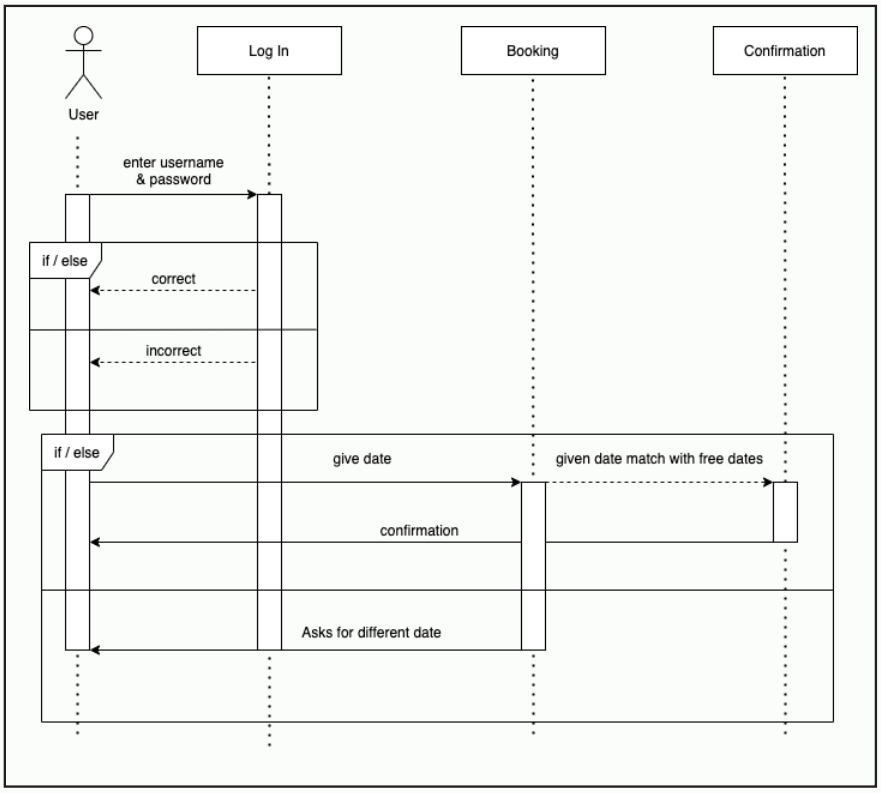
Chapter II

Technical Design

Sequence diagram



Log in sequence diagram

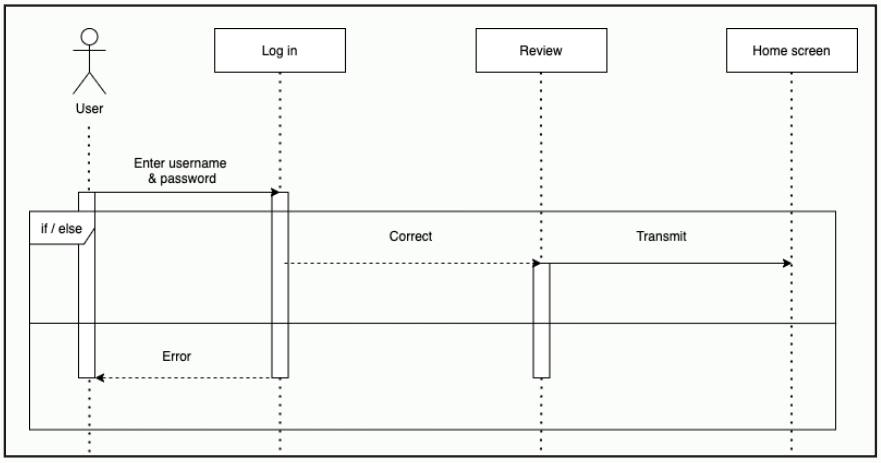


Reservation sequence diagram

Chapter II

Technical Design

Sequence diagram



Review sequence diagram

Chapter III

Accountability Document

In this chapter the accountability for different milestones in the back end and front end are going to be touched upon. Furthermore, certain technical decisions are going to be explained.

Front end

For the front end I used the following programming languages and technicalities :

> HTML

> CSS

One CSS File

In order to optimize loading time, I decided to work with one CSS file for the entire application. One CSS file helped to keep everything related to styling at one place.

> JavaScript

> Reactjs

> React Router Dom

The router allows the application to navigate between different components and can change the browser URL. For the vacationhome application the react router dom can switch between the following components : board user, board admin, home, login, register and room.

> React Context

In the beginning phase of the project I had for each room / apartment a React class. To make this more dynamic I used the React Context. React Context provided a possibility to share the values between components without passing every prop.

> React Date picker

In order to pick different check-in & check out date the react date picker was used.

> React Star Rating

For the review section, I chose for the react star rating to make it possible for the user to rate their stay.

Chapter III

Accountability Document

> React Validation

This library is able to validate the form that is used during log in and sign up. The React validation verifies the email when a new user wants to make an account.

> Axios

Allows to make HTTP calls to REST endpoints which is set up in the back end.

> Bootstrap

The CSS framework Bootstrap was applied for the log in, register and table of reservation components.

Front end reflection

In this paragraph the general learning lessons of my front end programming is going to be elaborated. Furthermore, I am going to touch upon the things I do not have in my application which would be adjusted in the next phase.

 # learning lesson 01

While trying to connect the database with the front end I first began with a local database with multiple children in a JSON document. Later in the project I wanted to put everything in PostgreSQL. However, SQL is a relation database and therefore not recommended to store sequenced items into one cell. In order to work around this problem, I now have one local database and one online database which is not ideal and should be changed in the next phase.

#learning lesson 02 - react router dom

Even though it was made possible through react router dom to switch between components, I disregarded the security behind the URL. Now the unauthorized user is able to switch to the admin website through the URL. In the next phase I would implement the Private Route element in order to restrict this from happening.

#learning lesson 03 - react date picker

For the next phase, I want to add a function which would disable certain unavailable dates in order to indicate to the user that these dates are not available.

#learning lesson 04 - make reservation

The main function should have been to make a reservation. However, due to time constraints I was not able to implement this yet. The back end already provides a REST api for this but I was not able to implement this function together with the available dates functions yet.

Chapter III

Accountability Document

Back end
In the backend I used the following programing language and technicalities:

> Java, Spring Boot

> PostgreSQL

> JPA

> Maven

Back end reflection
The most difficult tasks were the security element of the application and the reservation section. Hereby, I am detailing some of the insights which I have gathered during my programming process.

 # learning lesson 01 - WebsecurityConfigure Adapter class
Is the CRUX to the security application. This provides a HTTP / security configuration to configure CORS and other session managements in the application.

#learning lesson 02 - OncePerRequestFilter
This provides a doFilterInternal method which includes the parsing and validating of the JWT tokens.

#learning lesson 03 - CheckReservation Function
The check reservation function checks if the dates of the requested reservation is already occupied. This took a long time due to the fact that I needed to add multiple if functions to check whether the dates fall between the starting and ending points.

Chapter III

Rest end points

Methods	URLS	Actions
GET	/api/apartment	
GET	/api/apartment/ {apartmentId}	
POST	/api/auth/signin	{ "username" : "admin", "password" : "123456" }
POST	/api/auth/signup	{ "username" : "admin", "email" : "admin@admin.com", "password" : "123456", "role" : ["admin"] }
POST	/api/ availableapartments	
GET	/api/files	
GET	/api/reservation/	
POST	/api/reservation/	{ "reservationNumber" : 156702, "price": 12.95, "checkInDate":"2012-04-23T18:25:43.511Z", "checkOutDate": "2013-04-23T18:25:43.511Z", "noGuests": 12, "payment": true, "billingAddress": "Teststraat", "apartmentId": 1 }
GET	/api/reservation/{id}	
PUT	/api/reservation/{id}	{ "id":"1", "apartmentName":"Number2", "hasRoom":"true",

Chapter III

Rest end points

		<pre>"price":"400"
		}</pre>

DELETE	/api/reservation/{id}	
GET	/api/review	

POST	/api/review	<pre>{
		"commentId" : "1",
		"comment" : "Test Review",
		"userName" : "ReviewTester",
		"Rating" : 3
		}</pre>

GET	/api/review/{commentId}	
GET	/api/test/admin	
GET	/api/test/all	
GET	/api/test/user	

POST	/api/updateReview/{oldCommentId}	<pre>{
		"commentId" : "3",
		"comment" : "Hello123"
		}</pre>

POST	/api/upload	
GET	/api/user/{id}