Technical Interaction Design, Final Report

KSTEIND1KU

Jonas Balin (jbal@itu.dk)

With major contributions by
Amalie Frøling Pedersen (amap@itu.dk)
Gustav Nørgaard Pedersen (gupe@itu.dk)

2022-02-23

Contents

1 Introduction

This report will detail the design and implementation of a browser-based application designed based on a case. The structure of the report is as follows:

- Presentation of problem statement
- Initial design phase (tasks, virtual windows, CREDO, LoFi prototype)
- Usability testing
- HiFi prototype
- Implementation
- Reflection on process

It is the purpose of this report to outline the design process in a manner such that the reader is familiar with the resulting product, its benefits and limitations, and the rationales behind choices made.

The structure of the report generally follows the iterative design process of the project. Note that the iterative nature does not guarantee consistency between steps, as choices made earlier could be subject to change in latter stages. This report details the development process chronologically without retroactively altering elements to fit later conclusions. These deviations should be interpreted as part of the iterative design process. Where relevant, deviations from prior decisions will be discussed.

This report is written exclusively by Jonas Balin during January/February of 2022. However, the work described in the report is mostly based on work done by Balin, Gustav Pedersen, and Amalie Pedersen through Autumn and Winter 2021. This is mentioned as it impacts the latter stages of the design phase noticeably. To ensure proper accreditation, the report will delineate between the work conducted cooperatively and the work conducted individually, where appropriate.

Unless otherwise specified, the theories and tools used for the initial design phases are highly based on descriptions found in Søren Lauesen's book, *User Interface Design* [1].

Accompanying this text, there is an appendix which contains higher resolution screenshots of the HiFi prototypes described in chapter 8.1 and 8.2.

2 Problem Statement - ScanCar

This section is a copy of the ScanCar case description. If you are already familiar with it, you can proceed to task descriptions

ScanCar is a country-wide car rental agency with a fleet of more than 2000 cars and 37 rental offices. A rental office comprises a reception that rents out the cars, a parking lot where the cars are stored, and a service center that makes the cars ready for the next rental.

The country is divided into regions. Within a region, cars can be moved from office to office with short notice. The reason may be demand for cars at a certain office, but also the fact that some offices, for instance in city areas, have parking space for rather few cars. This process is called short-term allocation of cars. (Long-term allocation exists as well. It may involve moving cars between regions, but it is not part of this assignment.)

As an example, the rental offices in Copenhagen City have few parking slots. As a result, cars are transferred several times a day between the rental office at the airport, with ample parking space, and the other rental offices in the region.

The reception is headed by the rental manager who participates in the daily reception tasks. He is also responsible for ordering transfers of the necessary cars.

Today he manages the transfers manually by means of a spreadsheet for the office. He tries to get an overview of the available cars for the next 48 hours and calls the other offices to have spare cars transferred. This is cumbersome.

At present an office spends two months to train a new employee. A large part of the time is spent learning to use the administrative systems. In the old days this was no problem because staff tended to stay with ScanCar for years. Lately, however, it has been harder to keep staff, so the long training time becomes an issue.

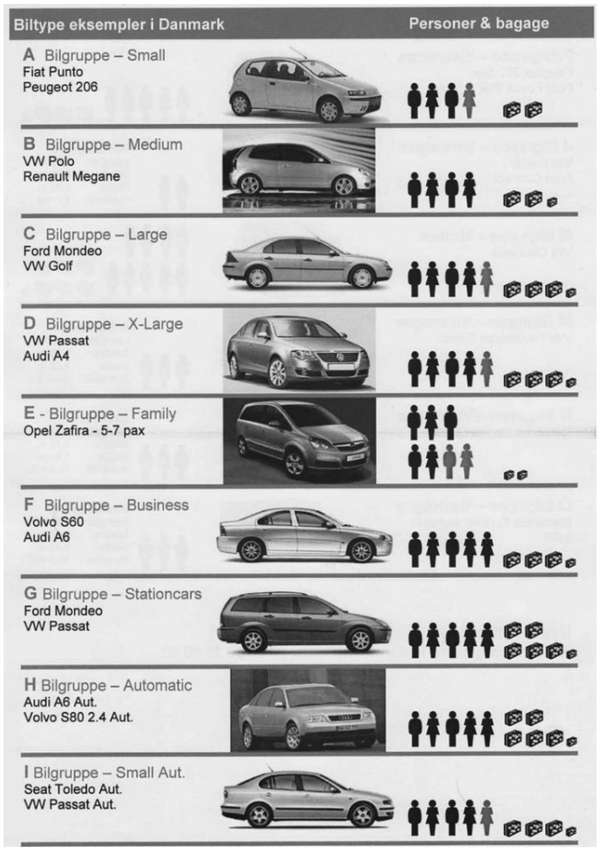
2.1 About the cars

About 20-30 customers pick up a car each day at an average rental office. Sometimes, for instance around public holidays, more than 50 cars may be picked up.

Cars are divided into approximately 10 car groups, e.g. small, medium, business, etc. as shown in Figure 1. Notice that each group may contain several car models. All cars in a group have the same price.

The ScanCar head office records the following for each car:

- Car model (e.g. Volks Wagen 1.9TDI)
- License number (e.g. VY 53 342)
- Color (a three-letter code, e.g. BLK for black)
- Fuel type (D for diesel and P for Petrol)
- Number of doors (e.g. 2, 4 or 5)



Biltype eksempler i Danmark

Personer & bagage

A Bilgruppe – Small
Fiat Punto
Peugeot 206

B Bilgruppe – Medium
VW Polo
Renault Megane

C Bilgruppe – Large
Ford Mondeo
VW Golf

D Bilgruppe – X-Large
VW Passat
Audi A4

E - Bilgruppe – Family
Opel Zafira - 5-7 pax

F Bilgruppe – Business
Volvo S60
Audi A6

G Bilgruppe – Stationcars
Ford Mondeo
VW Passat

H Bilgruppe – Automatic
Audi A6 Aut.
Volvo S80 2.4 Aut.

I Bilgruppe – Small Aut.
Seat Toledo Aut.
VW Passat Aut.

Figure 1: Car Groups (Danish: Bilgruppe)

2.2 Car booking

Most customers book the car online, but it must be done at least 24 hours in advance. Otherwise it may not be possible to transfer the car. Because cars can be moved around within 24 hours, it doesn't matter where in the region the car is at present. The short-term allocation ensures that a suitable car is available at the right spot at the right time. (Assuming that everything works as planned.)

The receptionists and the rental manager can see the online bookings on the screen. Usually they don't care about them until the customer picks up the car.

Many customers book by phone, for instance because they want the car sooner than 24 hours or because they want to change an on-line booking.

Customers can not book a specific model, but only a car in a specific car group. (Otherwise the fleet is not used efficiently, for instance because too many cars have to be transferred.)

The customer specifies where he wants to pick up the car and when (date and time). He also specifies where and when he wants to return the car. It may happen that cars are returned to another office than planned. This is accepted if the customer gives notice early on.

All offices are open every day from 7 to 21 (14 hours). Time is counted in two-hour periods so that cars may be picked up at 8, 10, 12, etc. Similarly car return may take place no later than 8, no later than 10, etc. When a car has been returned, two hours are set off for making it ready for the next pickup. So a car returned no later than 10 may be picked up at 12 from the same office. If it has to be moved, more time may be needed. Experienced service staff knows how much.

In busy periods, all cars in a specific group may be booked. The customer may then choose another group or another period. In the rental office, you may hear phone dialogs such as this one:

We don't have a medium car at 10:00, but if you can wait until 12, you may get one. But it must be returned no later than July 22nd, 10:00. By the way, if you like a large car, you can have it whenever you want

The receptionists also record the customer's name, address, age, phone, and driver's license ID. Usually they check whether the customer already is on file. In that case they check whether the data is still okay. They may send the customer a confirmation with a booking number and the name, address and phone of the pick-up office.

2.3 Car pickup

When the customer arrives to pick up the booked car, he may freely choose between the cars that are ready for pick up - assuming that they are in the group he has booked. Usually he is highly concerned about which car he gets. He might for instance prefer a Golf or a red car if available.

If no car is ready in the group he booked - in spite of all planning - the receptionist may upgrade the customer. This means that he gets a car in a higher group than the one he booked, but at the same price as the lower group.

The receptionist also checks the customer data, for instance the driver license ID. If necessary, he changes the data.

Next he guides the customer to the parking lot, checks the car with the customer, checks that the fuel tank is full, and notes the mileage. (Back at the office, he records the data in the system.) Finally he hands the customer the rental agreement and the car keys.

2.4 Walk-in

Walk-in customers ask for a car without having booked one in advance. A walk-in may choose any car available for pick up, unless it is needed for some booking. And he must return it before it is needed for some booking.

In addition, the same things take place as when a booked car is picked up. The receptionist records the customer data, guides the customer to the car, etc.

2.5 Car return

When the customer returns the car, the receptionist inspects the car and notes the mileage. He checks that the fuel tank is full. If not, he estimates how much is left. Back at the office, he records the data in the system for billing purposes.

2.6 Short-term allocation

Each rental office tries to keep some spare cars for walk-ins. The spare cars may also be used to cover unexpected situations such as larger car repairs and cars returned too late.

Today it is hard to keep track of spare cars. Using a spreadsheet, the rental manager computes the number of booked cars period by period, the number of returned or moved cars, and when they are ready. When additional cars are needed from other rental offices, the rental manager calls around to hear who have spare cars. They agree on how to transfer the cars, either on a large car-pickup or having a trainee drive them. Usually some offices end up having too many cars and some too few. The result is a bad utilization of the car fleet.

However, the rental managers are very good at estimating how many walk-in cars they can rent out at which time. The problem is to ensure that they are available.

The future system should let managers request or release spare cars. Several hours later, the system can tell him how many of these cars he will receive or send away, and at what time. To do this, the system must know all bookings and pick-ups in the region, and the expected return times and places. In dialog with the service centers, it can then plan when to transfer the cars and to where.

(The actual car transfers are handled by the service centers. On the screen they will see the cars to be moved. They also record when a moved car arrives and when a returned car has been made ready. You don't have to deal with the service centers here.)

The receptionists must at any time be able to see which specific cars are available in the car park right now. They might for instance see that two large cars are ready: Ford Mondeo UX 52 312 and VW Golf VY 53 342. They can also see that one of them is needed for a booking (it hasn't yet been decided which of them). The other is a spare car that might be used for walk-in.

2.7 Car states

A car can be in one of these states:

- Ready: It is available for pickup.
- Rented: It has been picked up.
- Returned: It has been returned, but is not yet ready for pickup.
- Transfer: It is on its way to another office.
- Unavailable: It cannot be used for some time, for instance because it is under repair.

In order to make statistics, ScanCar keeps track of all state changes. This will also allow the new system to calculate average car transfer times, and in this way optimize short-time allocation.

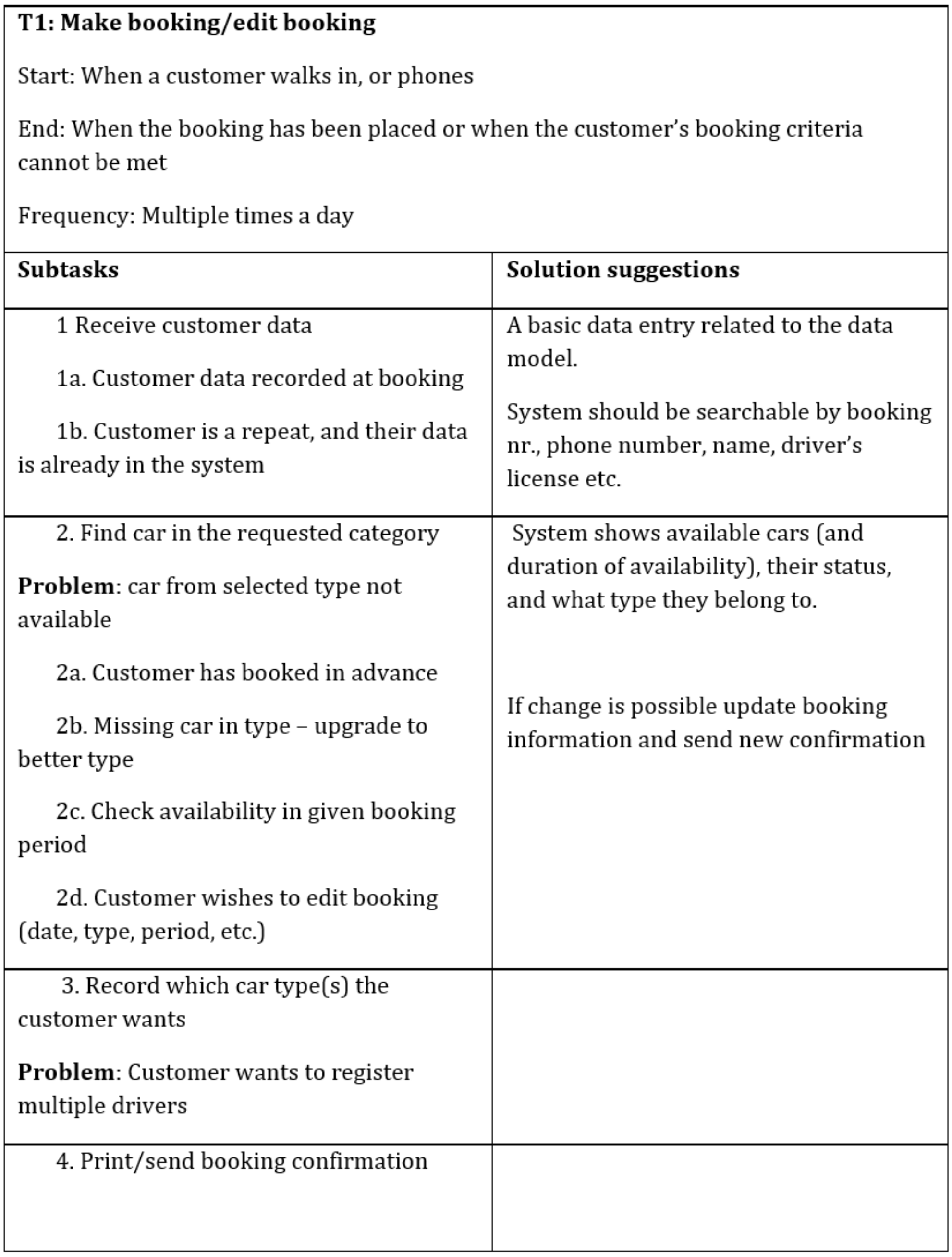
3 Task Descriptions

This section will present a description of tasks that the staff of ScanCar (and thereby the users of the coming software) could meet in their work-time. Typically, tasks are developed through user involvement. Given the hypothetical nature of the case, these tasks are based on assumptions and the problem statement. Their benefit is in mapping the primary purpose of the system-to-be and serving as use-case scenarios for usability tests.

The formulation of these tasks have attempted to follow the ”rules” for good tasks. The include:

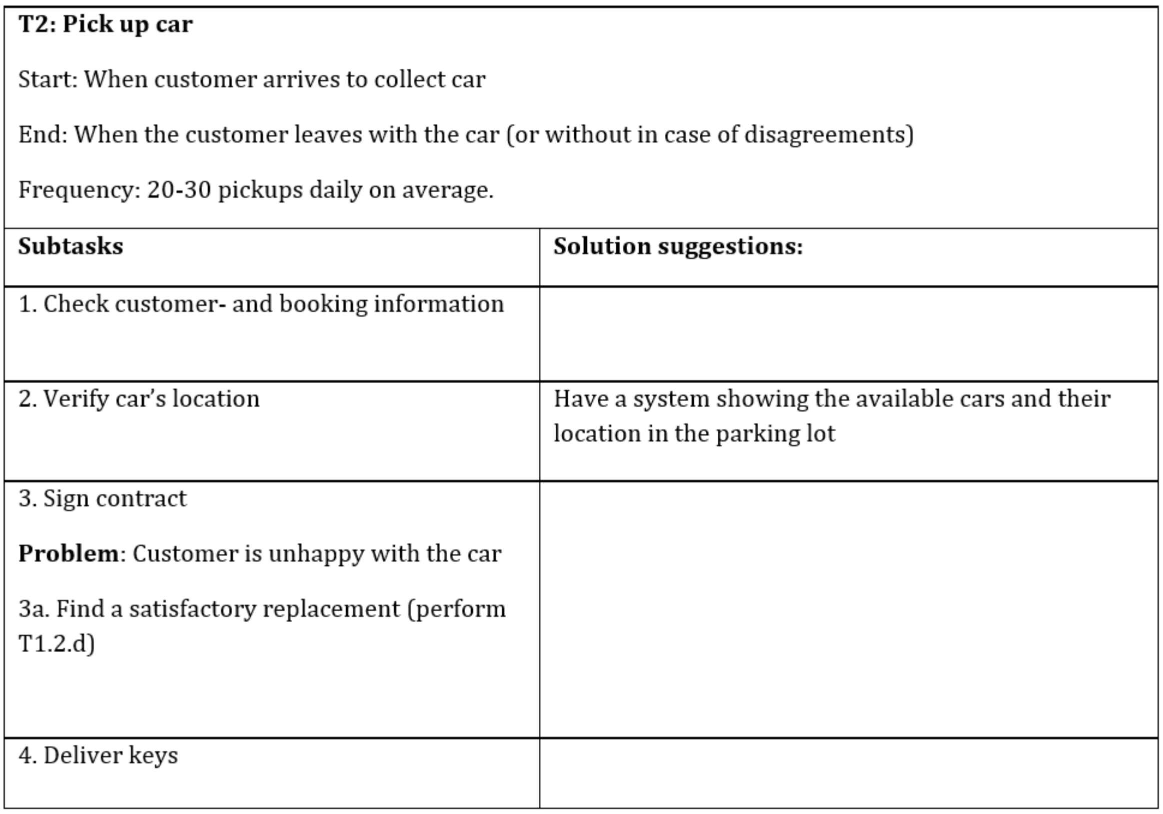
- Tasks must be enclosed - each task ”deserves a coffee break”
- Small related tasks are described as sessions without breaks
- Hide who does what, avoid imperative language
- Do not program: ”`if x then y else z`”

For the tasks, solution suggestions can also be written. This has been done sparingly, only when the suggested solution seems intuitive (with the understanding that intuition can be fickle).



T1: Make booking/edit booking Start: When a customer walks in, or phones End: When the booking has been placed or when the customer's booking criteria cannot be met Frequency: Multiple times a day	
Subtasks	**Solution suggestions**
1 Receive customer data 1a. Customer data recorded at booking 1b. Customer is a repeat, and their data is already in the system	A basic data entry related to the data model. System should be searchable by booking nr., phone number, name, driver's license etc.
2. Find car in the requested category **Problem**: car from selected type not available 2a. Customer has booked in advance 2b. Missing car in type – upgrade to better type 2c. Check availability in given booking period 2d. Customer wishes to edit booking (date, type, period, etc.)	System shows available cars (and duration of availability), their status, and what type they belong to. If change is possible update booking information and send new confirmation
3. Record which car type(s) the customer wants **Problem**: Customer wants to register multiple drivers	
4. Print/send booking confirmation	

Figure 2: Making/editing booking task



T2: Pick up car

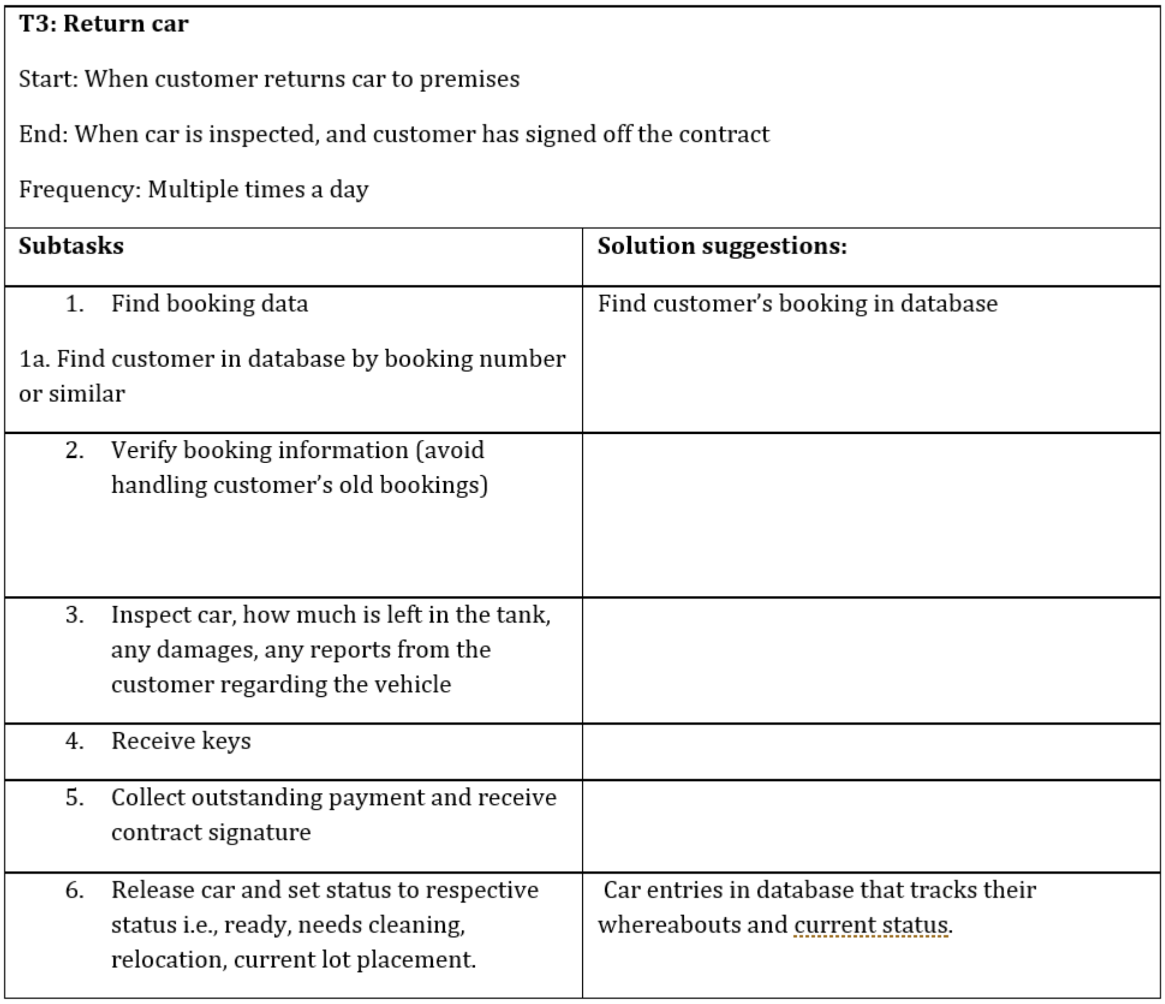
Start: When customer arrives to collect car

End: When the customer leaves with the car (or without in case of disagreements)

Frequency: 20-30 pickups daily on average.

Subtasks	**Solution suggestions:**
1. Check customer- and booking information	
2. Verify car's location	Have a system showing the available cars and their location in the parking lot
3. Sign contract **Problem**: Customer is unhappy with the car 3a. Find a satisfactory replacement (perform T1.2.d)	
4. Deliver keys	

Figure 3: Picking up car task



T3: Return car

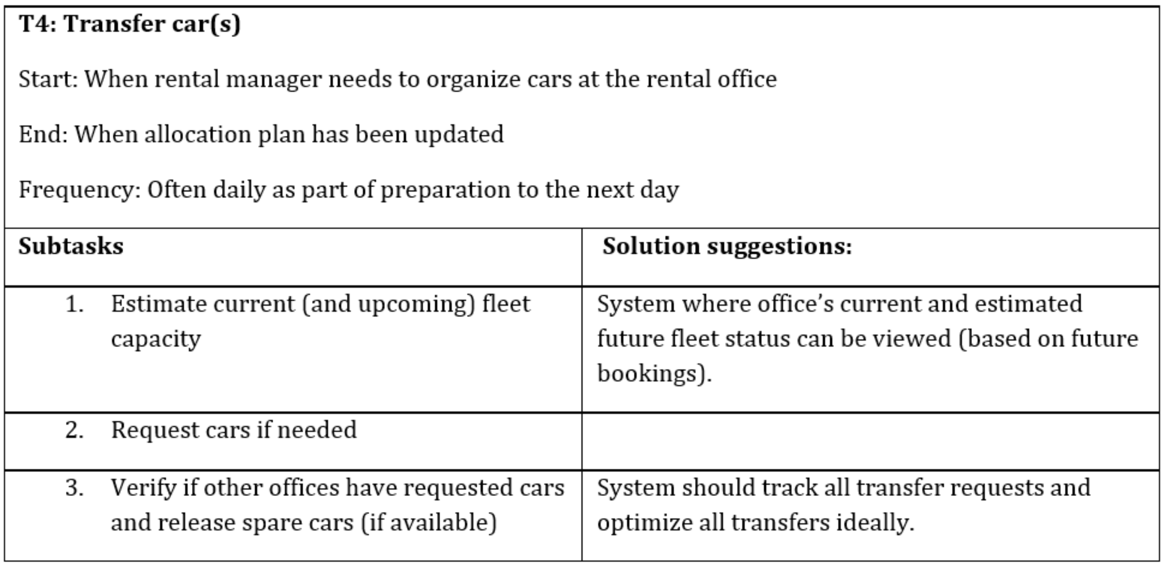
Start: When customer returns car to premises

End: When car is inspected, and customer has signed off the contract

Frequency: Multiple times a day

Subtasks	**Solution suggestions:**
1. Find booking data 1a. Find customer in database by booking number or similar	Find customer's booking in database
2. Verify booking information (avoid handling customer's old bookings)	
3. Inspect car, how much is left in the tank, any damages, any reports from the customer regarding the vehicle	
4. Receive keys	
5. Collect outstanding payment and receive contract signature	
6. Release car and set status to respective status i.e., ready, needs cleaning, relocation, current lot placement.	Car entries in database that tracks their whereabouts and current status.

Figure 4: Returning car task



T4: Transfer car(s)

Start: When rental manager needs to organize cars at the rental office

End: When allocation plan has been updated

Frequency: Often daily as part of preparation to the next day

Subtasks	**Solution suggestions:**
1. Estimate current (and upcoming) fleet capacity	System where office's current and estimated future fleet status can be viewed (based on future bookings).
2. Request cars if needed	
3. Verify if other offices have requested cars and release spare cars (if available)	System should track all transfer requests and optimize all transfers ideally.

Figure 5: Transferring car task

4 Entity-Relationship Diagram

The entity-relationship diagram is a data model with the purpose of showing entities' relationships with each other. It is important to note that this ERD is not hypothetical but rather an illustration of the entities, their attributes, and relations to each other expressed in the problem statement. The benefit of the ERD is that it can give an overview of relations and attributes which can be highly useful for future system design.

Figure 6 shows the ERD. Figure 7 shows a legend of the ERD.

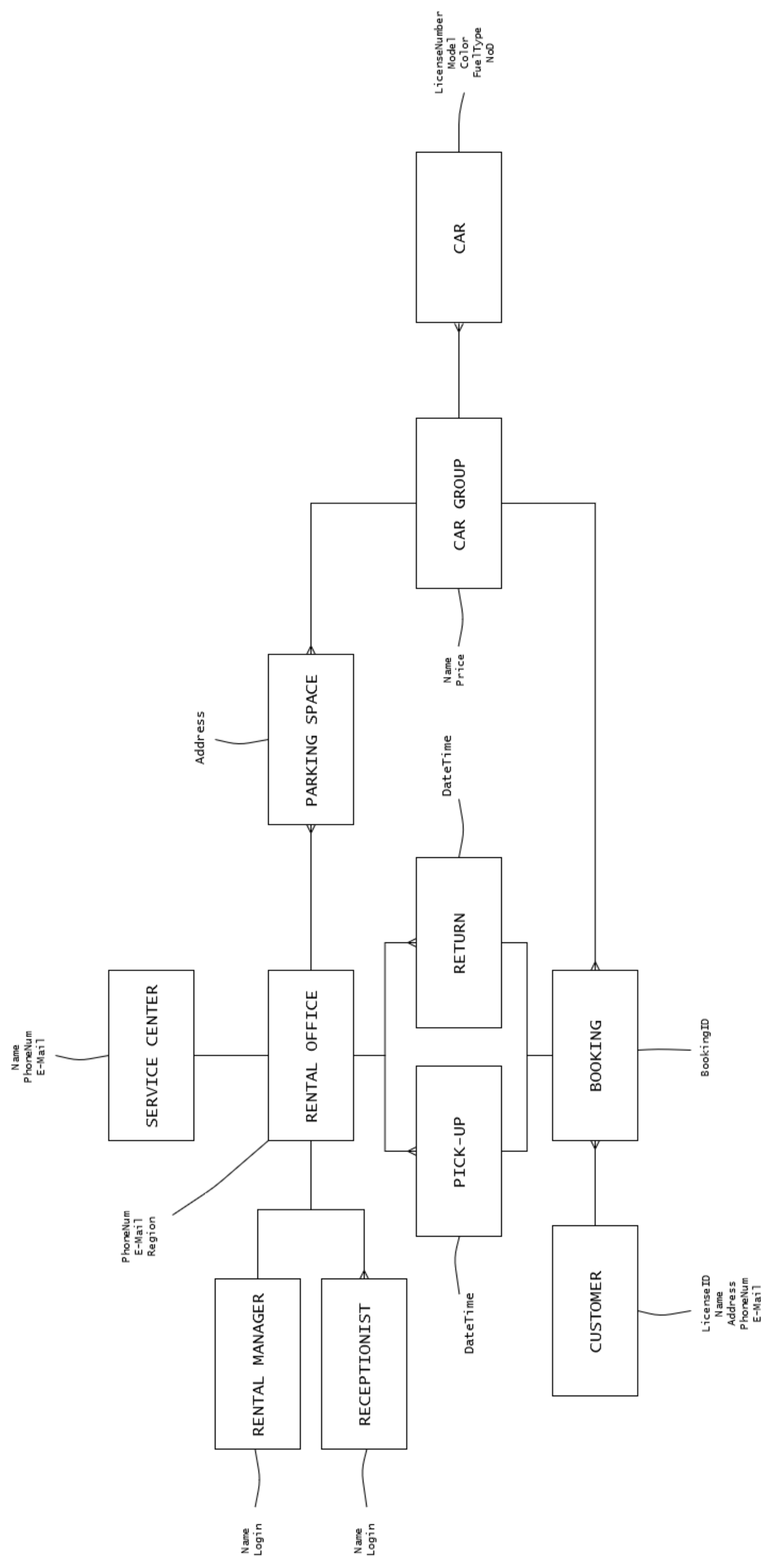


Figure 6: Entity-relationship diagram

CUSTOMER: One customer may have more bookings. A customer is connected to a rental office through the bookings' pick-up and returns.

BOOKING: A booking has one customer. A booking always has a pick-up and return, both of which are associated with a rental office (but not necessarily the same). A booking is also affiliated with a car group, which determines what kind of car the customer gets (depending on which car is available in the lot affiliated with the pick-up office).

CAR GROUP: A car group can have multiple bookings, and multiple cars. A car group may also be associated with multiple parking spaces (if five cars from group A are in five different lots). With this logic, car groups are then unique to their parking space. This serves to create a link between a specific car group to a specific rental office.

CAR: Each car has one car group. Through this connection, each car's booking and parking space can be defined.

PARKING SPACE: Each parking space represents a single space which can contain a car. Each parking space has one car group (and by extension one car), and is associated with one rental office.

RENTAL OFFICE: Each rental office may have multiple parking spaces. Each parking space associated with each rental office is used to determine which car groups (and thereby cars) the office currently has. A rental office may have multiple pick-ups and returns. Each pickup and return is linked with a booking. A rental office has one rental manager, and may have multiple receptionists. These are differentiated because a rental manager may have more privileges than the others (e.g. transfers). Each rental office has one service center for car maintenance.

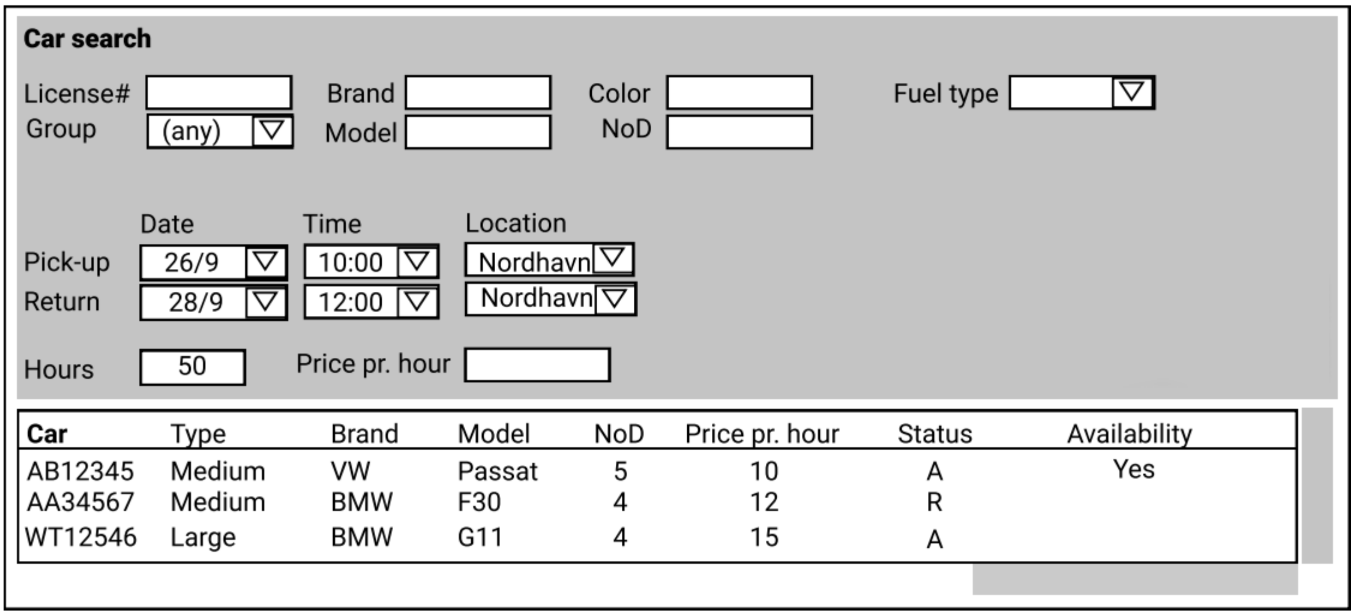
RENTAL MANAGER AND RECEPTIONIST: These are associated with a rental office. Through this connection, they are able to access all pertinent information, e.g. pick-up and return dates, parking space capacity, and so on.

Figure 7: Entity-relationship diagram legend

5 Virtual Windows CREDO

5.1 Virtual Windows

Virtual windows represent user-oriented persistent data. The essential purpose of virtual windows is to illustrate the data that different tasks need to allow the user to view or manipulate. Initially, virtual windows are not to include any functions or illustrations of these (e.g. buttons). In latter design stages, functions may be added to virtual windows, which are executed by buttons.



Car search

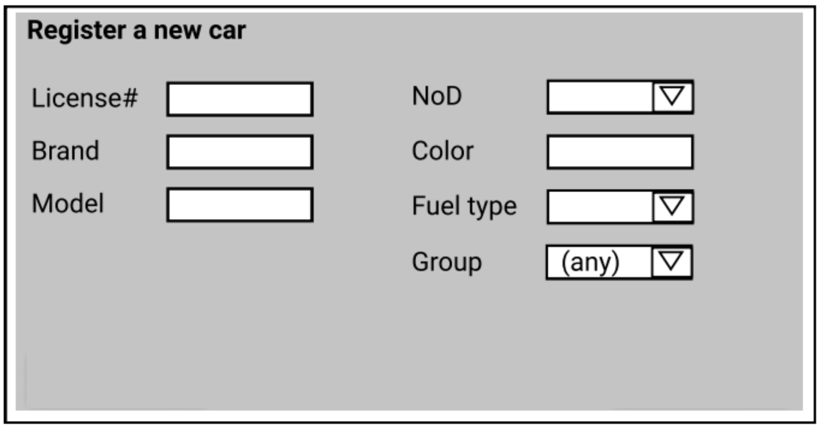
License#		Brand		Color		Fuel type	
Group	(any)	Model		NoD			

	Date	Time	Location
Pick-up	26/9	10:00	Nordhavn
Return	28/9	12:00	Nordhavn

Hours: 50   Price pr. hour:

Car	Type	Brand	Model	NoD	Price pr. hour	Status	Availability
AB12345	Medium	VW	Passat	5	10	A	Yes
AA34567	Medium	BMW	F30	4	12	R	
WT12546	Large	BMW	G11	4	15	A	

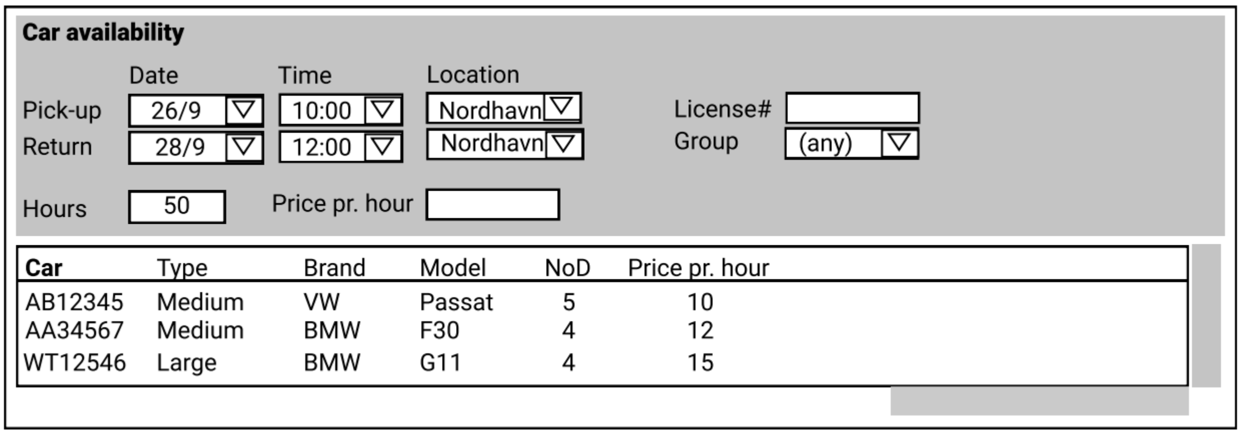
Figure 8: Car search virtual window



Register a new car

License#		NoD	
Brand		Color	
Model		Fuel type	
		Group	(any)

Figure 9: Car registry virtual window



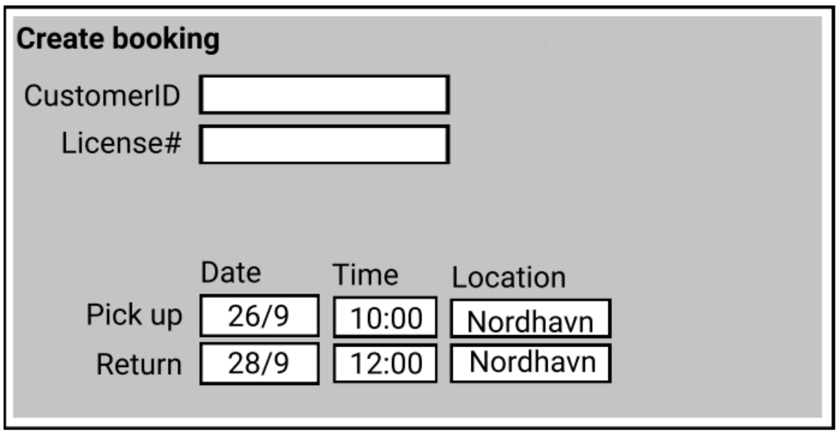
Car availability

	Date	Time	Location		
Pick-up	26/9 ▽	10:00 ▽	Nordhavn ▽	License#	
Return	28/9 ▽	12:00 ▽	Nordhavn ▽	Group	(any) ▽

Hours: 50   Price pr. hour:

Car	Type	Brand	Model	NoD	Price pr. hour
AB12345	Medium	VW	Passat	5	10
AA34567	Medium	BMW	F30	4	12
WT12546	Large	BMW	G11	4	15

Figure 10: Car overview virtual window



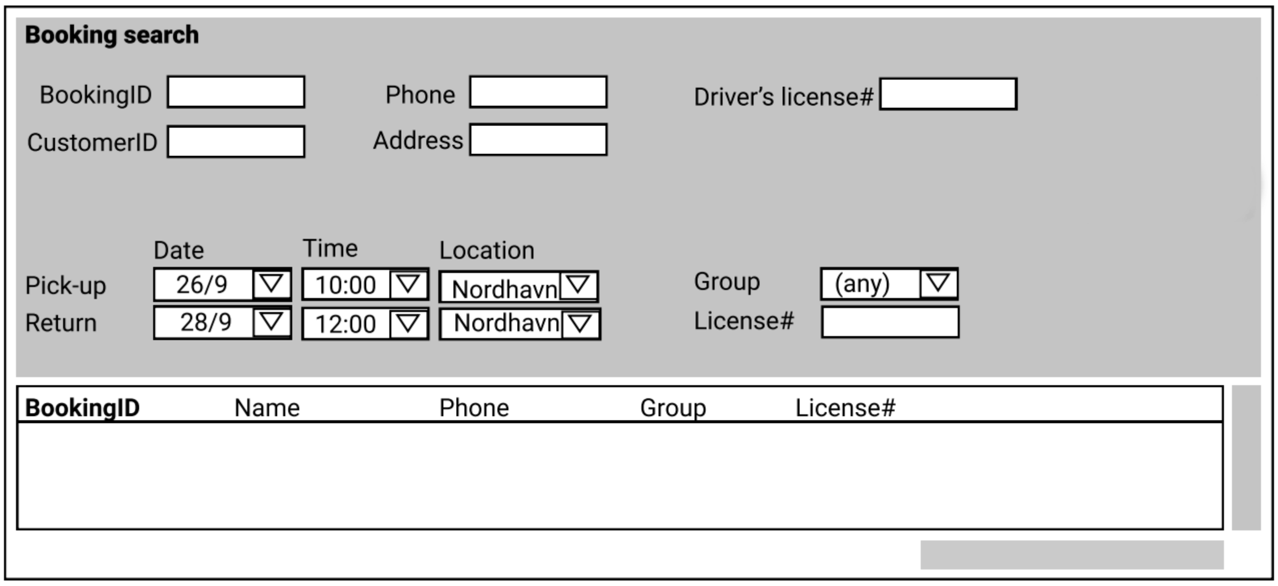
Create booking

CustomerID:

License#:

	Date	Time	Location
Pick up	26/9	10:00	Nordhavn
Return	28/9	12:00	Nordhavn

Figure 11: Booking creation virtual window



Booking search

BookingID: Phone: Driver's license#:

CustomerID: Address:

	Date	Time	Location		
Pick-up	26/9 ▽	10:00 ▽	Nordhavn ▽	Group	(any) ▽
Return	28/9 ▽	12:00 ▽	Nordhavn ▽	License#	

BookingID	Name	Phone	Group	License#

Figure 12: Booking search virtual window

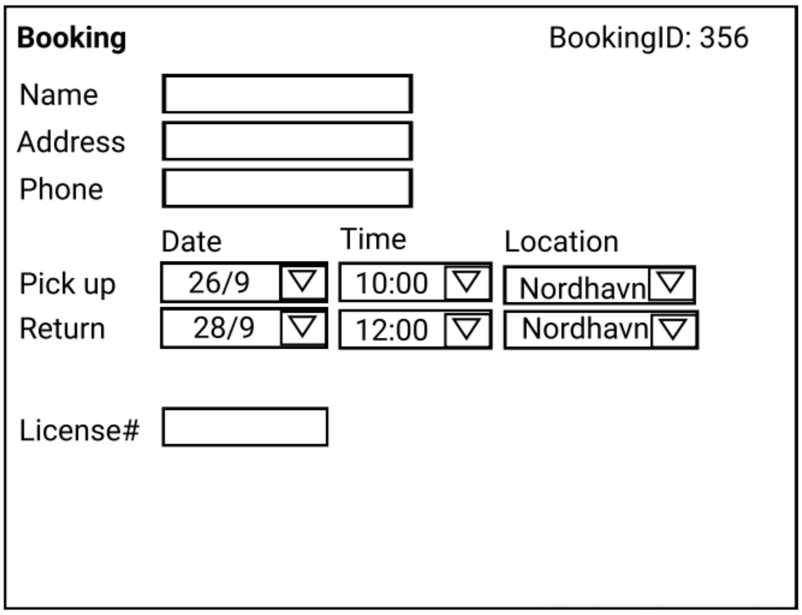


Figure 13: Booking edit virtual window

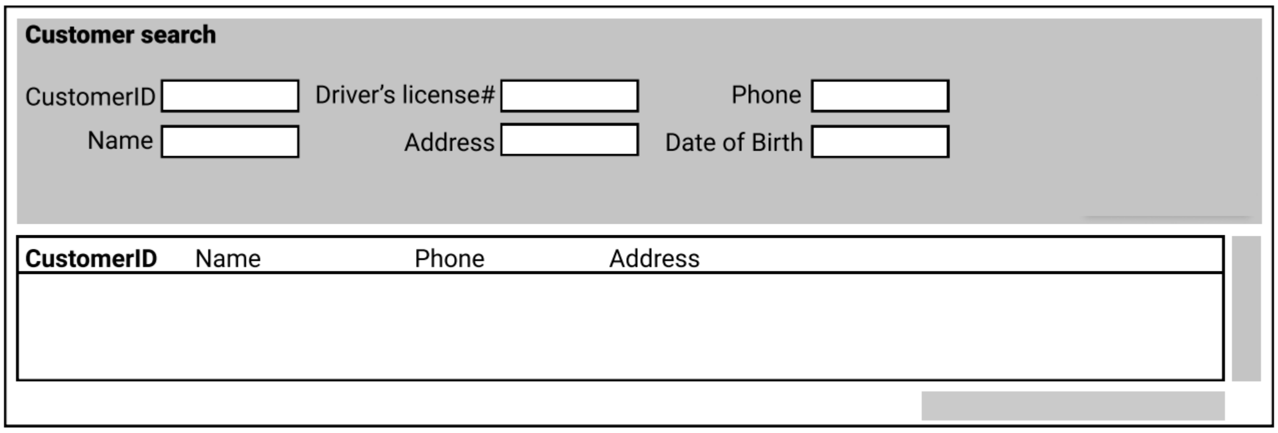


Figure 14: Customer search virtual window

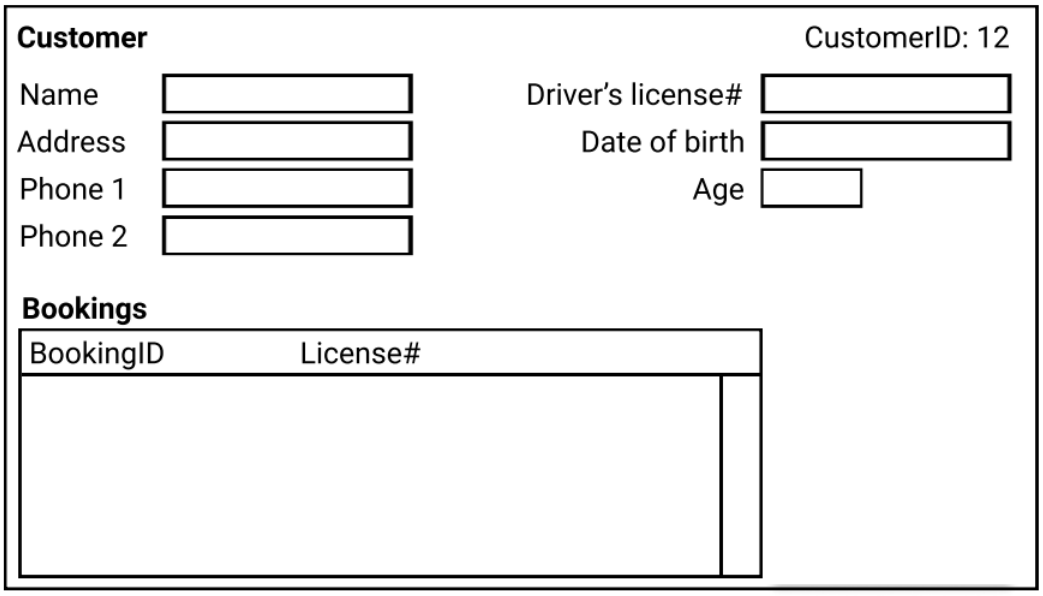


Figure 15: Customer edit virtual window

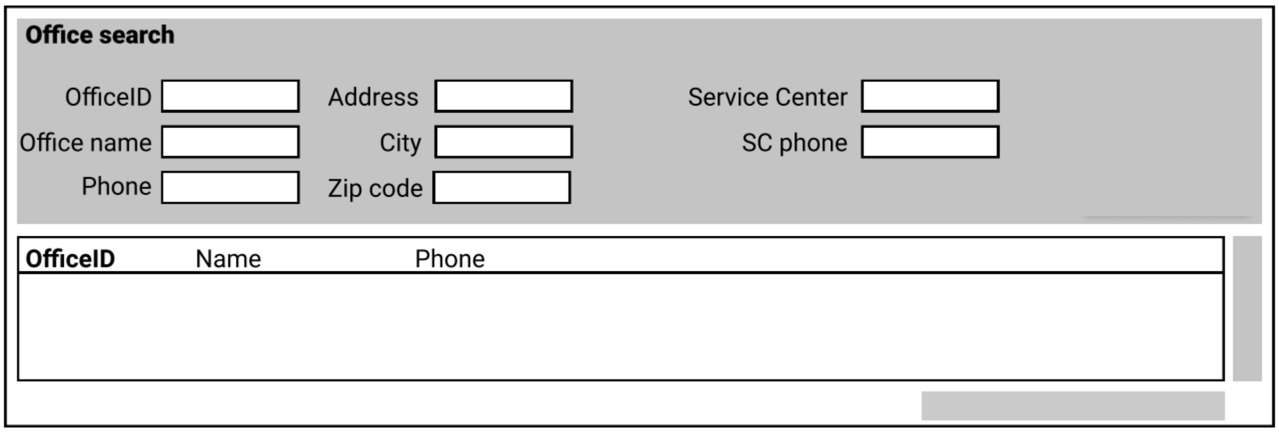


Figure 16: Office search virtual window

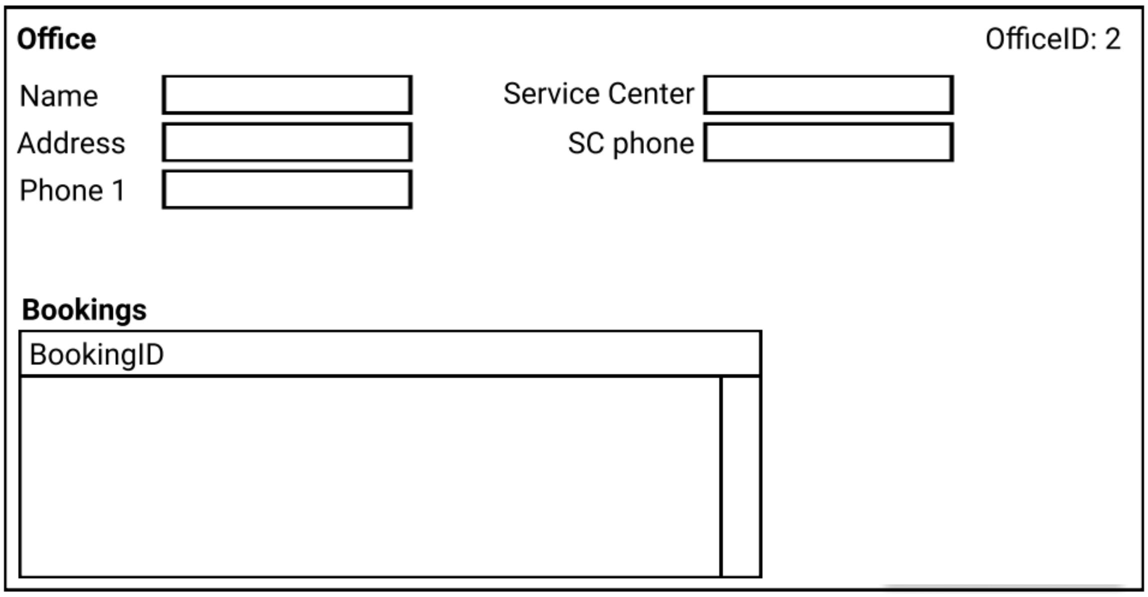
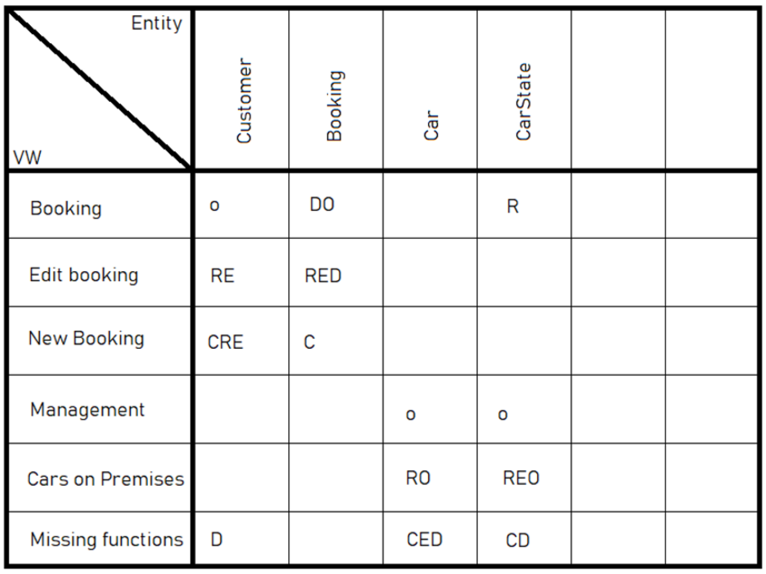


Figure 17: Office details virtual window

The virtual windows shown were developed during the initial design phases and were briefly retouched before the proceeding design phase. Due to changes in direction in the latter design phases, the virtual windows highlight elements that have since become de-emphasized to limit the scope. These include the office virtual windows, which were therefore not considered for CREDO checks.[1]

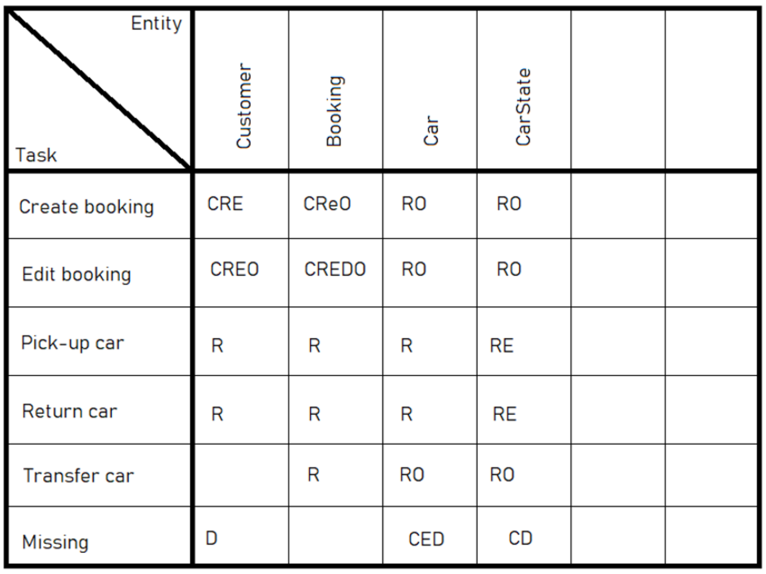
5.2 CREDO

The CREDO[2] method can be described as a matrix giving an overview of virtual windows' and tasks' required data manipulation and overview. The overview that a CREDO matrix can give is useful in identifying potential gaps of data, and mapping functions to virtual windows.



Entity / VW	Customer	Booking	Car	CarState		
Booking	o	DO		R		
Edit booking	RE	RED				
New Booking	CRE	C				
Management			o	o		
Cars on Premises			RO	REO		
Missing functions	D		CED	CD		

Figure 18: CREDO check over virtual windows



Entity / Task	Customer	Booking	Car	CarState		
Create booking	CRE	CReO	RO	RO		
Edit booking	CREO	CREDO	RO	RO		
Pick-up car	R	R	R	RE		
Return car	R	R	R	RE		
Transfer car		R	RO	RO		
Missing	D		CED	CD		

Figure 19: CREDO check over tasks

[1]Though the customer windows become relevant again in the further developments of the application, which will be detailed.

[2]Create, Read, Edit, Delete, Overview

As the CREDO checks lay out, certain functionalities are missing from both the virtual windows and tasks. These include deletion of customers, creating, editing and deletion of cars, and creation and deletion of cars. It could be theorized that these would be beneficial to have (e.g. a car is out of commission and should be deleted from the system, or a customer requests data deletion as per GDPR regulation), but as these are not explicitly required by the tasks they have not been handled in the CREDO check. The benefit of including these could be argued, and future designs will take these considerations into account.

6 LoFi Prototype

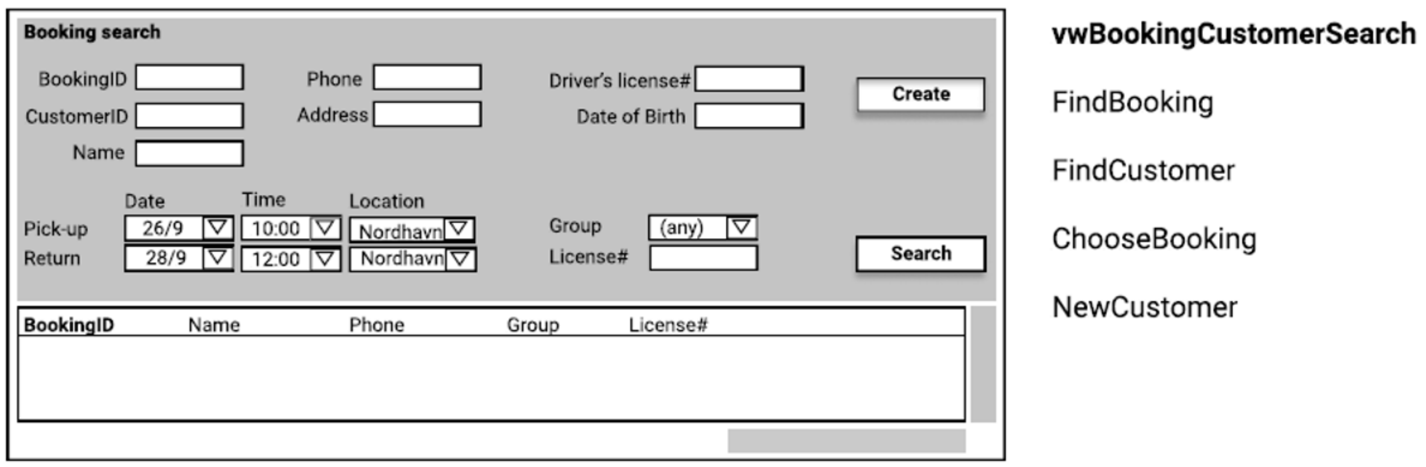
The development of the LoFi prototype functions thusly:

- Combining virtual windows into screens
- Identify buttons and functions needed on each screen

The task could be considered finished when each screen had enough buttons to facilitate all the required tasks. It is worth noting that elements like design style and navigation were not considered. The purpose of the LoFi prototype in this instance was to add functions and contextualize these within the virtual windows.

6.1 Initial LoFi prototype

During the initial design of the LoFi prototype we did not clearly communicate entry of certain data, nor did we contextualize these within screens. The initial LoFi prototype can be seen below.



Booking search

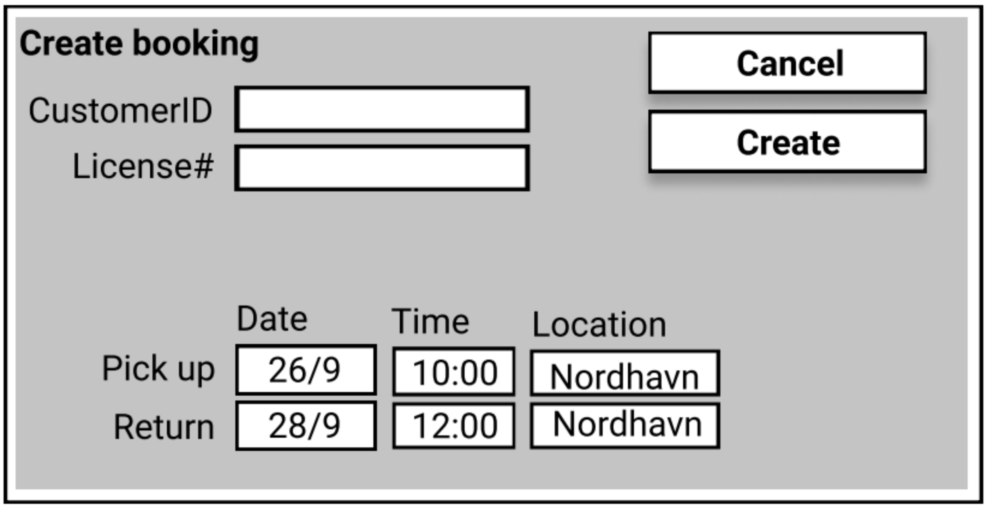
BookingID, CustomerID, Name, Phone, Address, Driver's license#, Date of Birth

	Date	Time	Location
Pick-up	26/9	10:00	Nordhavn
Return	28/9	12:00	Nordhavn

Group (any)
License#

BookingID	Name	Phone	Group	License#

Figure 20: Booking search VW with functions

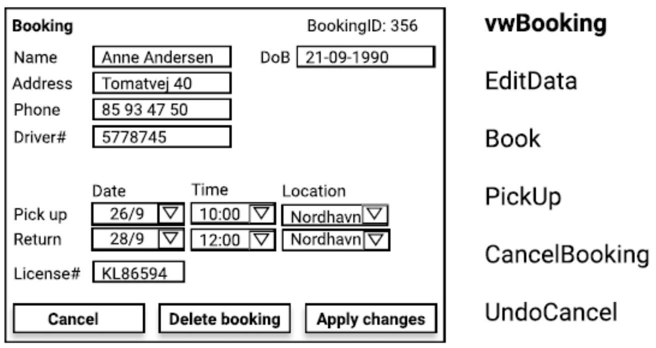


Create booking

CustomerID
License#

	Date	Time	Location
Pick up	26/9	10:00	Nordhavn
Return	28/9	12:00	Nordhavn

Figure 21: CREDO check over tasks



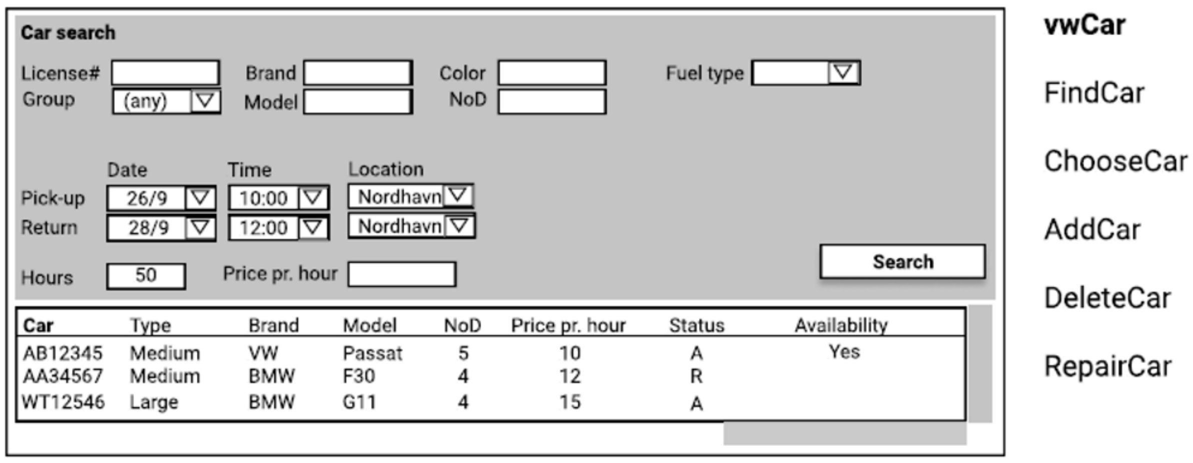
Booking BookingID: 356

Name	Anne Andersen	DoB	21-09-1990
Address	Tomatvej 40		
Phone	85 93 47 50		
Driver#	5778745		

	Date	Time	Location
Pick up	26/9	10:00	Nordhavn
Return	28/9	12:00	Nordhavn

License# KL86594

Figure 22: CREDO check over tasks



Car search

License#		Brand		Color		Fuel type	
Group	(any)	Model		NoD			

	Date	Time	Location
Pick-up	26/9	10:00	Nordhavn
Return	28/9	12:00	Nordhavn

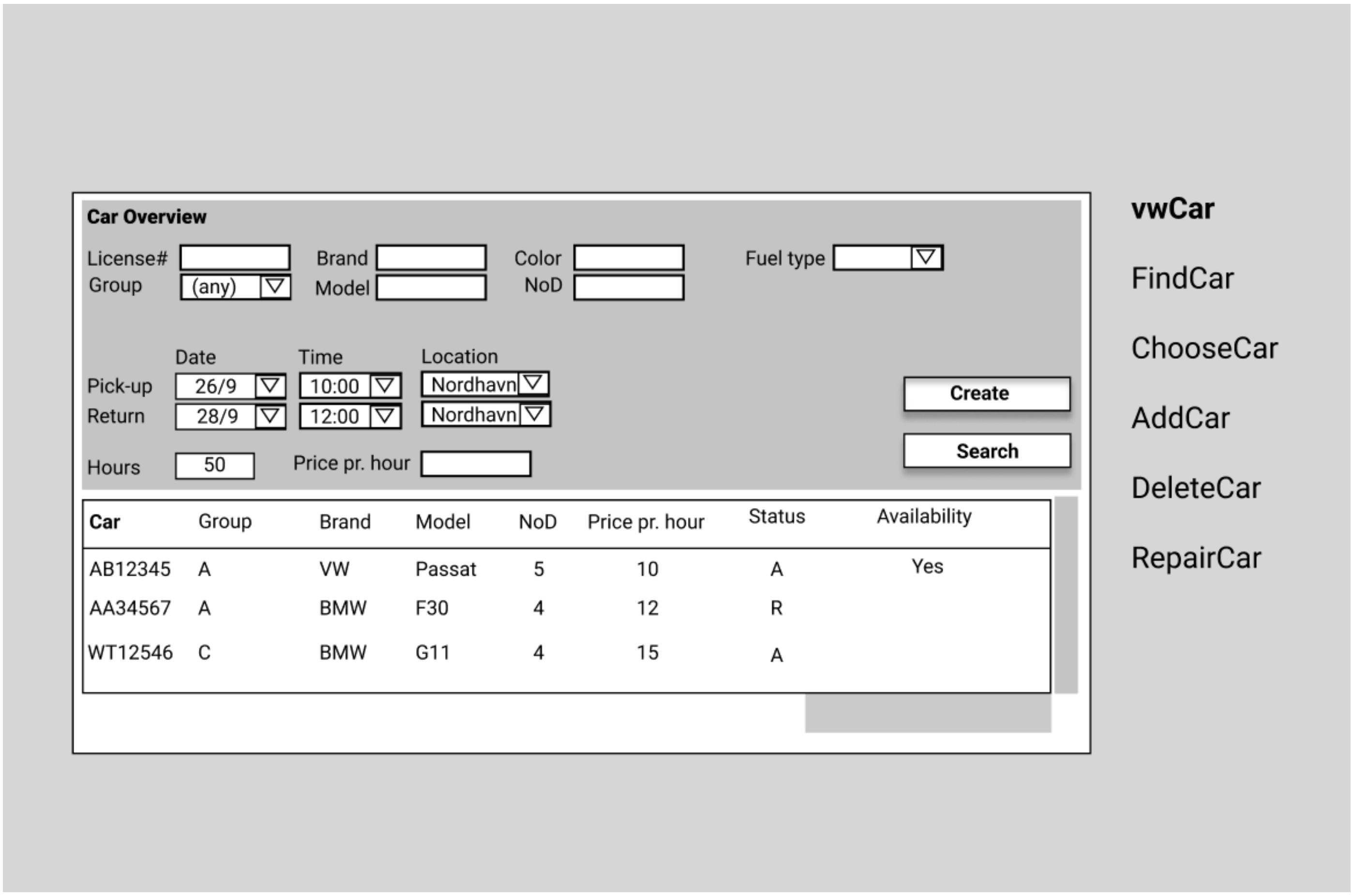
Hours 50 Price pr. hour

Car	Type	Brand	Model	NoD	Price pr. hour	Status	Availability
AB12345	Medium	VW	Passat	5	10	A	Yes
AA34567	Medium	BMW	F30	4	12	R	
WT12546	Large	BMW	G11	4	15	A	

Figure 23: CREDO check over tasks

6.2 Updated LoFi prototype

As part of the further developments, I chose to update the LoFi prototype to better fit the theory and the intended design without making massive changes. This includes contextualizing the existing LoFi prototype such that the virtual windows are represented in screens, and adding missing data inputs. These serve as better illustrations though it is worth noting that given the late update, they did not have an impact on the design process.



Car Overview

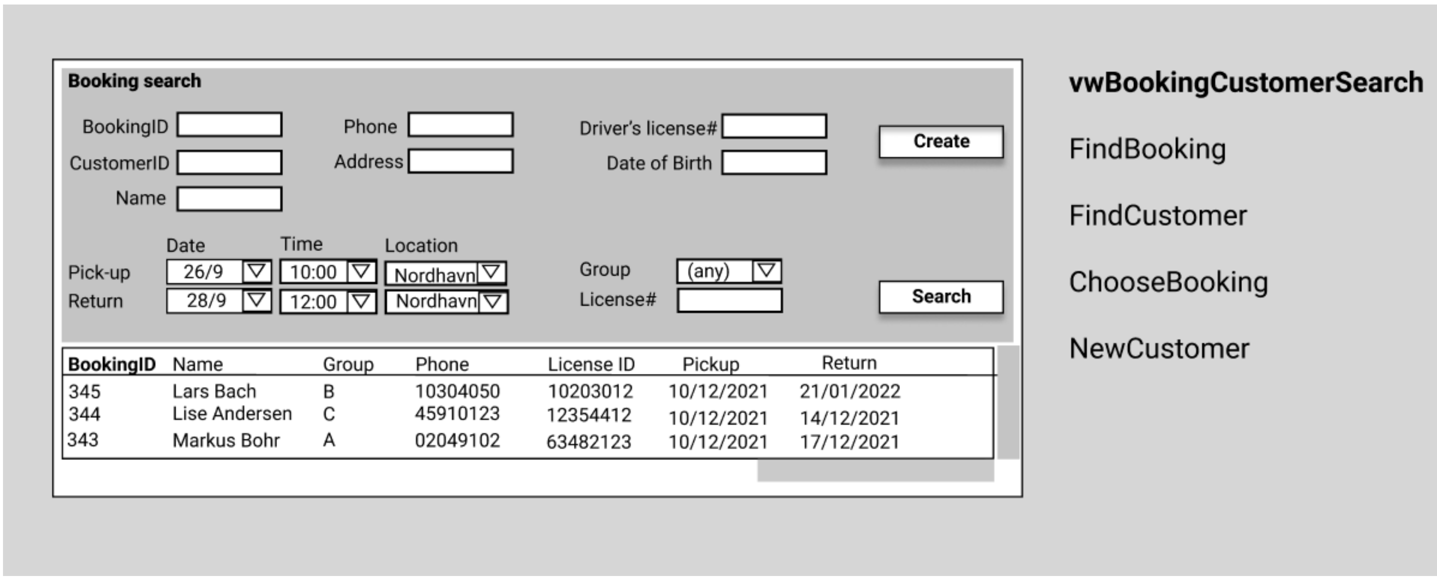
License#		Brand		Color		Fuel type	
Group	(any)	Model		NoD			

	Date	Time	Location
Pick-up	26/9	10:00	Nordhavn
Return	28/9	12:00	Nordhavn

Hours 50 Price pr. hour

Car	Group	Brand	Model	NoD	Price pr. hour	Status	Availability
AB12345	A	VW	Passat	5	10	A	Yes
AA34567	A	BMW	F30	4	12	R	
WT12546	C	BMW	G11	4	15	A	

Figure 24: Car overview page



Booking search

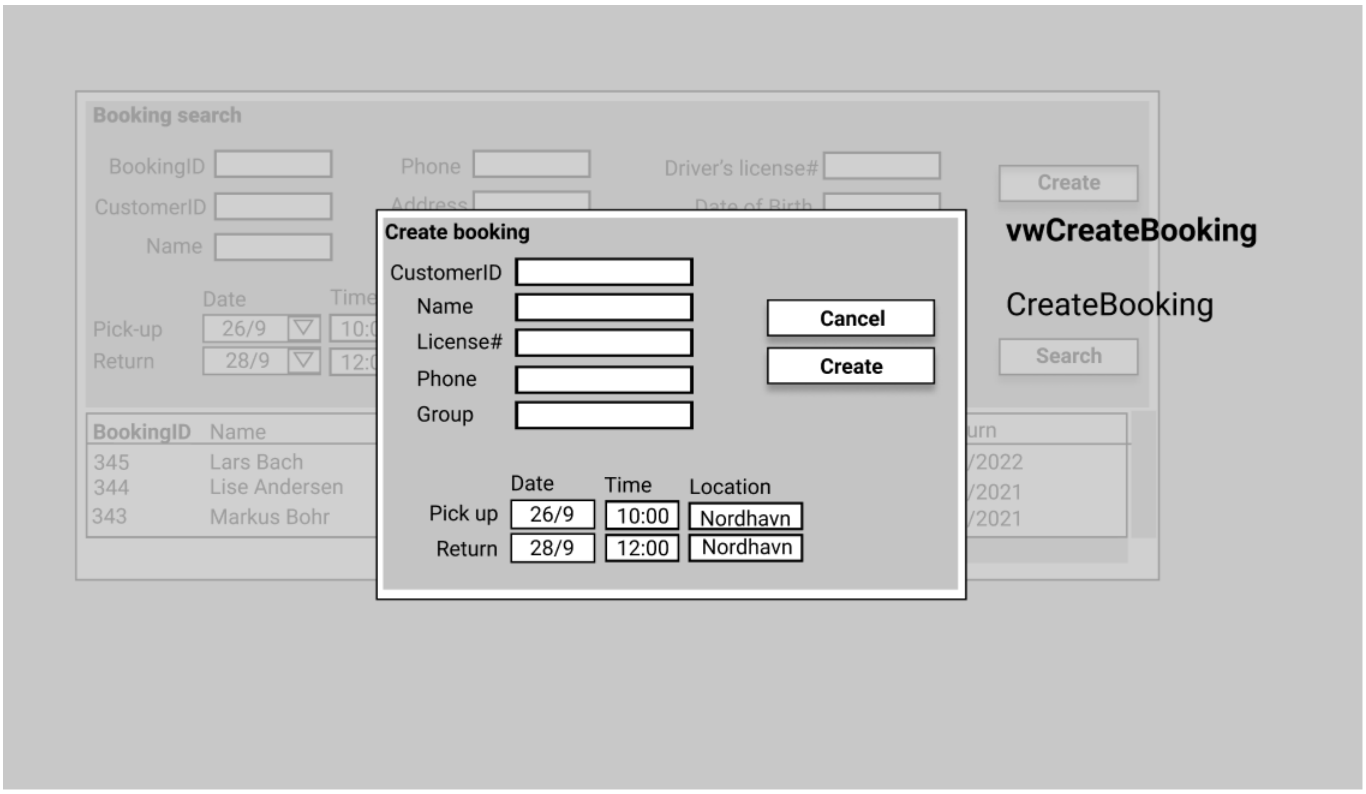
BookingID		Phone		Driver's license#	
CustomerID		Address		Date of Birth	
Name					

	Date	Time	Location
Pick-up	26/9	10:00	Nordhavn
Return	28/9	12:00	Nordhavn

Group (any)
License#

BookingID	Name	Group	Phone	License ID	Pickup	Return
345	Lars Bach	B	10304050	10203012	10/12/2021	21/01/2022
344	Lise Andersen	C	45910123	12354412	10/12/2021	14/12/2021
343	Markus Bohr	A	02049102	63482123	10/12/2021	17/12/2021

Figure 25: Booking search page



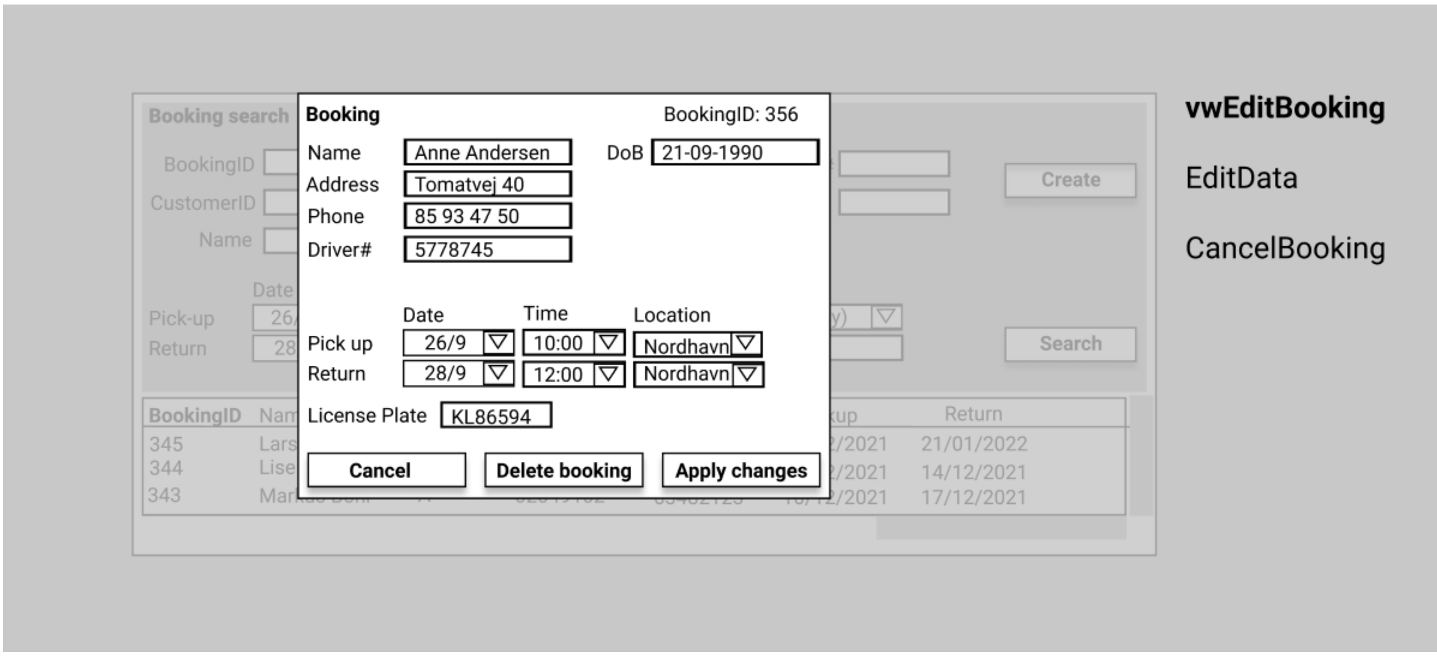
Create booking

CustomerID			
Name			
License#			
Phone			
Group			

	Date	Time	Location
Pick up	26/9	10:00	Nordhavn
Return	28/9	12:00	Nordhavn

BookingID	Name
345	Lars Bach
344	Lise Andersen
343	Markus Bohr

Figure 26: Create booking window pop-up overlay



Booking BookingID: 356

Name	Anne Andersen
DoB	21-09-1990
Address	Tomatvej 40
Phone	85 93 47 50
Driver#	5778745

	Date	Time	Location
Pick up	26/9	10:00	Nordhavn
Return	28/9	12:00	Nordhavn

License Plate KL86594

Figure 27: Edit booking window pop-up overlay

7 Usability tests

Usability tests are a frequently used tool of designers to identify usability problems [1]. Usability testing as a framework spans multiple different techniques and can be conducted at several different stages of the design process. Each technique carries its own benefits and downsides, and it is important the facilitators of the test account for these when planning, as well as conduct the test in an appropriate manner.

The course of our project was marred by usability testing issues. The first usability test was conducted in a manner that its results were limited and it was generally deemed unusable. This necessitated a second usability test. Despite failing the initial usability test, it will be described as it both is a part of and impacted the design process.

7.1 First usability test

The first usability test was conducted based on the initial HiFi prototype. The section describing it can be found **Here**.

This test was marred by two significant issues. Using the categorization of testing errors laid out by Lauesen (2005)[1], these errors can be categorized as follows:

- Missing functionality: The prototype showed an Offices section yet its screens were not implemented in any way.
- Set-up error: The prototype had a severe lack of interact-ability. This had the negative effect of funneling the user into paths that we expected them to take (and therefore programmed), which lessened the potential discovery of unaccounted usability issues.

Beyond Lauesen's categories, the usability test also suffered from poorly defined tasks that informed the user of what to do rather than letting them figure out a solution to a problem using the system.

The usability test did highlight lacking subjective satisfaction for the respondent. Namely, they noted the overall color scheme as being very "boring", with gray on gray colors being very prominent.

7.2 Second usability test

The second usability test was conducted based on the final HiFi prototype. The section describing it can be found **here**.

Numerous steps were taken to avoid the errors of the first usability test. These steps will be described now.

7.2.1 Task definitions

The tasks were noted as a serious weakness of the first usability test. To avoid the same issue, we changed the description of the tasks such that they reflect possible real-life scenarios that a user (in this case a receptionist) might encounter without prompting them whatsoever. The formal task definitions can be viewed in figure 28. These tasks were not given to the participant on text, but rather acted out by the facilitator. This will be described in later sections.

Task 1: Louise Simpson calls the office and requests to book a car. She wants the largest type of car available. She wants to pick it up at the Amager office on the 12th of December 2021 10:00 and return it to the Nordhavn office on the 17th of December 16:00.

Her address is 123 Tornerosevej, her phone number is +45 4100 9010, license ID is 00104910

Task 2: Martin Liston calls the office and says he made a booking for a week with pick-up some time in January 2021, but due to a change of plans he would like to push pickup back to February 15th 2021.

If asked: He does not want to change car-group if there is no available car in the group he originally requested. He also does not want to budge on the new date. If neither are possible, he prefers to have the booking cancelled entirely.

Task 3: Jonathan Skinner has arrived at the Amager office and is ready to pick up his booked car [task completion is defined as updating the booking as having a picked up car – no exchange of keys].

Task 4: Your experienced rental manager goes on holiday and asks you to cover for him. Before leaving he said that the office should at least have 25% of its capacity of cars available at a given time to account for walk-ins. If this isn't the case, you must get cars from other offices. One day the rental manager calls and to hear about the current status of the fleet.

Figure 28: Usability test tasks

7.2.2 The user

Principally, the ideal participant for a usability test is a person who represents the intended user group of the product. Usually, such as the case for ScanCar contracting designers to make a new system, the customer will often supply their own employees as their expertise, familiarity, and coming use of the software makes them ideal participants.

Due to our resource and time constraints, it was not feasible to find someone who fit the profile of a ScanCar receptionist. Instead, a mid 20s female student of History at the University of Copenhagen was chosen instead. While we had a large supply of familial relations at the IT University who could help, it was deemed beneficial to use a participant who is not familiar with IT systems as they might discover usability issues we had not anticipated. However, it is undeniably an issue that the participant was not related to the domain of reception or automotives.

The participant was given a reworded version of the problem statement (less verbose so as not to overwhelm them), and they were primed on the scenario and their role as receptionist in this. This is not an ideal method as it can be difficult for a person to both imagine themselves in a profession entirely unknown to them while critically examining their experience with a software designed for people within that profession.

7.2.3 Conducting the test

The structure of the test was as follows:

1. The user was given a description of the case and their role
2. The user was presented the application as their primary tool for the job
3. The user was "set to work", tasks were presented dynamically by the presenter through narration and role-play
4. After all tasks had been completed, a brief post-test interview was conducted

Present for the test were the participant, a facilitator, and a note-taker. The facilitator's primary role was to engage the participant in the test, introduce them to the concepts, narrate and role-play the participant's "day in the job", and conduct the exit interview. The note-taker took notes, relieving the facilitator from the pressure so they could maintain a more natural presence and dialogue with the participant.

7.2.4 Findings

The problems highlighted by the task are listed below, using Lauesen's categorization:

- Annoying – Task 4: The user found that only 20% of the cars were unavailable and they needed to be at 25% (per task description). The rental manager (interviewer) asked them to keep a somewhat even distribution of cars in the different groups. The participant noted that while there was an overview of cars available there was no deeper insight in which car groups these were in. The user had to navigate to the Cars on Premises page and manually study the parking lots.
- Annoying – Task 4: The user noted difficulty in determining the actual values of the charts in management.

- Missing Functionality – Task 3: There is no way to update the car's statuses on the premises page.
- Annoying – Task 4: The user noted annoyance at the carGroup request buttons. They did not allow text inputs and could not be reset, meaning each erroneous click would have to be undone with another one.
- Set-up error – Task 1: Due to the prototype's limitations the prototype included information in a new booking that the user had not created (customer license)

Based on the response of annoyance, the stacked bar chart had been changed to instead represent the number of available cars for each car groups on the y-axis, with the x-axis still representing dates over the coming week. With this change, users should hopefully be able to get a quick overview of the estimated availability of each car group for the coming week.

After the second usability test, we decided to make some changes to the booking overview table, which we implemented in our implementation. We changed the edit button on each booking in the table and added a delete button.

We realized that our Hi-Fi prototype did not allow the user to change the status of the booking (a function under edit booking, which would also affect the CarState), so we added that during implementation.

8 HiFi Prototype

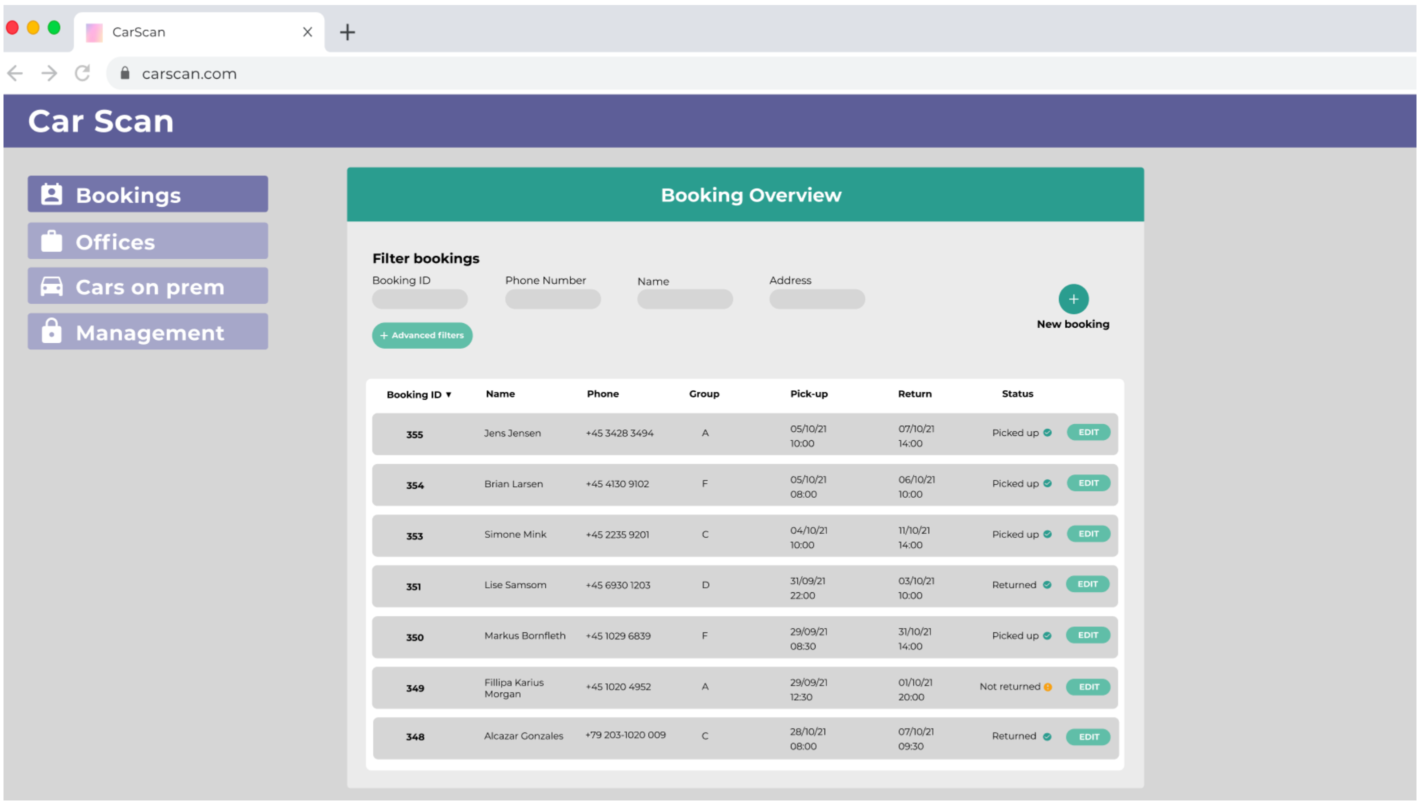
This section will show the HiFi prototype along with the primary design considerations behind it.
Based on the previously presented work, a HiFi prototype was to be developed. As this design was to be used in a professional context, the primary question we sought to answer with the design was *"How can the design help the users complete their tasks most efficiently?"* This approach led to what can be perceived as a utilitarian design where data, and the overview and manipulation hereof, was in focus.

As part of a narrowing of the scope, viewing and editing customer-specific information was limited for the HiFi prototype.

The LoFi prototype was also found to be insufficient as a guide for the entire system, operating more as functional virtual windows. For this reason, the following HiFi prototypes have more features and considerations than the LoFi prototype does. This, of note, includes improved car overviews and car transfer implementation.

8.1 HiFi prototype, initial iteration

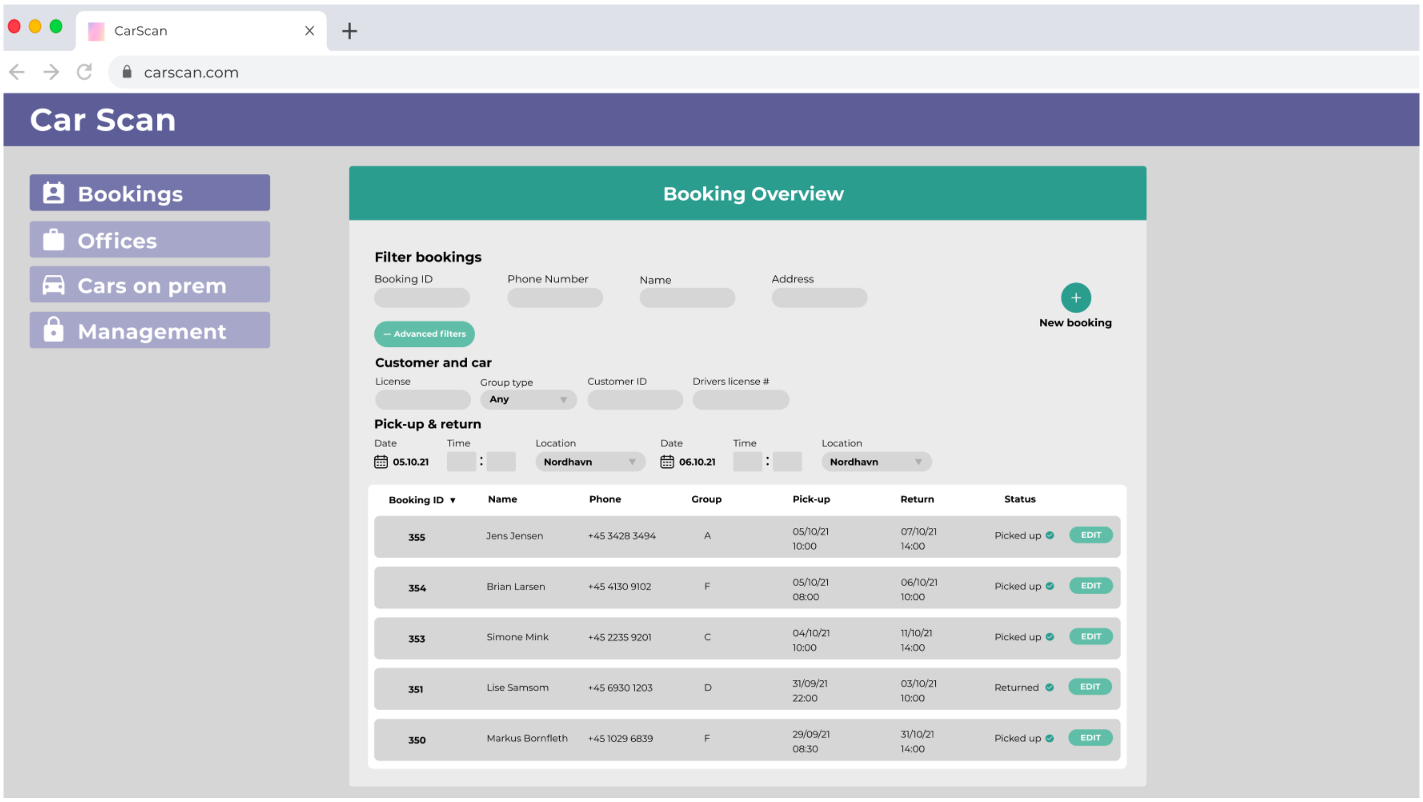
The first iteration of the design was done prior to any usability tests. This can be seen in the following figures, but explanations behind it will not be deepened as the implementation follows the second iteration.[3]



Booking ID ▾	Name	Phone	Group	Pick-up	Return	Status	
355	Jens Jensen	+45 3428 3494	A	05/10/21 10:00	07/10/21 14:00	Picked up ✓	EDIT
354	Brian Larsen	+45 4130 9102	F	05/10/21 08:00	06/10/21 10:00	Picked up ✓	EDIT
353	Simone Mink	+45 2235 9201	C	04/10/21 10:00	11/10/21 14:00	Picked up ✓	EDIT
351	Lise Samsom	+45 6930 1203	D	31/09/21 22:00	03/10/21 10:00	Returned ✓	EDIT
350	Markus Bornfleth	+45 1029 6839	F	29/09/21 08:30	31/10/21 14:00	Picked up ✓	EDIT
349	Fillipa Karius Morgan	+45 1020 4952	A	29/09/21 12:30	01/10/21 20:00	Not returned ●	EDIT
348	Alcazar Gonzales	+79 203-1020 009	C	28/10/21 08:00	07/10/21 09:30	Returned ✓	EDIT

Figure 29: Booking Overview / landing page. Relevant for tasks related to creation of bookings and manipulation thereof

[3]High resolution version of the images can be found in the appendix sub-folder "HiFi Iteration 1"



Filter bookings

Booking ID | Phone Number | Name | Address

Customer and car

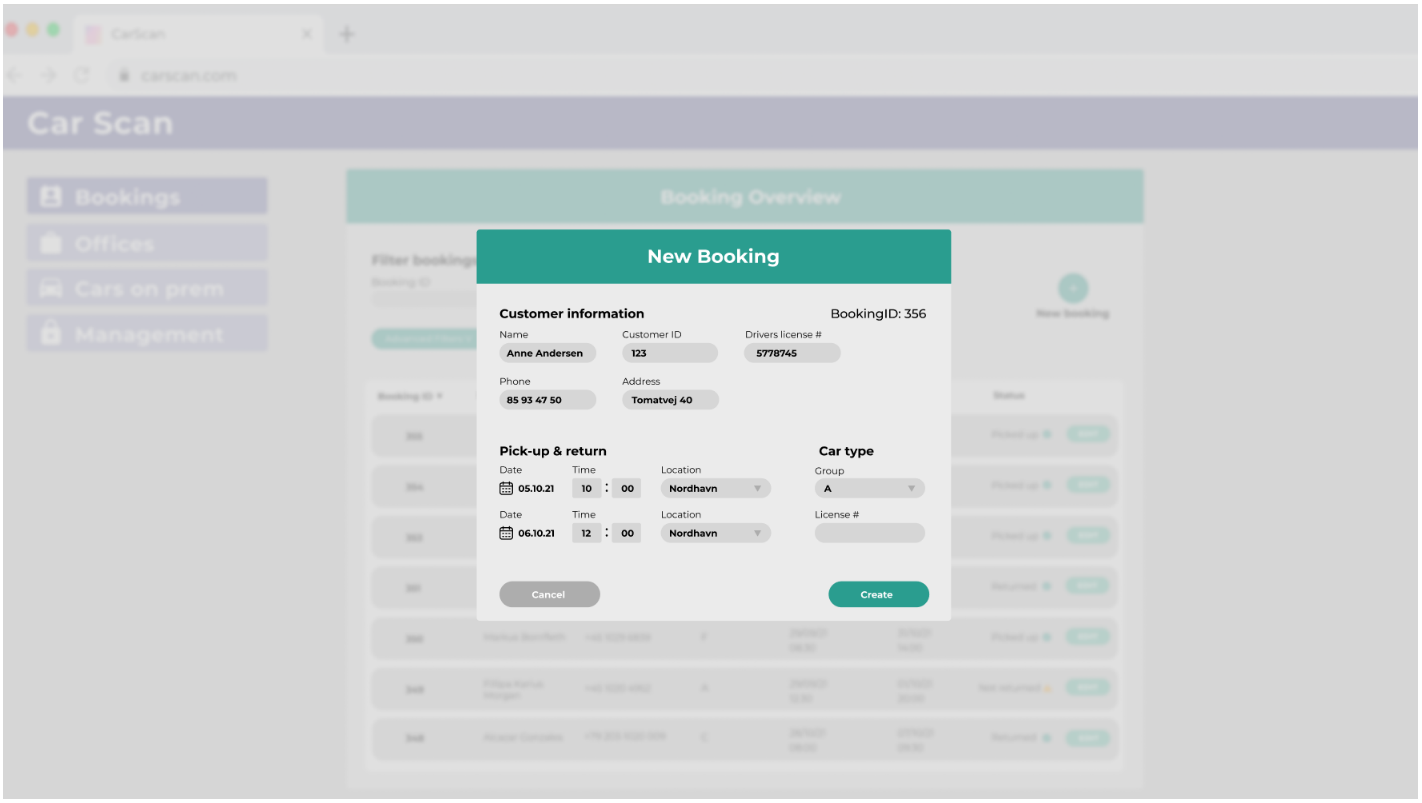
License | Group type: Any | Customer ID | Drivers license #

Pick-up & return

Date: 05.10.21 | Time | Location: Nordhavn | Date: 06.10.21 | Time | Location: Nordhavn

Booking ID	Name	Phone	Group	Pick-up	Return	Status
355	Jens Jensen	+45 3428 3494	A	05/10/21 10:00	07/10/21 14:00	Picked up ✓
354	Brian Larsen	+45 4130 9102	F	05/10/21 08:00	06/10/21 10:00	Picked up ✓
353	Simone Mink	+45 2235 9201	C	04/10/21 10:00	11/10/21 14:00	Picked up ✓
351	Lise Samsom	+45 6930 1203	D	31/09/21 22:00	03/10/21 10:00	Returned ✓
350	Markus Bornfleth	+45 1029 6839	F	29/09/21 08:30	31/10/21 14:00	Picked up ✓

Figure 30: Booking overview with advanced filters



Customer information BookingID: 356

Name: Anne Andersen | Customer ID: 123 | Drivers license #: 5778745

Phone: 85 93 47 50 | Address: Tomatvej 40

Pick-up & return

Date: 05.10.21 | Time: 10 : 00 | Location: Nordhavn

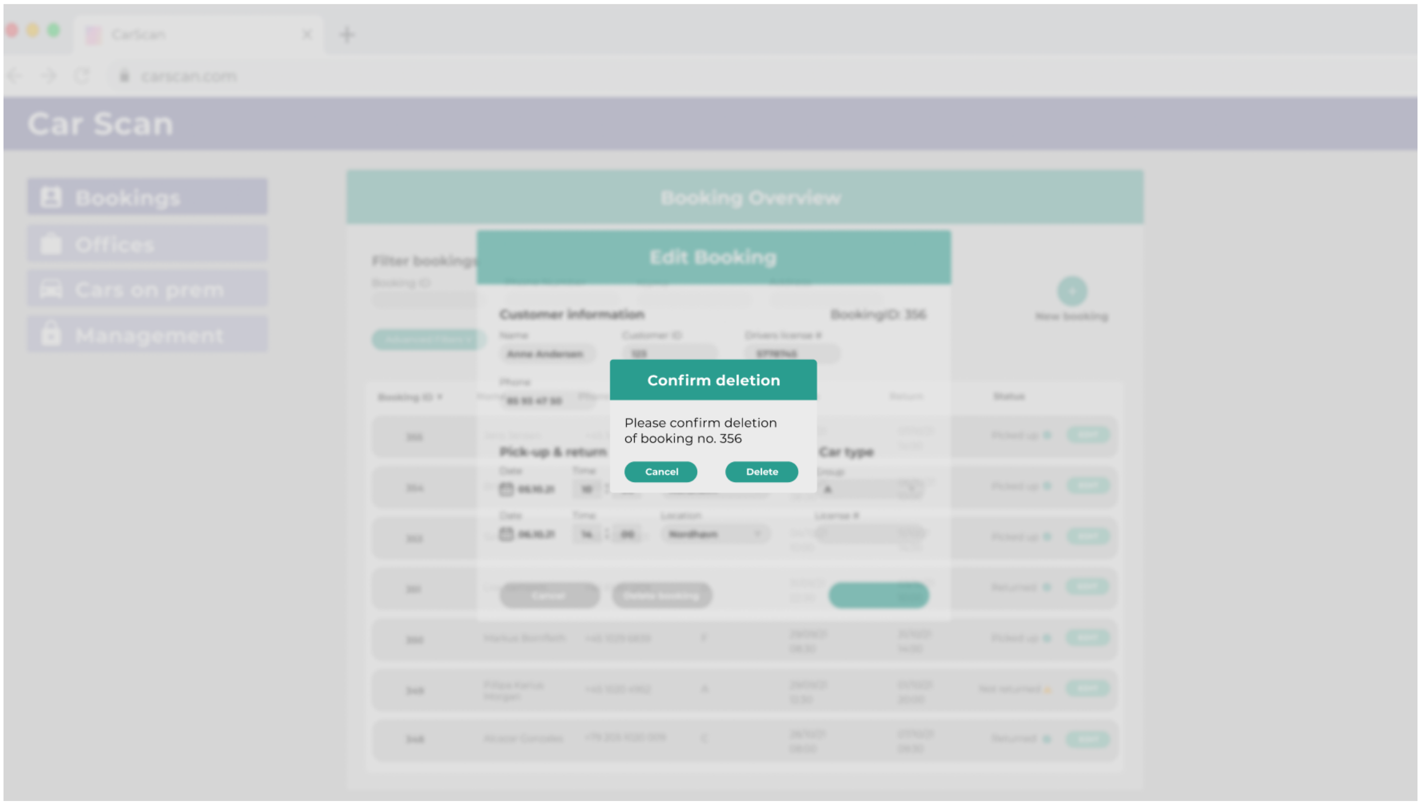
Date: 06.10.21 | Time: 12 : 00 | Location: Nordhavn

Car type

Group: A

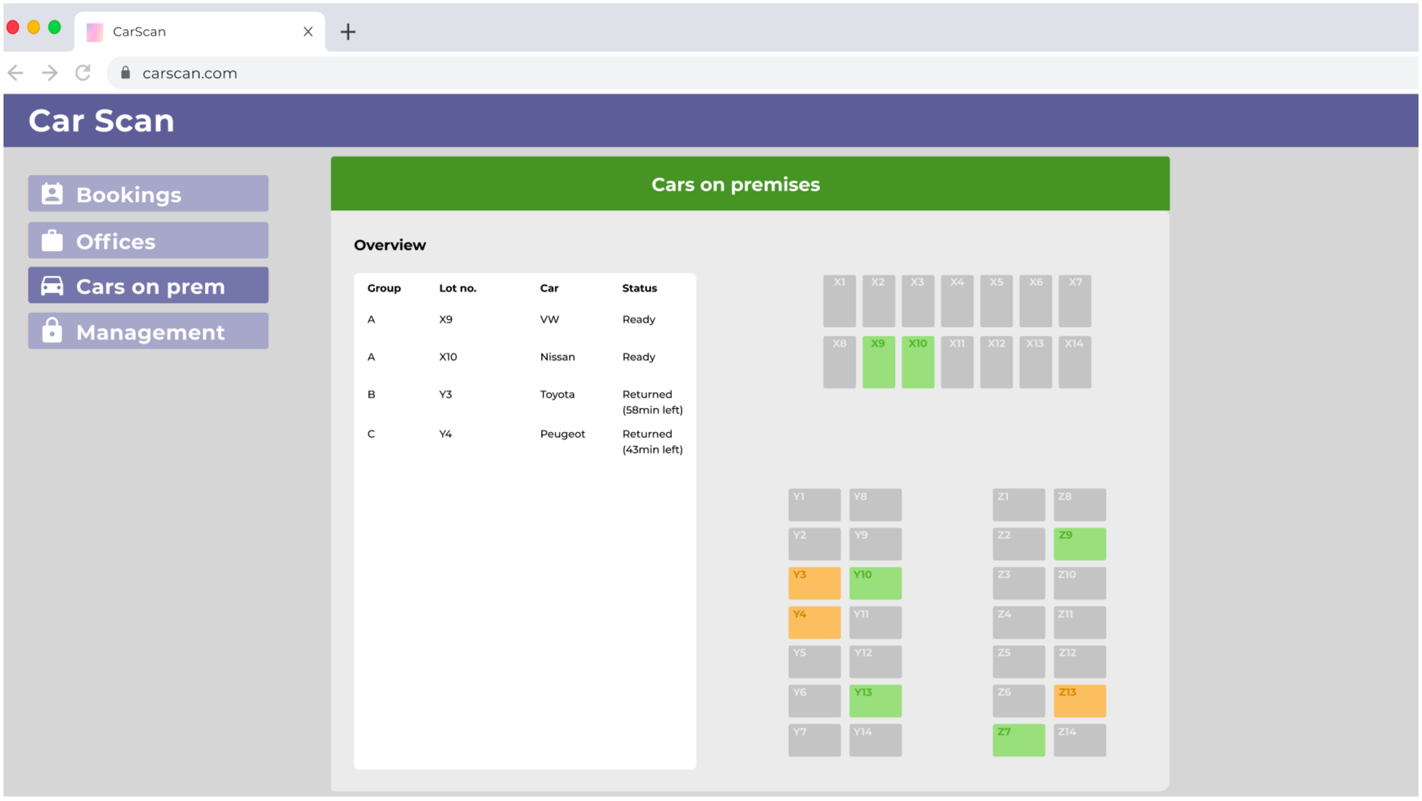
License #

Figure 31: New booking screen, related to tasks of creating new bookings per customer request



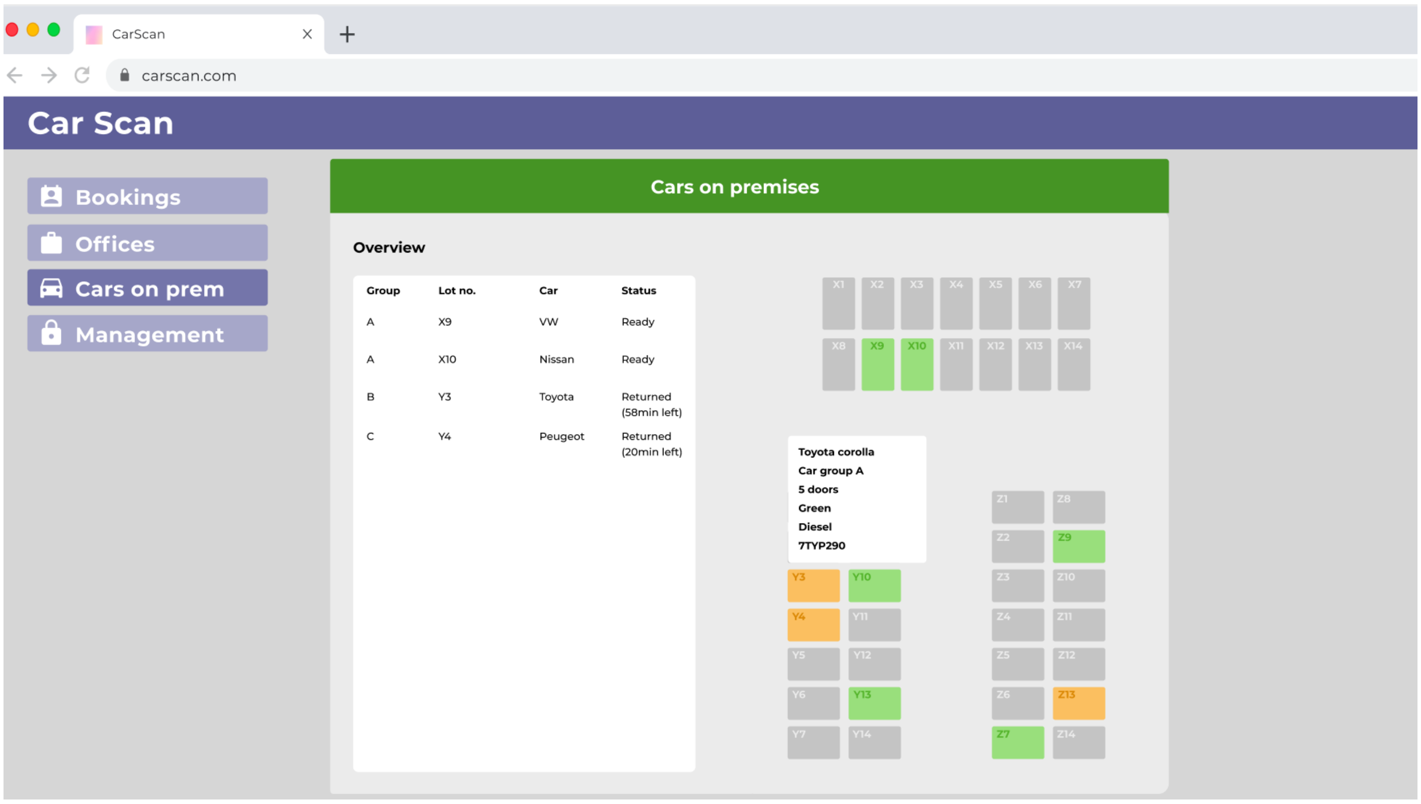
Please confirm deletion of booking no. 356

Figure 32: Delete booking confirmation, related to tasks of cancellations of bookings



Group	Lot no.	Car	Status
A	X9	VW	Ready
A	X10	Nissan	Ready
B	Y3	Toyota	Returned (58min left)
C	Y4	Peugeot	Returned (43min left)

Figure 33: Cars on premises overview related to pickup, returns, and manipulation of cars. The blocks on the right represent lot spaces on a parking lot and would vary between offices.

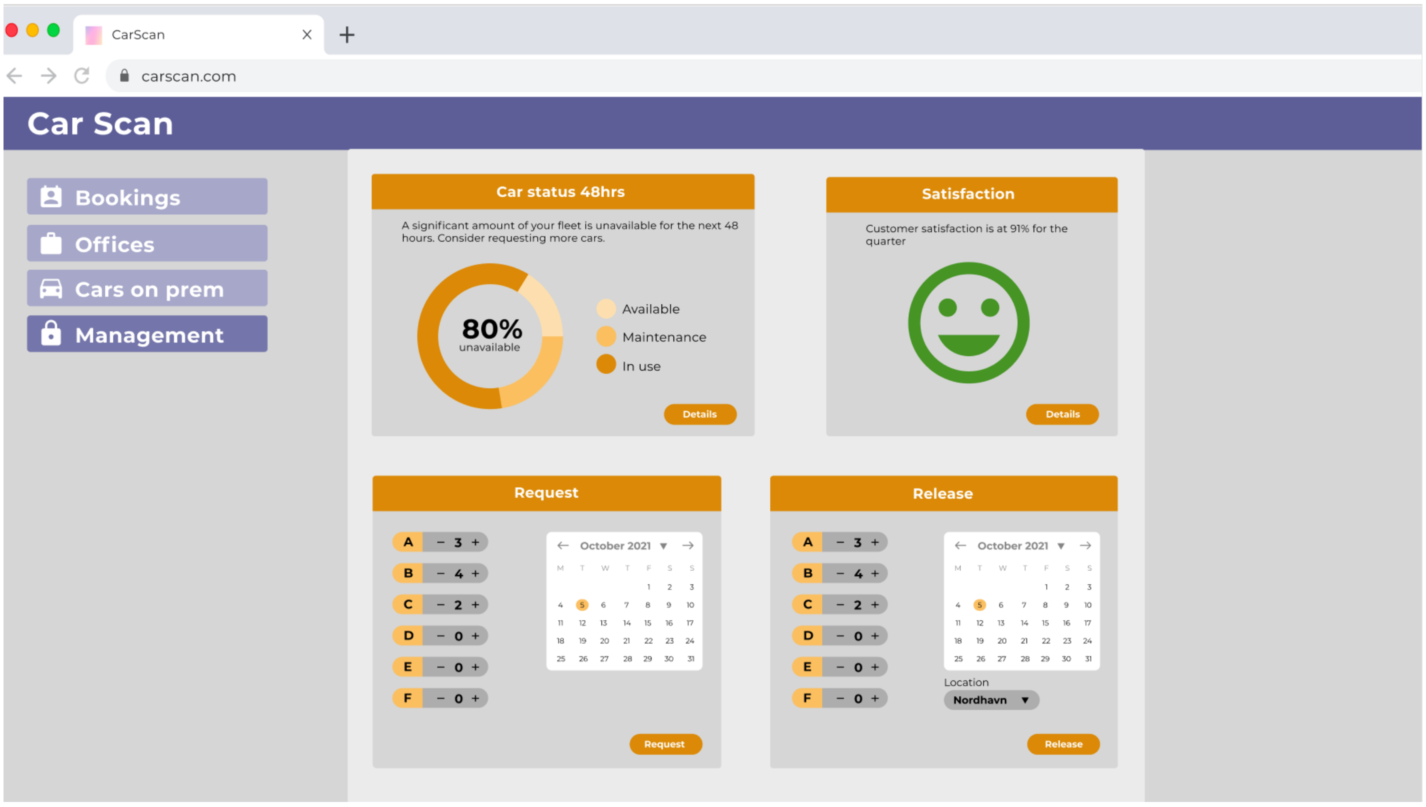

CarScan
carscan.com
Car Scan
Bookings
Offices
Cars on prem
Management
Cars on premises
X1 X2 X3 X4 X5 X6 X7
X8 X9 X10 X11 X12 X13 X14
Y3 Y4 Y5 Y6 Y7 Y10 Y11 Y12 Y13 Y14
Z1 Z2 Z3 Z4 Z5 Z6 Z7 Z8 Z9 Z10 Z11 Z12 Z13 Z14

Overview

Group	Lot no.	Car	Status
A	X9	VW	Ready
A	X10	Nissan	Ready
B	Y3	Toyota	Returned (58min left)
C	Y4	Peugeot	Returned (20min left)

Toyota corolla
Car group A
5 doors
Green
Diesel
7TYP290

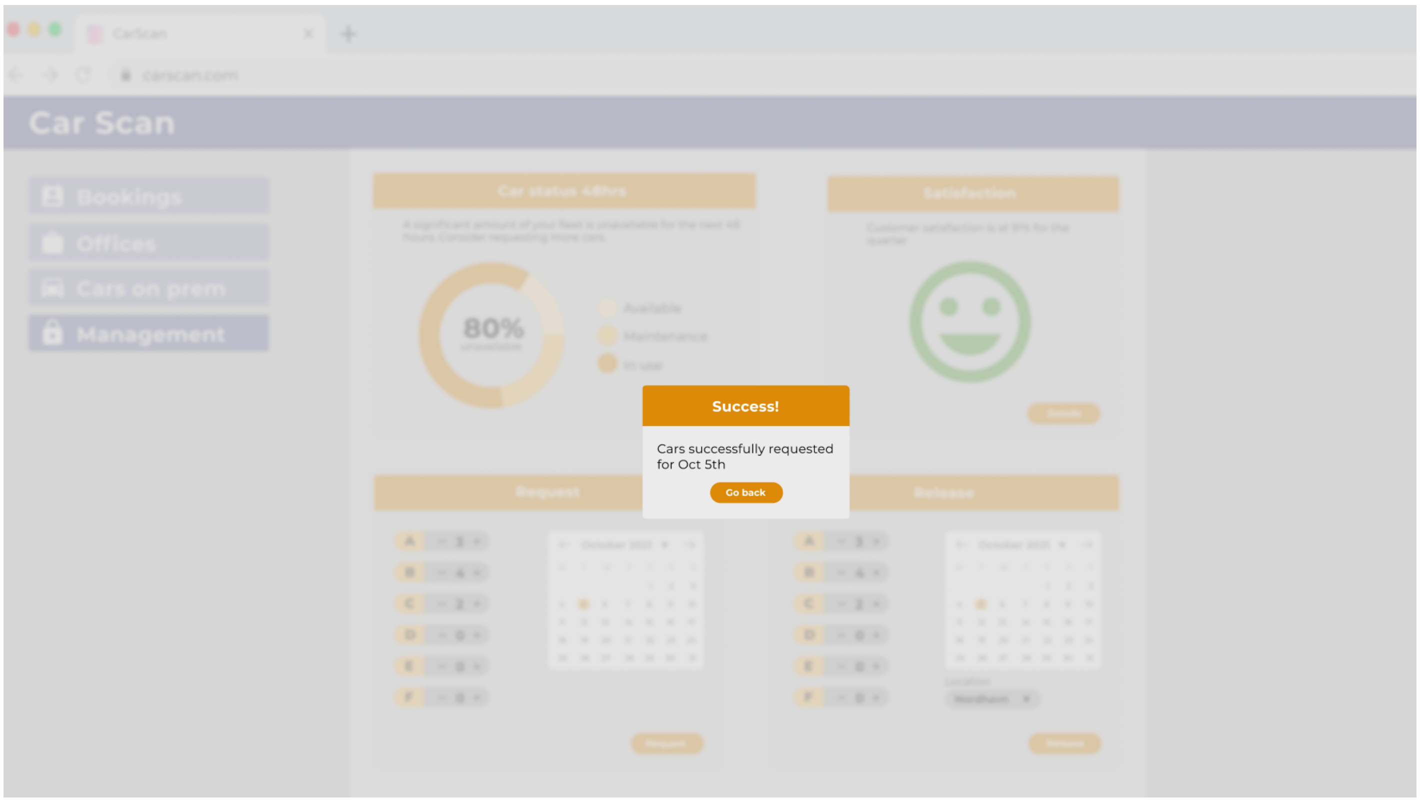
Figure 34: Cars on premises screen with pop-up on hover



A significant amount of your fleet is unavailable for the next 48 hours. Consider requesting more cars.

Customer satisfaction is at 91% for the quarter

Figure 35: Management overview screen related to tasks of tracking car transfers, requesting and releasing cars



Cars successfully requested for Oct 5th

Figure 36: Management release confirmation

8.2 HiFi prototype final iteration

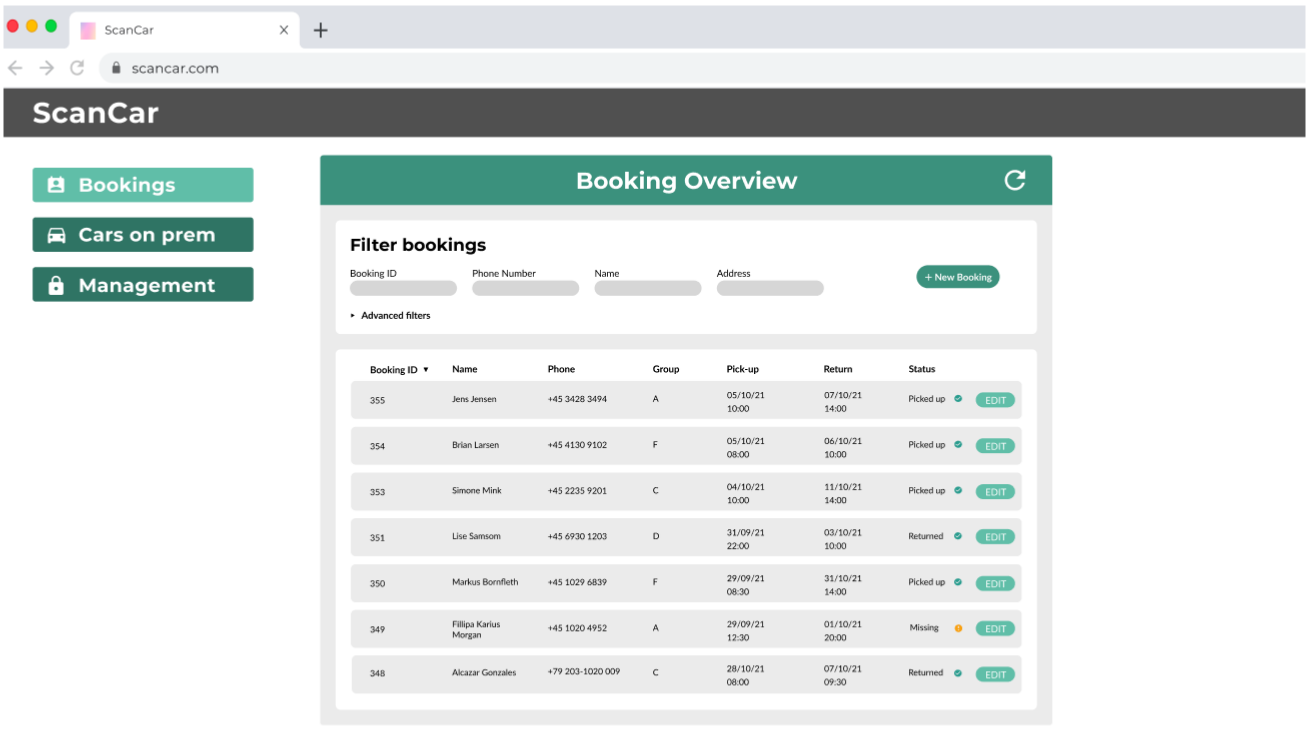
Based on both the first usability test and course feedback this design was largely overhauled. First, the design suffered from both boring and overuse of color. Every page was given its own accent color to theme them as distinct sections of the application, but this was found to be both overwhelming and confusing. Furthermore, every page had an abundance of grayscale colors, with most of the content being placed on gray blocks which itself were contained within a darker gray background. The design was marred by issues of consistency as well, including button designs and content spacing. However, the functional elements of the design, herein the booking overview and parking lot overview, were both deemed understandable and intuitive.

To rectify these, a design guide was created. The guide was made to be both as thorough as possible (without restraining creativity too much) and justifiable by rooting these decisions in literature. For the sake of brevity, only the most significant aspects of the guide will be described.

One of the more significant changes was the introduction of ”modules”. Modules, in this case, refers to related blocks of content that are contained within the same white background, which itself are placed on a gray ”canvas”. This achieves two things: First, through the Gestalt Law of Proximity, content relations are denoted by placement within the same module (and anything not contained within a module can be considered empty space). Second, it limits the boredom expressed by the UI by limiting the grayness. These modules may also have sub-modules and titles denoted by green labels.

Coloring also received an overhaul. The primary colors were changed to be shades of green. Secondary colors included shades of orange to communicate destructive actions (cancelling an action) or unavailability. The lightest shades of green were used to signify availability.

The management page was also changed quite a bit. The first iteration had trackers for customer satisfaction, but as these were not deemed to be relevant for the problem statement, they were dropped. Instead, a bar chart tracking the estimated availability of cars the following days was added (the data herein would be based on pickups and drop-offs).



Filter bookings

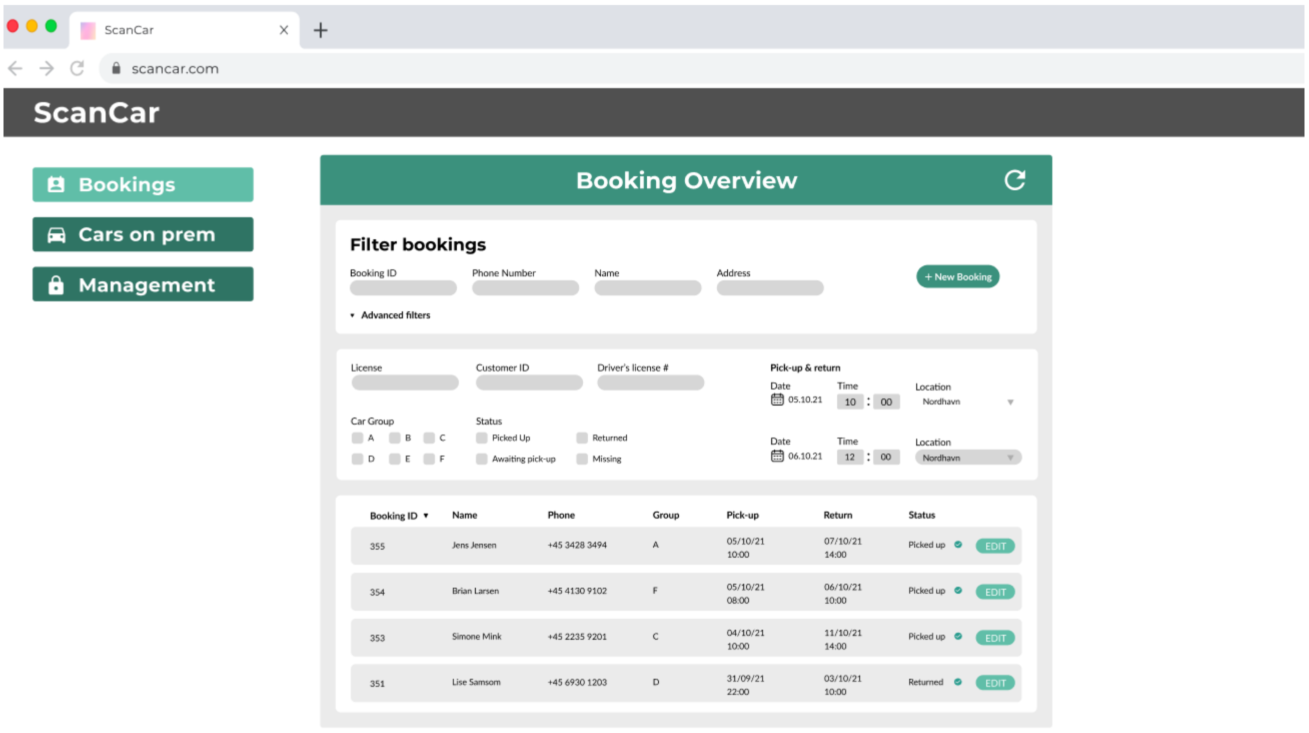
Booking ID · Phone Number · Name · Address

+ New Booking

▸ Advanced filters

Booking ID ▾	Name	Phone	Group	Pick-up	Return	Status	
355	Jens Jensen	+45 3428 3494	A	05/10/21 10:00	07/10/21 14:00	Picked up ●	EDIT
354	Brian Larsen	+45 4130 9102	F	05/10/21 08:00	06/10/21 10:00	Picked up ●	EDIT
353	Simone Mink	+45 2235 9201	C	04/10/21 10:00	11/10/21 14:00	Picked up ●	EDIT
351	Lise Samsom	+45 6930 1203	D	31/09/21 22:00	03/10/21 10:00	Returned ●	EDIT
350	Markus Bornfleth	+45 1029 6839	F	29/09/21 08:30	31/10/21 14:00	Picked up ●	EDIT
349	Fillipa Karius Morgan	+45 1020 4952	A	29/09/21 12:30	01/10/21 20:00	Missing ●	EDIT
348	Alcazar Gonzales	+79 203-1020 009	C	28/10/21 08:00	07/10/21 09:30	Returned ●	EDIT

Figure 37: Booking overview related to tasks of creation and manipulation of bookings



Filter bookings

Booking ID · Phone Number · Name · Address

+ New Booking

▾ Advanced filters

License · Customer ID · Driver's license #

Car Group: ☐ A ☐ B ☐ C ☐ D ☐ E ☐ F

Status: ☐ Picked Up ☐ Returned ☐ Awaiting pick-up ☐ Missing

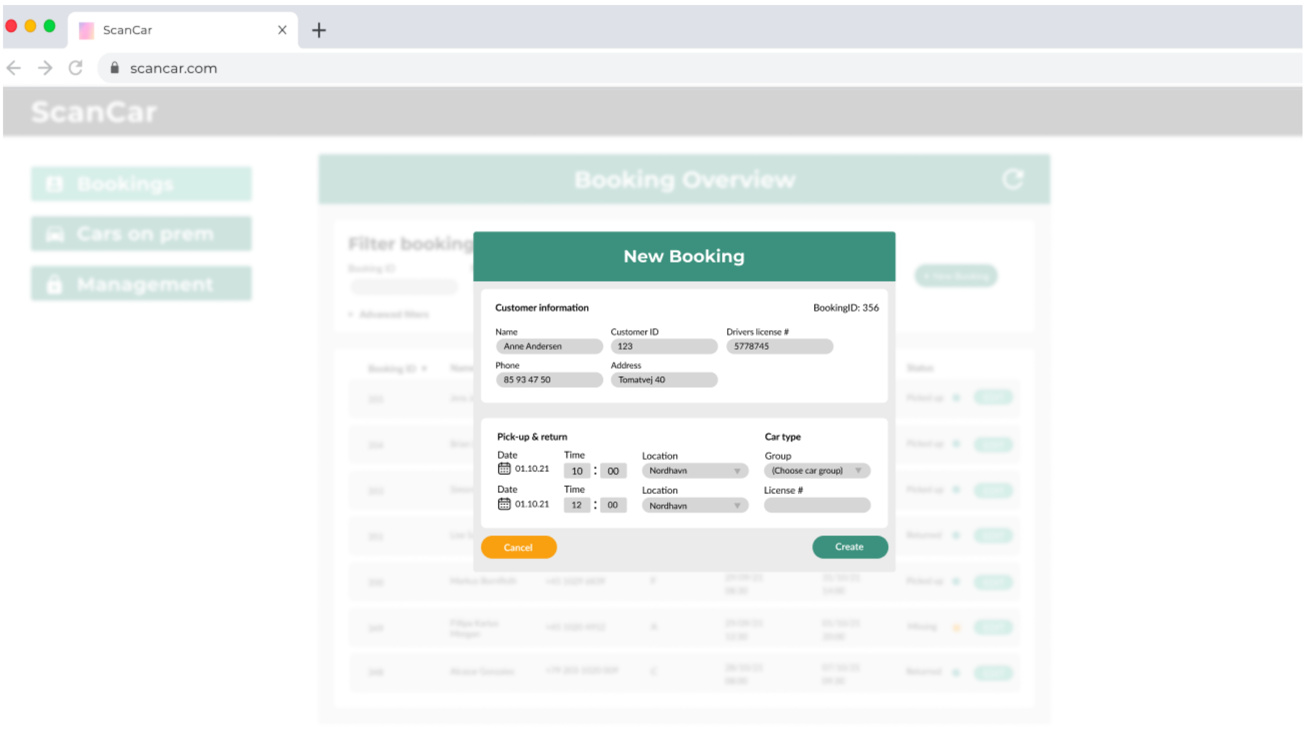
Pick-up & return

Date: 05.10.21 · Time: 10 : 00 · Location: Nordhavn

Date: 06.10.21 · Time: 12 : 00 · Location: Nordhavn

Booking ID ▾	Name	Phone	Group	Pick-up	Return	Status	
355	Jens Jensen	+45 3428 3494	A	05/10/21 10:00	07/10/21 14:00	Picked up ●	EDIT
354	Brian Larsen	+45 4130 9102	F	05/10/21 08:00	06/10/21 10:00	Picked up ●	EDIT
353	Simone Mink	+45 2235 9201	C	04/10/21 10:00	11/10/21 14:00	Picked up ●	EDIT
351	Lise Samsom	+45 6930 1203	D	31/09/21 22:00	03/10/21 10:00	Returned ●	EDIT

Figure 38: Overview with advanced booking filtering



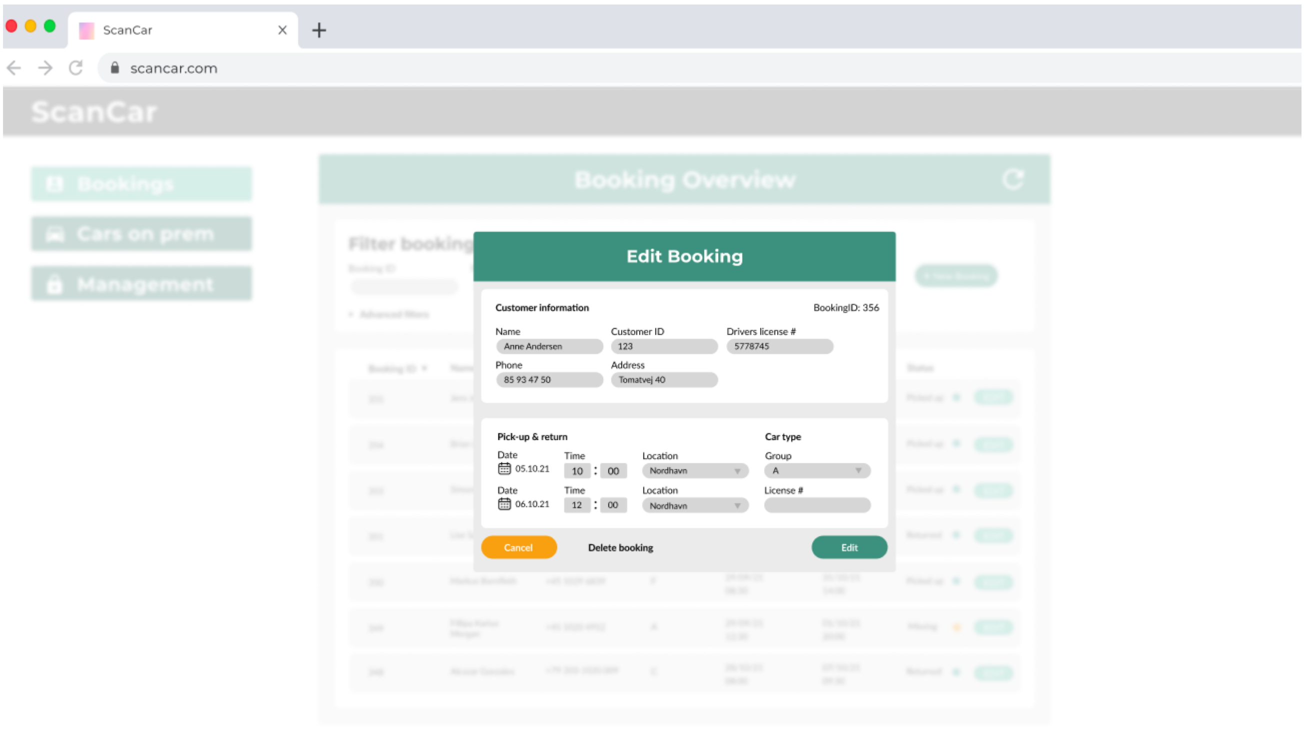
Customer information BookingID: 356

Name: Anne Andersen
Customer ID: 123
Drivers license #: 5778745
Phone: 85 93 47 50
Address: Tomatvej 40

Pick-up & return
Date: 01.10.21 Time: 10 : 00 Location: Nordhavn
Date: 01.10.21 Time: 12 : 00 Location: Nordhavn

Car type
Group: (Choose car group)
License #:

Figure 39: New booking modal related to creation of booking screen



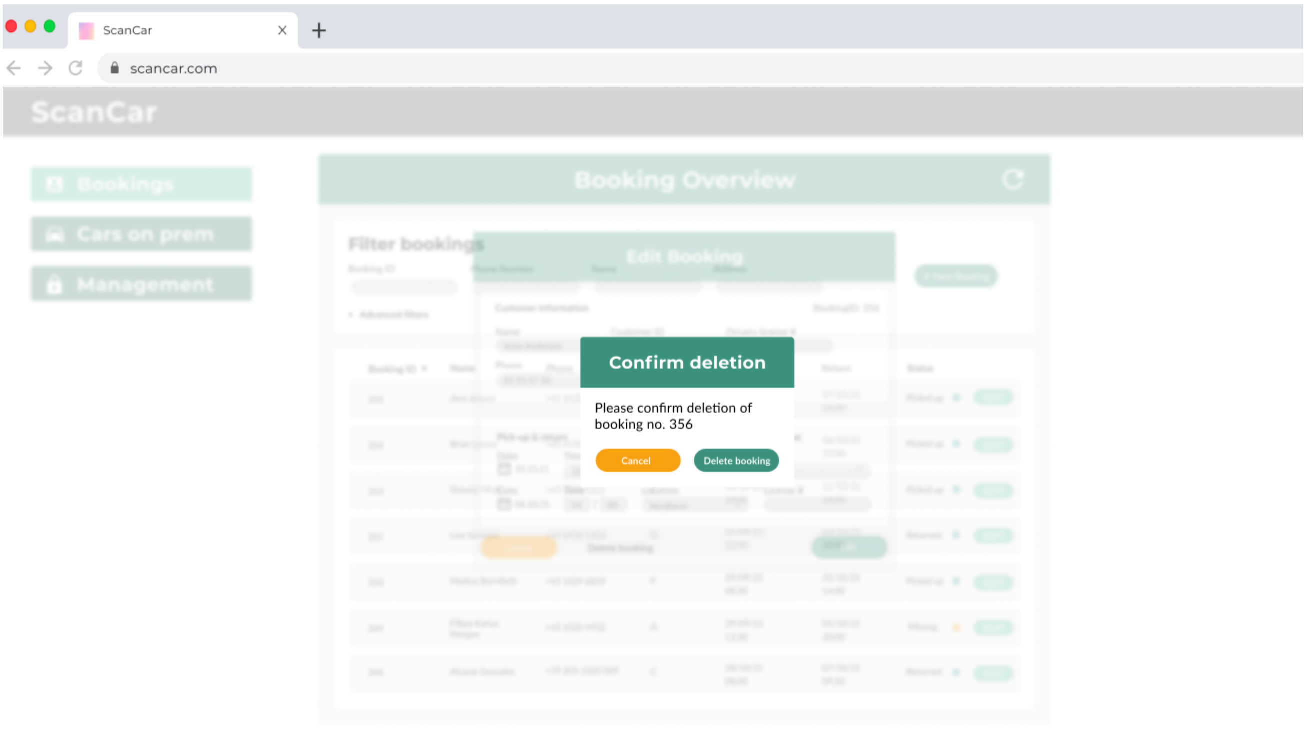
Customer information BookingID: 356

Name: Anne Andersen
Customer ID: 123
Drivers license #: 5778745
Phone: 85 93 47 50
Address: Tomatvej 40

Pick-up & return
Date: 05.10.21 Time: 10 : 00 Location: Nordhavn
Date: 06.10.21 Time: 12 : 00 Location: Nordhavn

Car type
Group: A
License #:

Figure 40: Edit booking modal

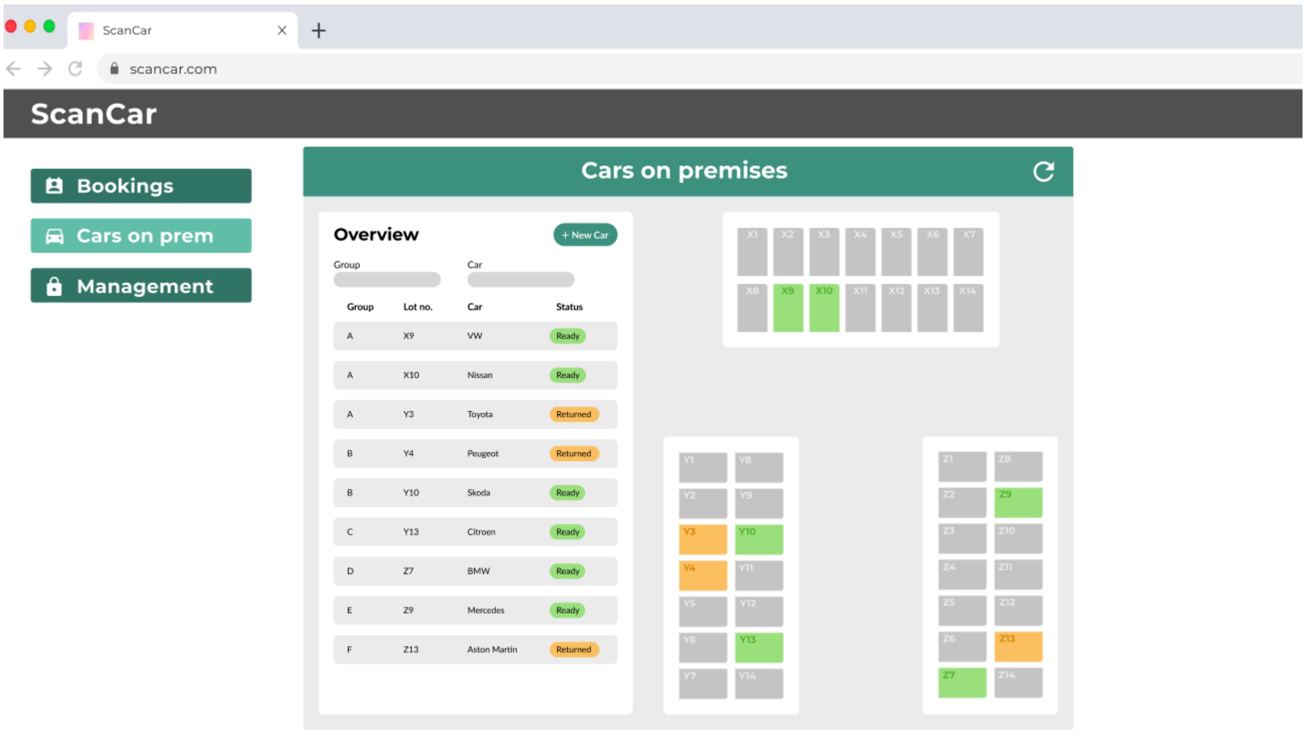


Confirm deletion

Please confirm deletion of booking no. 356

Cancel | Delete booking

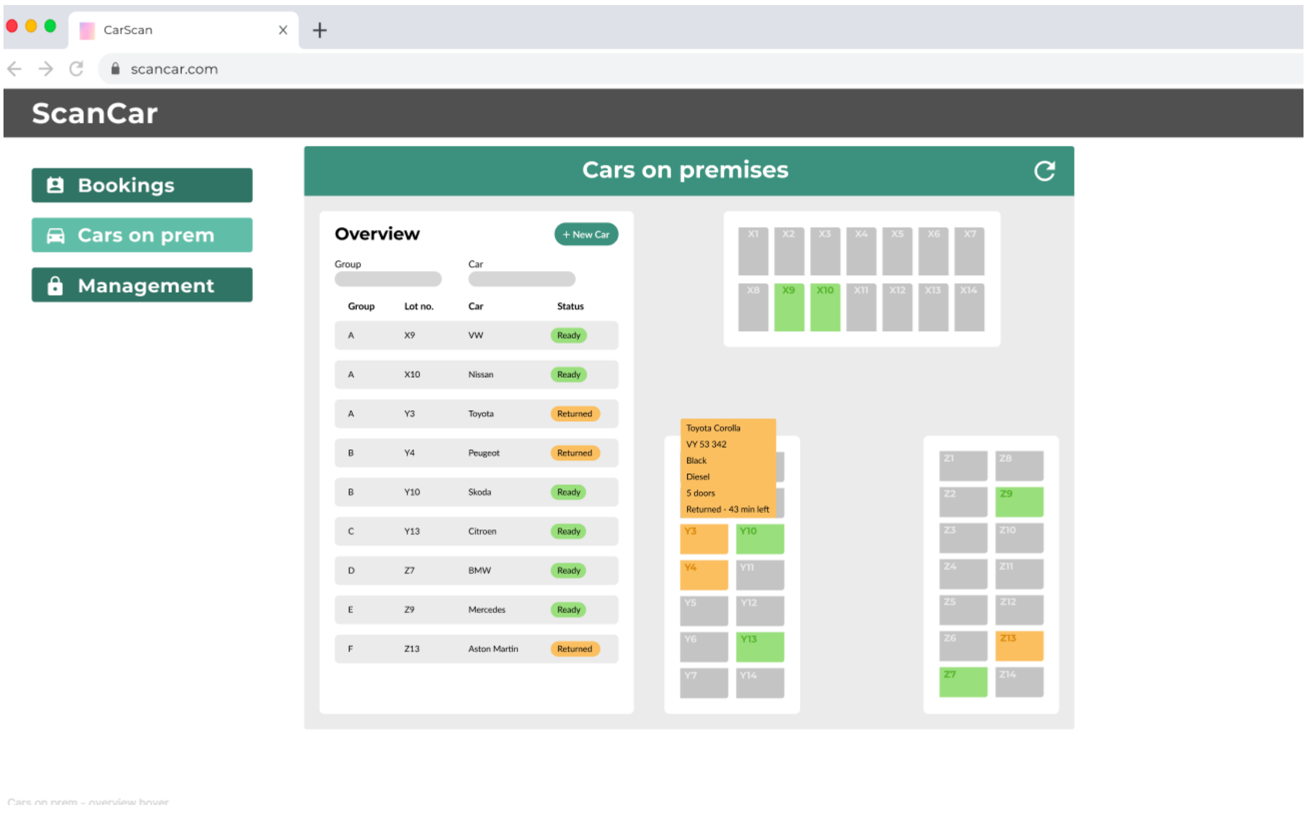
Figure 41: Booking deletion confirmation modal



Overview

Group	Lot no.	Car	Status
A	X9	VW	Ready
A	X10	Nissan	Ready
A	Y3	Toyota	Returned
B	Y4	Peugeot	Returned
B	Y10	Skoda	Ready
C	Y13	Citroen	Ready
D	Z7	BMW	Ready
E	Z9	Mercedes	Ready
F	Z13	Aston Martin	Returned

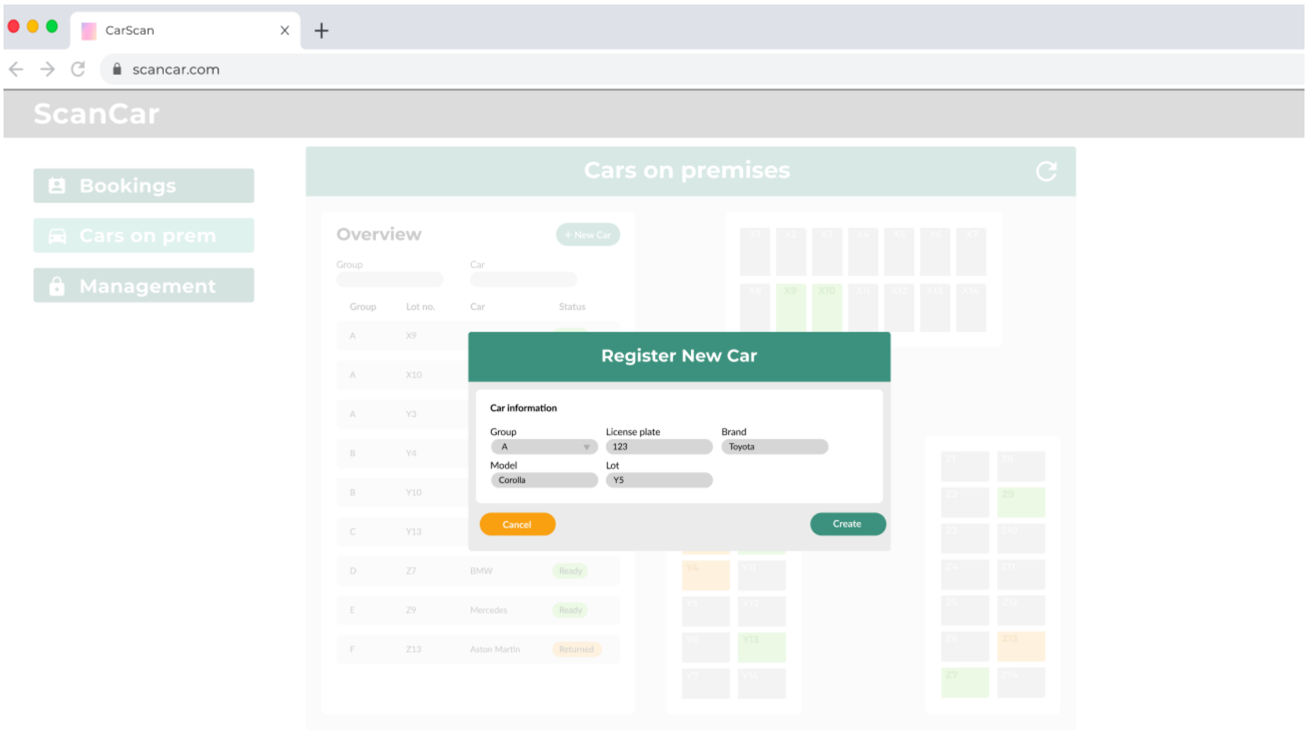
Figure 42: Cars on premises overview related to pickup, returns, and manipulation of cars. The blocks on the right represent lot spaces on a parking lot and would vary between offices.


CarScan
scancar.com
ScanCar
Bookings
Cars on prem
Management
Cars on premises
Overview
+ New Car
Group
Car
X1 X2 X3 X4 X5 X6 X7
X8 X9 X10 X11 X12 X13 X14
Y3 Y4 Y5 Y6 Y7 Y10 Y11 Y12 Y13 Y14
Z1 Z2 Z3 Z4 Z5 Z6 Z7 Z8 Z9 Z10 Z11 Z12 Z13 Z14
Cars on prem - overview hover

Group	Lot no.	Car	Status
A	X9	VW	Ready
A	X10	Nissan	Ready
A	Y3	Toyota	Returned
B	Y4	Peugeot	Returned
B	Y10	Skoda	Ready
C	Y13	Citroen	Ready
D	Z7	BMW	Ready
E	Z9	Mercedes	Ready
F	Z13	Aston Martin	Returned

Toyota Corolla
VY 53 342
Black
Diesel
5 doors
Returned - 43 min left

Figure 43: Cars on premises with hover menu



Car information

Group	License plate	Brand
A	123	Toyota

Model	Lot
Corolla	Y5

Figure 44: New car registration modal

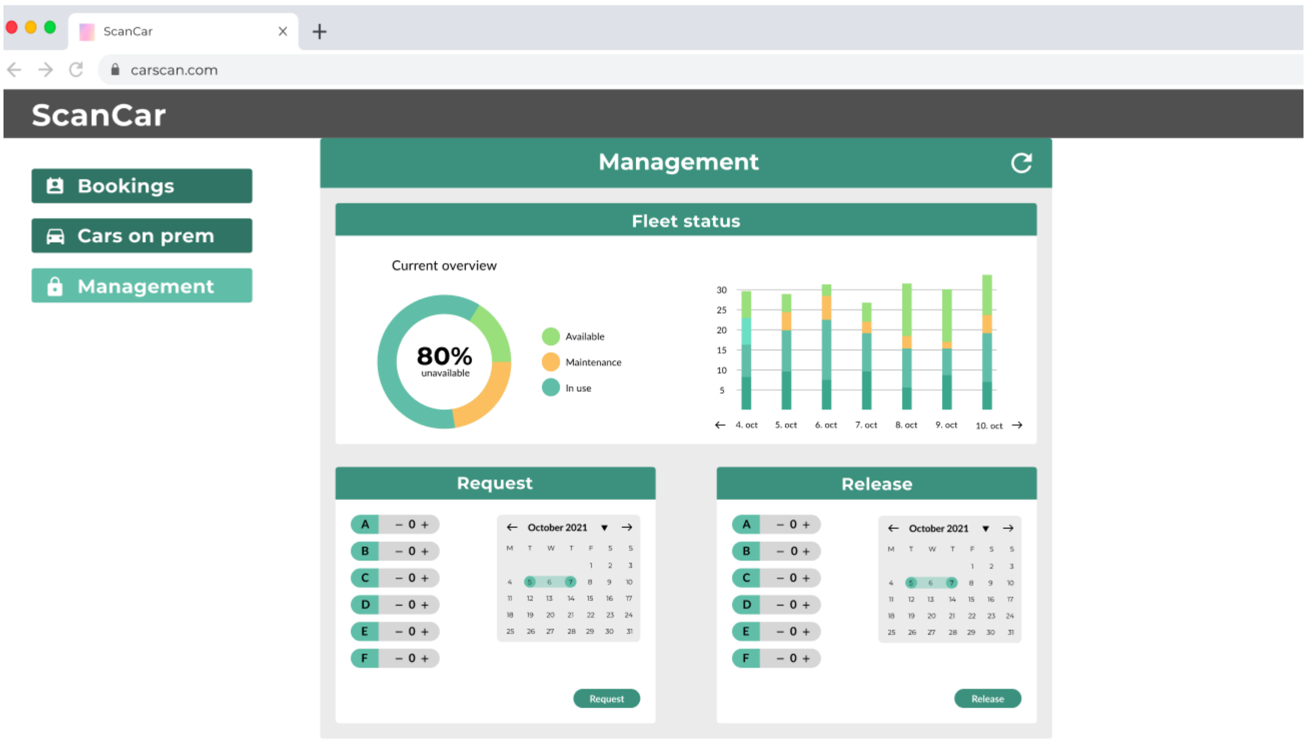


Figure 45: Management page related to fleet maintenance

9 Implementation

This section will describe the implementation of the HiFi prototype to a React app. As described in the paper, this implementation can also be categorized as having two iterations. The first was done cooperatively by Balin, Pedersen, and Pedersen. Where any changes in the second iteration were done exclusively by Balin. To ensure proper accreditation, this section will be divided into sections detailing each iteration.

9.1 Initial implementation

A repo of the initial implementation can be found via **this link**.

Of screens shown in the HiFi prototype, the initial implementation features the booking overview page along with new and edit booking modals, the car premises page along with a list of cars, and the management page. It is, however, missing the registration of cars modal.

Minor changes have also been made between the HiFi prototype and the implementation based on user feedback and design considerations. The edit button in the rows of the booking overview have been altered to a pen icon, and a trashcan has been added to allow deletion of rows. The bar chart in the management page was altered to show car availability based on groups per each day (based on local dummy data). This was based on the usability feedback that deemed the previous bar chart being less useful as it only showed overall car availability and not per group.

While the implementation is visually similar to the HiFi prototype, it is unfortunately severely lacking in many of the intended usability features. The prototype was envisioned as using a central database to handle storage and creation of items such as bookings, car placement, and car transfers. A database was created for the implementation, and the booking- and car data shown in the implementation is read from this database. However, editing, deleting, and updating elements to and from the database is not supported.

Beyond the missing features, the implementation also suffers from some severe bugs. The cars on premises page generates a parking lot overview and colors lots based on where the cars are located. However, this feature only works when the user navigates away from CarPrem and back again.

9.2 Second iteration

The second iteration is a clone from the original repo which has since been worked on. It can be accessed via **this link**.
The iteration implemented the following screens on the following files:

- Booking overview: `BookingOverview.js`
- Cars on Premises: `CarPrem.js`
- Management: `Management.js`
- Booking modal: `BookingModal.js` (component)
- Car modal: `CreateCarModal.js` `EditCarModal.js` (components)

The second iteration features numerous additions, fixes, and improvements over the first. Most notably, the application now features read, write, and delete features of bookings and car registrations. Furthermore, a new database table over customers has been introduced and elements involving these have been worked into the UI. The user can now see all customers, create new customer entries, and see each customer's associated bookings. When a new booking is created, a query checks if the customer already exists (based on their license identification). If the customer does not exist, they are added to the system, if they do exist the booking is added to their list. This would ideally be used to minimize data entry for the user, e.g. by filling out a customer's information when creating a new booking, if the customer information is in the database (this feature is not supported as of now).

The Cars on Premises page also now features a (rudimentary) menu, which details vehicle information, when hovering over occupied parking spaces. The menu is not finished as it should ideally be colored to indicate the car's status and the text formatting could improve, it is nevertheless an addition which serves to improve the user experience.

The implementation has also been given a massive quality assurance pass. This has had the benefit of increasing consistency and reducing bugs. For consistency and orderliness, all files are named with PascalCase, most files are organized within subfolders to clearly indicate their purpose, and comments have been added to every file where pertinent. The nasty bug related to coloring occupied car slots has been "fixed". Fixed, in this case, is a tenuous description for it, however, as the solution merely times out the rendering of the parking spots by a few seconds, within which the list should have been read. This is perfectly fine for most instances, but it could theoretically cause an issue with slower Internet connections if the timeout expires before the list is fetched.[4]

Despite the improvements, the second iterations does have lingering issues. First, it is still not possible to filter in the booking overview. Furthermore, the request and release buttons in the management overview are still function-less and the bar chart is still reliant upon dummy-data. I made the decision not to implement these as the underlying system required to facilitate their operation would be too time- and resource consuming. Creating new bookings, customers, and cars does not update the displayed list, requiring a page refresh.

[4]This issue should ideally be solved in the future, but for now it has a very low chance of happening as the list data is contained to a maximum of 42 JSON objects which is a negligible amount of storage.

10 Reflections

Generally, I think the process was marred by numerous issues that ultimately had severe negative impacts on the finished implementation. While I do not think the product could overall be considered a "failure", I do think the process suffered from misuse and misunderstanding of theoretical tools, technical limitations, and poor planning. In the following sections I will reflect on these in greater detail.

10.1 Planning and execution

The core issue of the development of this project, in my opinion, lies in fundamental flaws in either understanding or execution of certain tools and theories. While failure to grasp or execute can be understandable given that the project had a sort of "learning by doing" structure, given that later developments are built upon prior stages, the shoddy foundation caused a negative rippling foundation.

First and foremost, the initial usability test was deemed to be a failure to the extent that its findings were nigh unusable. This negatively impacted development as the further stages (re-design based on feedback) were not as fruitful.

Overall the project also suffered from a lack oversight regarding the formalized requirements of the problem statement. The problem statement emphasizes importance of retaining customer data. This emphasis was included in both the CREDO checks and virtual windows. In spite of this, they were discarded in the HiFi prototypes despite their inclusion being nontrivial. I have since added these features to the implementation to a moderate level of success, yet it is clear that they are not thoughtfully included in the design due to the lack of planning (e.g. automatic filling of details for new bookings if the customer is a prior). Exclusion of nontrivial issues can be defensible, e.g. when narrowing a scope to fit a deadline, in our case I do not think it was properly formulated nor justified. Ultimately, it led to the project suffering from lacking focus.

10.2 Technical difficulties

It bears noting that none of the project members had experience with React prior to the course. This, almost naturally, had significant impact on the technical aspects of development. I will reflect on some of these now.

10.2.1 Database development

Perhaps the biggest misstep of the initial implementation was the database design. The database design for the initial implementation was good. It considered numerous factors such as primary and candidate keys along with how these interacted with each other. Unfortunately, good database design leads to a large number of intricacies and constrictions that can make it difficult to work with.

In our case this manifested in a very well designed database that was a nightmare to program API calls for, ultimately resulting in none but read API calls being present in the initial implementation despite a massive amount of work dedicated to them. For the second iteration, I completely scrapped the prior database, created a new one with a simpler structure, and successfully implemented CRUD API calls for every element. While this database does not strictly follow proper design, it does serve its illustrative purpose quite well.

10.2.2 External React components

The application uses a number of external React components, namely modal pop-ups and data table overviews, both of which were used from the MUI library. Using external React components is one of the main benefits of React, providing large libraries with numerous features, saving developers the cost and time associated with creating their own solutions. However, a benefit that external components do not give over self-developed solutions is the intimate knowledge of the component's workings. In our case, this had several negative ramifications.

The MUI components are extremely feature rich. While this is a massive benefit, to a group of React beginners it can also be a major hurdle. The MUI components rely on custom APIs that do not necessarily follow form and structure of React. This can lead to confusion for beginner developers as they are unsure if the changes they want to implement should be custom-coded in React or if the API facilitates it. Even if the API facilitates it, the documentation can be extensive.

I think using the external components as we did is defensible. They provide built in features that we rely upon, where self-built solutions would likely not function nearly as well nor run as smoothly. Furthermore, all of the components have been modified with custom code which increased our familiarity with the components as well as giving them desired features. The other side of the coin is that attempting to modify those components, even with simple additions, proved to be highly time consuming, perhaps more than pursuing a self-developed option.

10.2.3 Responsive design with Figma

Figma proved to be an extremely powerful tool during the design process. Being able to quickly make sketches for how a UI should look while presenting it in a manner not too unlike a final product is a great commodity to have access to. One of the strongest benefits of Figma is that every element in Figma can be represented by CSS values, making it highly convenient to style HTML elements the exact same as their Figma counterparts. This is something we frequently used during the implementation development.

However, an oversight was done in regards to responsive sizing. For each screen in Figma, we utilized a template of a web browser with 1366x768 dimensions. Every single element within each screen was sized to fit appropriately following the design guide, using absolute values of pixels. This introduced a problem in the latter stages as it became apparent that any size declared in Figma could not be usable for the actual implementation. Sizes that look good on a smaller 1366x768 resolution would appear much smaller on most consumer monitors which have higher native resolutions.

While this wasn't a severe issue, it did cause headaches as sizing which was carefully planned for in the mock-up had to be re-evaluated in the implementation. For future projects, using relative sizing of elements in Figma within containers would have prevented this issue from occuring.

References

[1] Søren Lauesen. *User Interface Design A Software Engineering Perspective.* Pearson Education Limited, 2005.