Flexible Car Rental

- P1-Project -

Project Report
B219

Aalborg University
Department of Computer Science

Miro, Google Calendar, Monday, Google Docs, GitHub

Department of Computer Science
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Flexible Car Rental

Theme:
SDG 12 - Ensure sustainable consumption and production patterns

Project Period:
Fall Semester 2021

Project Group:
B219

Participant(s):
Andrzej Piotr Dudko
Casper Bjerre Jensen
Casper Christensen
Melih Özata
Nicolai Bergulff
Sebastian Daastrup
Thomas Bjeldbak Madsen

Supervisor(s):
Ondrej Franek

Copies: 1

Page Numbers: 65

Date of Completion:
December 21, 2021

Abstract:

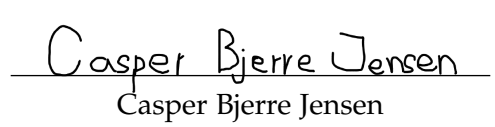
How can a software based solution incline citizens of Aalborg to use the cars that they already have, instead of buying new cars, and therefore on a small scale leading to a decrease in car production, resulting in a more sustainable use of resources?
This report has been made because of the P1-project on Aalborg University. The report's background comes from the topic of the Sustainable Development Goal 12 with its aim to ensure sustainable consumption and production patterns. The objective of this study is to make cars and renting more sustainable. The main motivator for the study is that cars are parked 95% of the time in their lifetime, and the study focuses on improving this with flexible car rental. The study goes through the use and need of cars, the CO_2 emissions from cars and a software solution for this. The program has not been used in Aalborg Municipality, however we made a survey where the interest was large. The study has a lot of potential, but the limitations were limited time and competence.

Contents

Preface

Aalborg University, December 21, 2021

Andrzej Piotr Dudko
<adudko21@student.aau.dk>



Casper Bjerre Jensen
<caspje21@student.aau.dk>

Casper Christensen
<caschr21@student.aau.dk>

Melih Øzata
<mozata21@student.aau.dk>

Nicolai Bergulff
<nbergu21@student.aau.dk>

Sebastian Daastrup
<sdaast21@student.aau.dk>

Thomas Bjeldbak Madsen
<tbma21@student.aau.dk>

Chapter 1

Introduction

When the car was first invented back in the late 18th century[31], people had not yet comprehended the scale of the car and the impact it would have on the entire world. What they had not comprehended either, was the consequences. As the car production rose and better roads were built to fit, the world became smaller than ever before, as a few kilometers became a question of minutes instead of hours. Unfortunately, the consequences were yet to be discovered, as CO2 emission rose exponentially without anyone realizing that it was actually a problem before it was too late. Today the whole world depends on transport, which is why we can not just revert the changes the car did to the world. But what can we do then? In 2015 the United Nations made 17 world goals that has the purpose of turning the world for the better, including CO2 emission. This is what this report roots in, made by 7 software students from Aalborg University. The project will cover not only the inefficient use of cars, but also the main aspects of a possible software based solution for the municipality of Aalborg. The project describes, analyzes and interprets different data based on the daily use of cars and how our solution could help solve this issue. We chose to focus on Aalborg due to time and resource constraints, but the solution should be applicable elsewhere.

Chapter 2

Problem analysis

To address the actual problem that is being processed in the project, there is lot of information that has to be collected. This will be done using various different sources, methods and approaches. The project will start at the base, the United Nations' twelfth sustainable development goal.

2.1 12th World Goal

Figure 2.1: World goal 12: Responsible consumption and production [11].

The twelfth goal is responsible consumption and production. This goal is aimed to ensure sustainable consumption and production patterns. Throughout the last many years we have used a lot of natural resources, but the problem is that we have not utilized them responsibly. Consumption of products is far beyond what our planet can provide, and we therefore must learn how to use and produce in sustainable ways that will reverse the harm that we have inflicted on the planet. The goal of sustainable consumption and production is about doing more and better with less. It is also about decoupling economic growth from environmental degradation, increasing resource efficiency and promoting sustainable lifestyles [11][32].

Target 12.1
The target 12.1 seeks to implement the 10-year framework of programmes on sustainable consumption and production, all countries taking action, with developed countries taking the lead, taking into account the development and capabilities of developing countries [32].

Target 12.2
The target 12.2 is by 2030, achieve the sustainable management and efficient use of natural resources. This including optimal use of resources that have already been used, together with an invention of sustainable alternatives to heavy resource consuming processes[32].
The goal is set to 2030, but the improvements of efficiency are not happening fast enough. Domestic material consumption (DMC) is a way to measure the total amount of materials used from economies [32]. Looking at a global level the Domestic material consumption is equivalent to material footprint and in 2017 it reached 92 billion metric tons.

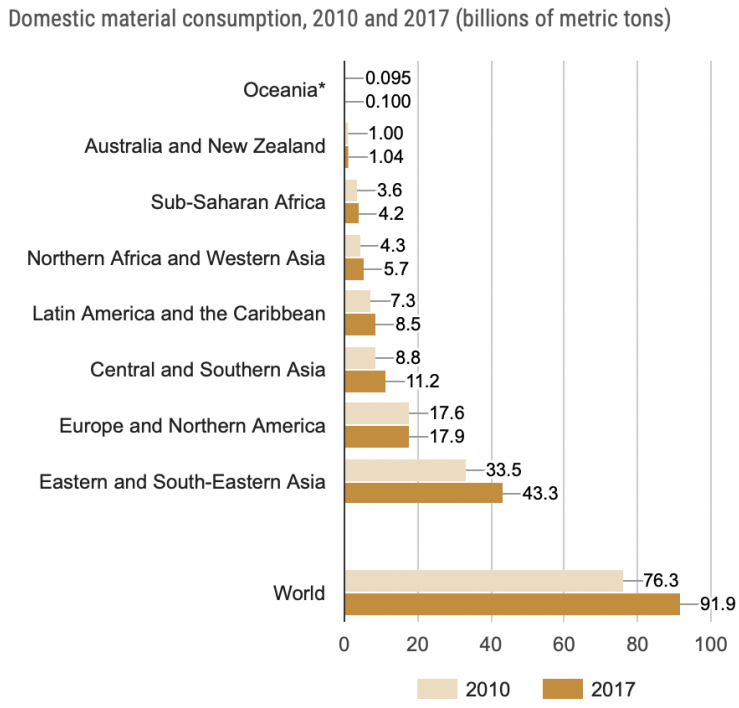


Figure 2.2: The table shows the different continents DMC from 2010 to 2017 [32]

As shown in figure 2.2 Eastern and South-Eastern Asia is accounting for two thirds of the Domestic material consumption world wide. The general trend of an increase in DMC, is most clear when looking at Eastern and South-Eastern Asia. This increase in DMC can also be associated with bigger economies and large scale manufacturing, such as China. Increasing DMC is a troubling as it indicates a hyper-productive consumer market where people buy cheap, new stuff, instead of reusing or investing in more expensive and higher quality products. This ten-

dency is often discussed concerning clothes and other smaller products sold in massive amounts, as fast fashion has become the norm through companies like HM or Shein. But cars are a big problem when it comes to over-consumption. The sheer amount of materials used and pollution emitted in car production is a massive incentive to make better use of the cars that have already been produced. Which is why this project came to be, as this problem does not seem to get the same spotlight as other industries, when it comes to sustainability.

Target 12.5
The target 12.5 is by 2030, substantially reduce waste generation through prevention, reduction, recycling and reuse [32].

A few facts and figures:

- Each year, an estimated one third of all food produced - equivalent to 1.3 billion tonnes worth around $1 trillion - ends up rotting in the bins of consumers and retailers or spoiling due to poor transportation and harvesting practices [32].
- If people worldwide switched to energy efficient light bulbs the world would save $120 billion annually [32].
- Should the global population reach 9.6 billion by 2050, the equivalent of almost three planets could be required to provide the natural resources needed to sustain current lifestyles [32].

The economic and social progress over the last century has been accompanied by environmental degradation that is endangering the very systems on which our future development and our very survival depends. Globally, we continue to use ever increasing amounts of natural resources to support our economic activity. The efficiency with which such resources are used remains unchanged at the global level. Thus we have not yet seen a decoupling of economic growth and natural resource use. Globally, the generation of waste is mounting. About one third of the food produced for human consumption each year is lost or wasted, most of it in developed countries [33].

When looking at stats from UN, the total global "material footprint" has increased 70% from 2000 to 2017.

- 1990: 43 billion ton
- 2000: 54 billion ton
- 2017: 92 billion ton

"Material footprint" refers to the total amount of raw materials extracted to meet final consumption demands. It is one indication of the pressures placed on the environment to support economic growth and to satisfy the material footprint needs of people. The rate of natural resource extraction has accelerated since 2000. Without concerted political action, it is projected to grow to 190 billion metric tons by 2060. What is more, the global material footprint is increasing at a faster rate than both population and economic output. In other words, at the global level, there has been no decoupling of material footprint growth from either population growth or GDP growth.

When looking at statistics from UN, the "material footprint" per capita has risen from 8.76 ton in 2000 to 12.18 ton in 2017[33].

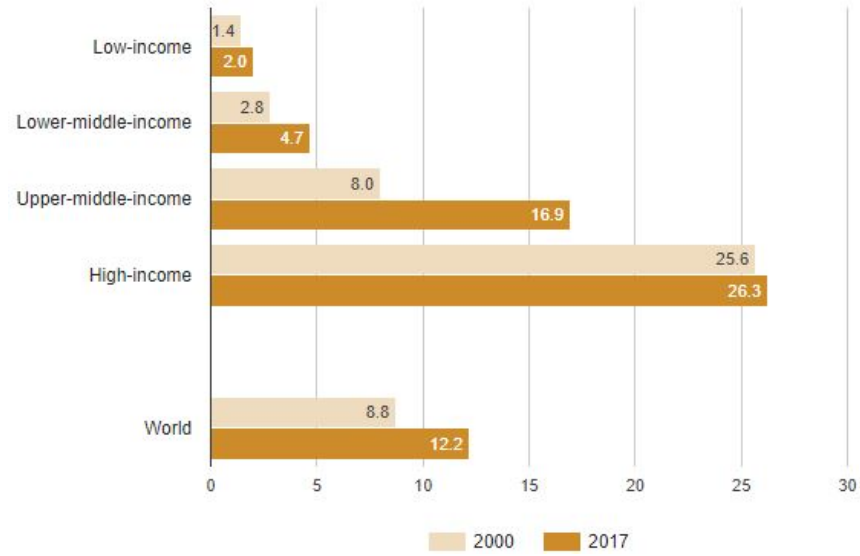


Figure 2.3: Material footprint per capita, 2000 and 2017 (metric tons per person) [33].

The "material footprint" per capita has also increased at an alarming rate. In 1990, about 8.1 metric tons of natural resources were used to satisfy an individual's needs. In 2017, that rose to 12.2 metric tons, an increase of 50%. That year, high-income countries had the highest material footprint per capita (approximately 27 metric tons per person), 60% higher that the upper-middle-income countries (17 metric tons per person) and more than 13 times the level of low-income countries (2 metric tons per person). The material footprint of high-income countries is greater than their domestic material consumption, indicating that consumption in those countries relies on materials from other countries through international supply chains. On a per-capita basis, high-income countries rely on 9.8 metric tons of primary materials extracted elsewhere in the world [33].

In 2002 the motor vehicle stock in **OECD** countries was 550 million vehicles (75% of which were personal cars)[1]. A 32% increase in vehicle ownership is expected by 2020. At the same time, motor vehicle kilometers are projected to increase by 40% in the same period.
The OECD organisation has a website with vast amounts of data, and they have the possibility to look at passenger car registrations throughout the years. The source displays a percentage change of car registrations from 2001 to 2018. Keep in mind, this is only passenger car registrations, this means that it does not include any other vehicular transport methods than passenger cars. The source explains that there is a massive fluctuation throughout the years from 2001 to 2018, when looking at all of the OECD countries the total percentage change from 2001 to 2018 is -14.4%. This means that from 2001 to 2018 a decrease in first-time registered passenger cars is apparent. This source does not specifically show that cars are not being bought, but just that there is a decrease or increase in percentage of cars being bought every year [16].
The increase in cars on the road does not only increase the material footprint in the production of the cars, but it also increases the emissions produced by the cars. The more cars driving on the roads, the more carbon dioxide released to the atmosphere. Passenger cars are responsible for around 12% of the total EU emissions of carbon dioxide. The EU is trying to reduce this by implementing targets to reduce the average emissions of the EU car fleet. Manufacturer's fleet is not to exceed its specific emission target for the given year, otherwise the manufacturer is to pay an excess emissions premium. These sorts of actions taken by the EU forces the manufacturers to make zero- and low-emission vehicles to not exceed the emission target[3]. But why has the car become such a threat to the environment, where did it all start? To better understand this, the project briefly jumps back in time to when the car was invented but the consequences were not discovered.

[1]**OECD** is an organisation for Economic Co-operation and Development which consists of 38 countries.

2.2 5 W's and 1 H

We first created a mind map or a 5W and 1H diagram on "Too many inactive private cars", this gave us a possibility to look at the issue very broadly. We realized that looking at the topic so broadly was difficult and the issues that came along were too broad to work on, we then started to work on a more specific topic that would help us to know what we should focus on. But we did see that there were some possibilities to work on even more therefore we started to look at some of these topics we discussed within the group. After a while we found that "Flexible Car rental" was something we as a group could in fact work with. Therefore we created another 5W and 1H with this topic. Even though this topic was still broad we started to focus on a path with private car owners, which helped a lot and it was easier to focus on. It helped us narrow down the overall broad issue with "Too many inactive cars" to "Flexible car rental" focusing on the private car owners.

The result of all of this is the clarification of what and where the project should be steered towards.

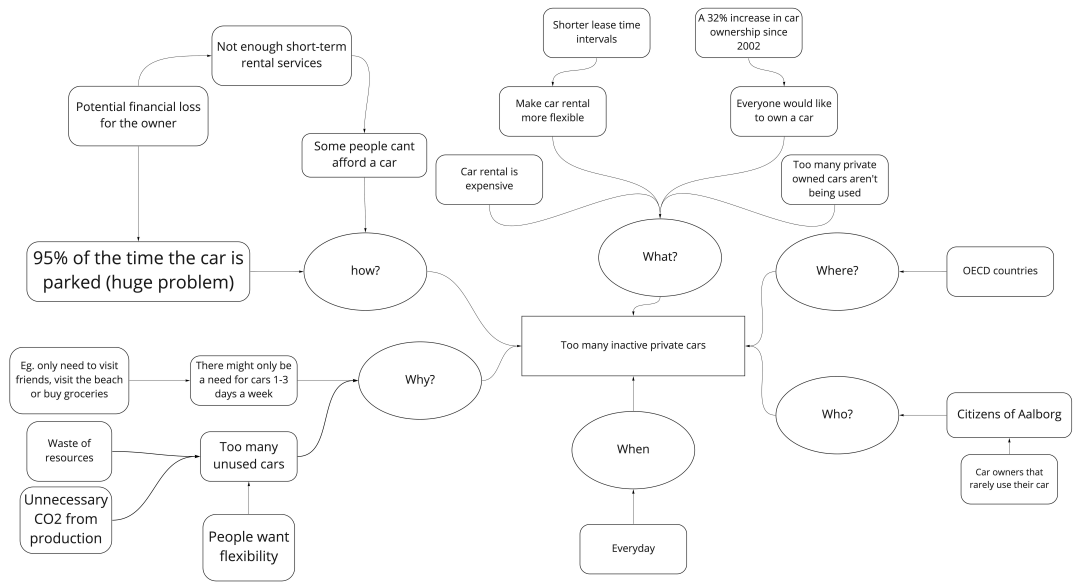


Figure 2.4: 1. 5W1H with the topic "Too many inactive cars". Made with Miro.com.

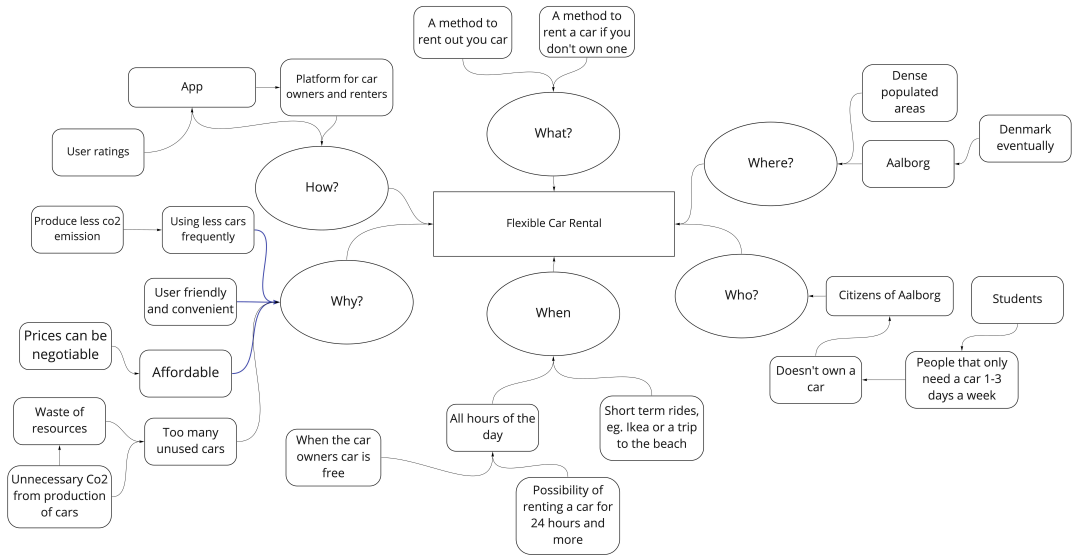


Figure 2.5: 2. 5W1H with the topic "Flexible Car Rental". Made with Miro.com.

2.3 Concept map

After we made the 5 W's and 1 H models, and discussed about the topic Flexible Car Rental, we had a lot of ideas on how this could be utilized in Aalborg municipality and what relevance it has to the UN World goal number 12. We therefore made a concept map, see figure 2.6. The concept map consists of elements that are related to the topic "Flexible Car Rental". Some elements are related to each other, this is visualized with a line, showing the direct connections.

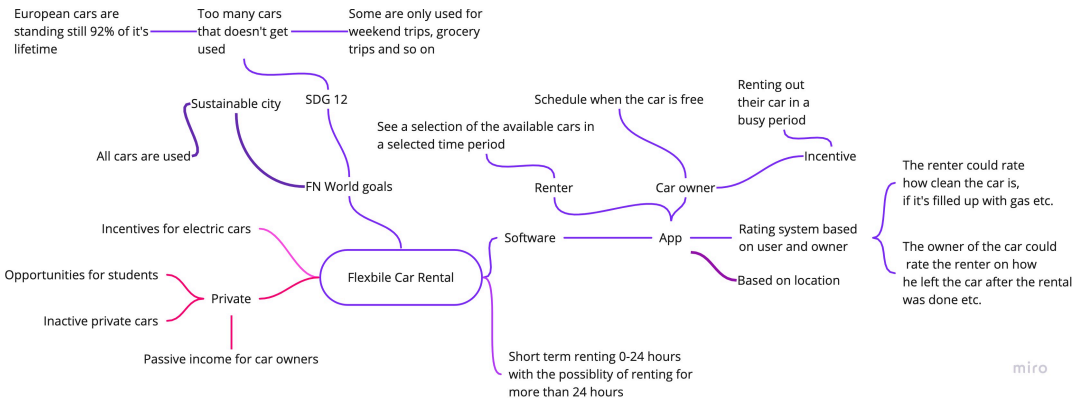


Figure 2.6: Concept map about the topic "Flexible Car Rental". Made with Miro.com.

2.4 A Brief History of the Car and Rental

The first car was officially invented in the late 1800's, which after some time of development, was influenced by Henry Ford, who introduced the world to his manufacturing method. This was based on assembly lines being set up, making it a lot easier to produce a car fast, resulting in making the car cheaper. This also meant it was easier to access for lower class citizens, which would furthermore increase demand. This would also prove to be problematic, because the evolution of the car driven by the money from the citizens, would mean that society would be adapted to having a car, making it less practical to not have one, whilst the companies would continue making even more cars. There was one aspect that the whole car evolution did not take into account before the damage was already done, CO2 emission. This had a huge impact on knocking the earth's atmosphere off balance, as the CO2 emission from both producing the car, driving the car and using gasoline were huge in conjunction with the amount of people owning a car[31]. Cars were also made primarily of steel in the late 1800s, which made the fuel efficiency quite bad, because the engine would have to move a lot unnecessary weight that did not make up for it any way[15]. As a lot of people bought a vehicle, some were also in need of renting one, maybe because they had a single errand that required them to use a car or other transport, but it would not be worth to buy one. But how did this renting system take shape?
Before cars were invented , people were renting out their horses. Then, when cars were invented those same people realized that there is a demand for renting out cars as well. This means that car rental has been around for more than a century [9]. The first data known of renting out a car is from the 1904 where a bicycle store in Minneapolis started to rent out cars, and some years after the German company which is well known today Sixt was established. It started with only 3 cars to rent, but lots have changed since. Sixt is one of the worlds largest car rental companies as of today [29].
In the 1990s lots of other Americans realized there was a huge potential in renting out cars while concluding that there also was a huge gap in this type of business. Many Americans started different types of renting companies, some of them are still operating today such as Hertz. Hertz are well known all around the world, and especially for tourists because they are one of the largest companies on the market for renting cars [30]. John D Hertz purchased the company from Walter L. Jacobs and the original name of the company was 'Saunders-Drive-It-Yourself system' [9]. This company started only renting out one car model named the Model T. Nowadays, companies have hundreds of different cars and car models from different brands. In the 21st century the demand of renting cars has not been decreasing [9]. These different rental companies can be found all around the world, in major and minor cities alike. As we are becoming a more and more globalized

world, people are travelling more and more at a higher rate and therefore the rental of cars and different car models are higher than they have ever been. Since the beginning of car rental, lots and lots of paperwork were required, but nowadays you only need a valid driver's license and a working credit card [9]. Now that a brief history over cars and car rental have been covered, it is easier to visualize how the problem of cars occurred to begin with, but one question that stands would be: "what is it with cars today, that makes them so inefficient, production and usage wise?". This will be explained in the following section.

2.5 CO_2 emission from cars and why it is inefficient

Many factors play a role in the cost of production for cars, with some of these factors being volume and luxury features in a car. With a higher production volume, the cost of production will decrease, as automation and standardization make the process less labor intensive and utilizes the raw materials better. As the base product in a high volume car manufacturer is made so efficiently, adding luxury features to the production will greatly increase the costs. This is due to each individual car having to be installed with different components, which increases labor and material expenses. Some car brands are known for their high-end cars, which are expensive to produce, such as Lamborghini and Ferrari. This is partially due to the high production costs, which stems from low volume manufacturing and costly raw materials [10].
Some of the materials that are used for building modern cars are steel, plastic, fiberglass and titanium etc. [35]. Some of these materials are a natural resource that comes from iron, which is a nonrenewable resource [5], though it is a highly recyclable one. One of the largest industries of consuming raw materials are in fact the automobile industry, even though they do not turn the materials into e.g steel, they will have a company that turn raw materials into the materials they will use [12].
The CO2 emissions from the car production companies have since 2005 decreased with nearly 50%(48.5), but when compared to how much the CO2 dropped per individual car produced, the percentage is considerably lower, only 33%. This concludes that the industry really puts an effort in, when it comes to reducing their CO2 emission from their production. With this in mind, car production is still a major offender when it comes to CO2 emissions, especially when factoring in the inefficient use of cars [4]. Now the production based inefficiencies have been covered, but what about the usage based ones? There are a lot of inefficiencies in the mobility system in Europe. The European car is parked around 95% of the time, which is a big waste of useful resources that have been used to build these cars [1]. And even when the car is being used, it is on average only carrying 1.5 people around, including the fact that a lot of the time is spent sitting in traffic or looking

for parking spaces. The efficiency of the car itself is also problematic, as a lot of the energy is not actually used to move any people. In petrol cars about 86% of the energy never reaches the wheels [8].
Another big disadvantage of petrol cars is that they are a major source of air pollution to which the OECD attributes more than 3 million premature deaths per year. However, some of these problems can be significantly reduced with more electric vehicles on the road, as they are at least three times more efficient and do not pollute the air like petrol cars [28].
The inefficiency in the still standing of cars and the number of people carried per trip will however not be solved simply be having more electric vehicles. Solving this requires a lot more from the people who use the cars. A change in car use ethics and a bigger awareness of the consequences are essential. Around half of the area in the cities are made up of infrastructure for transportation, yet congestion on the roads is still a big problem. All these combined are surprisingly high inefficiencies for a sector as big as the transport sector [8].

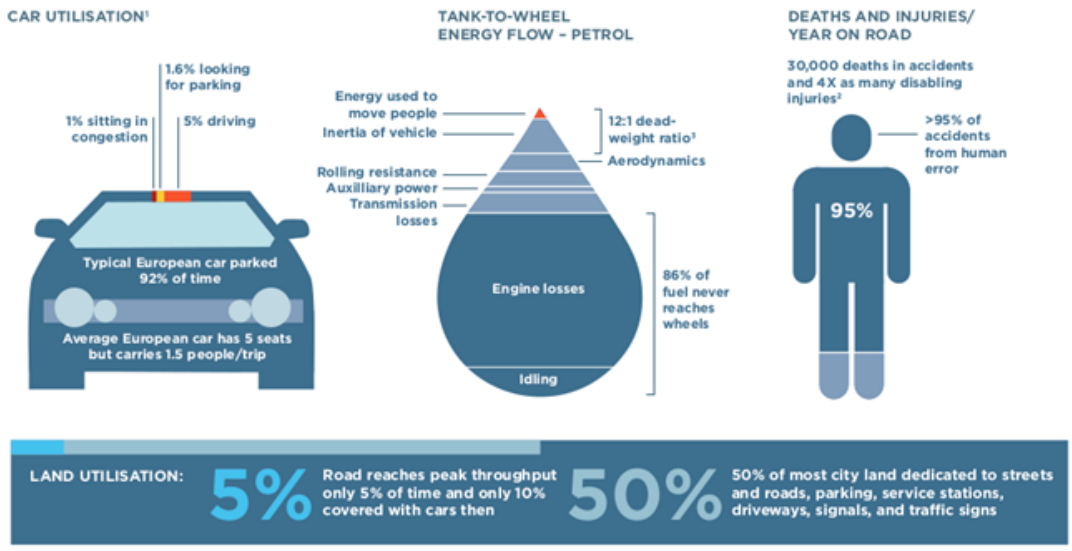


Figure 2.7: Waste in the mobility system [8].

The number of cars is still increasing and there are no indications of this stopping soon. So instead of decreasing the number of cars on the roads like we should, we are still purchasing more cars even though we are not using our cars very efficiently. In Denmark there were just over 2.1 million private cars in 2010, that number has increased to over 2.7 million in 2021. This is an increase of around 28% in just eleven years [26].
So, this of course leads to why people are choosing to buy more and more cars. One of the big reasons is the big economical growth in general. People need something to spend their money on and if households can afford it, they would probably opt

to have two cars for practicality. A lot of factors play in to whether families have cars or not. Some of these factors are which city they live in, their family type, size and their socioeconomic status. So, families living in larger cities, with a lower income and without children are less likely to own a car, while families living in rural areas, with a high income and children are more likely to own a car [27].
A big factor on ownership of cars is also public transportation, as access to high service level of public transportation in general lowers car ownership. Access to public transportation coincides well with geographical placement. People living in smaller cities or in the countryside have less access to public transportation and are therefore more likely to own a car. Families with high access to public transport are more likely not to own a car, whereas families with very limited access to public transport are more likely to own a car or maybe even two cars. The access to public transport has less importance on people with high commuting distances, and people with high commuting distances are more likely to own a car. This is most likely because if people work outside larger cities access to public transport is very limited. A large journey with public transport outside of major cities is very difficult [13].
Economically it makes sense to simply use public transport. The infographic [7] shows that a regular bus can already be more than 2.5 times as efficient at transporting passengers than cars. But public transport has some limitations.

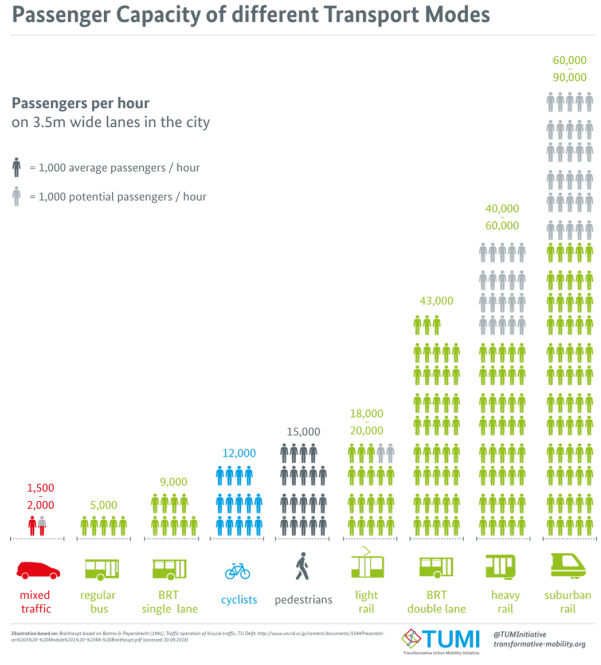


Figure 2.8: Passenger Capacity of Different Transport Modes [7].

- Schedule. When dealing with public transport the user always needs to ad-

just their plans to the schedule of public transport. This may not be an issue when busses come and go every few minutes but in smaller cities sometimes the frequency of busses are too inconvenient to justify the trip.

- Range. If a person is looking to travel someplace less populated the bus might not have a route to go there. Another thing adding to range is how far the users starting point and end point is from the nearest stop. This in combination with the schedule can create situations where half the trip would be spent waiting or walking, especially when changing busses mid-travel are required.

- Security. If the user brings many things with them on the bus they might not be able to 'protect' it meaning that they risk losing their belongings as well. While on a bus the passenger cannot just leave their items and go somewhere else. Something that they could do with a car.

- Ease. The problems mentioned above can all be overcome by the user by sheer will, you can walk anywhere if you really want to, but the goal is to make it easier while also putting more cars into use that otherwise wouldn't see any. With car rental its up to the user to choose the schedule of your transport, and therefore the range, and you can keep all your belongings safe in your the car. As well as an easier method to carry large cargo, and transport it from A to B than a bus, train and so on.

Now that a comparison between the cars and other alternatives has been done, and it can be concluded that there is a vast difference in several aspects between them, it would be appropriate to find out if this project's concept of a solution actually targets a solvable issue. This means that instead of getting rid of the car, how about a solution that ensures better use of it? This will be done in the following section.

2.6 Is there actually an issue?

To determine if there is a real problem at hand in this project, it was decided that a survey would be published, that finds out if the people of Aalborg also recognizes or realizes a problem with cars in Aalborg. Furthermore the survey will research possible interest in renting and renting out a car, which will work as a frame for the initial design of a solution. The survey was published Wednesday the 17th of November, and the data extracted to this project is on the 22nd of November. Mind that more answers can be submitted to the survey from this date, so the data that is being analyzed in the project can be different in the future.
The data that was extracted from the survey concludes, that their is a majority that believes, that there are too many cars in Aalborg.

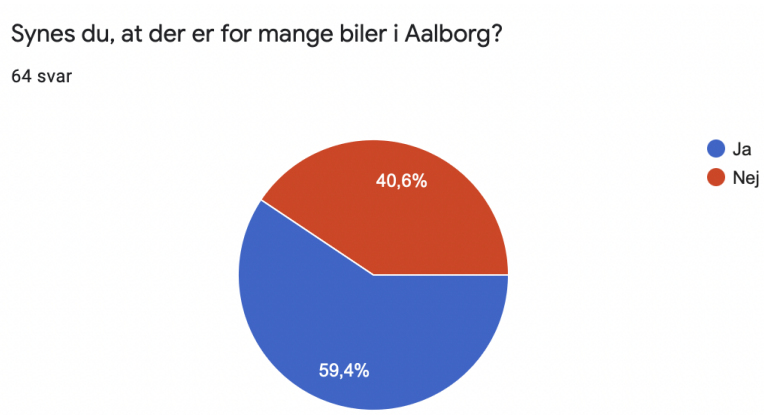


Figure 2.9: Do you think there are too many cars in Aalborg?

It can also be concluded that a majority of the respondents are amongst the age group of 22-25 and also being students. This may also be affected by the way the survey was published, through Facebook, but this is just an assumption and will not be proven in the report. A majority also stated that even though they think there are too many cars in Aalborg, they do not want to rent out their own car, mainly because of mistrust and the believe that the car renters will not treat the car according to the requirements of the car owner, as stated in the following quotes from the survey:

- "At folk ikke kan finde ud af at køre"
- "Folk ødelægger og sviner bilen til"
- "Skader og dårlig brug"
- "Ingen forsikring"
- "Folk ikke behandler ting der ikke deres eget ordentligt"

Translated to english this means: "People do not know how to drive", "People destroys and messes up the car", "Damage and bad use", "No insurance policy", "Do not treat things that do not belong to them self right"
To take this data into perspective, so that it can be utilized in the program design, it is concluded that the program will need some sort of insurance policy and maybe some kind of monitoring aspects that gives the owners a sense of safety. Determining whether or not this will be implemented in the initial program design will be discussed later.
Another aspect of the survey is where the respondents are broken up into two sections depending on their answer to whether or not they have a car. If they have a car, the respondent will be dealt car owner based questions and if they do not, they will be dealt car renting based questions. As for the car renting based questions, there are 62.5% of respondents who think that there are fine possibilities for renting a car, while 68.8% of respondents think that it could be attractive to rent a car on a short term basis, for example single rides like getting some furniture in IKEA. So a majority believe that it is easy enough to find a car to rent, but most likely not on such a short term basis, which may or may not affect the choice between renting a car or getting your own. Most of the people whom own a car, would not rent out their car, mostly because of the insecurities of renters not treating their car with the same respect they do.
In the survey the participants get asked if they think this sort of program would be a good idea, and most of the participants think this would be a good idea.

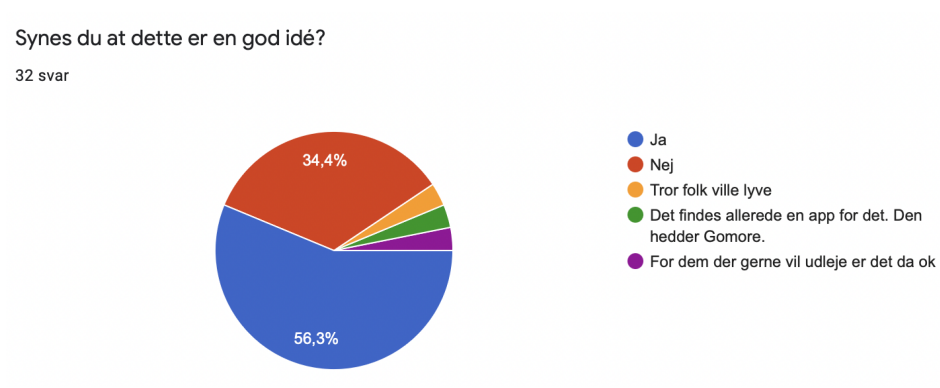


Figure 2.10: Do you think this is a good idea?

As shown in the above figure 2.7 there is a majority of 56.3%...who thinks that this is a great idea, and only 34.4%...who thinks that this in a bad idea.

2.7 What differentiates this project from the competition?

To visualize what makes this project different from the competition, we made a chart detailing some of the core concepts of our idea and other car rental services.

	GoMore	Tadaa	Share Now	Our Project	Hertz (physical car rental)
Renting intervals	Minimum 4 hours	Per day or 30 min intervals	Minutes or day packages	Hourly	Days
Car Ownership	Private	Tadaa	Share Now	Private	Hertz
Prices	The car owner decides what the car should cost, GoMore takes a cut.	30,- per 30 min 495,- per day	As low as 1.5,- per min Package deals	Car owner decides the hourly cost of the car. Per kilometer.	The price depends on different variables such as age, country and which car you want to rent.
Location	Denmark	Aalborg, Randers, Aarhus, Horsens, Vejle, Svendborg, Næstved and Hedehusene.	Copenhagen	Aalborg	Copenhagen, Aarhus, Aalborg and Odense. Mayor cities in Europe.
Availability	The car owner decides the availability of their car, and to whom they rent it.	Either everyone can sign up for the specific area, or you need to be a part of a housing association	You can find the cars in and around Copenhagen, whenever they are vacant	The car owner determines when their car can be rented	The cars can be rented at the Hertz locations in the cities listed above.
Requirements	Renter has to be at least 21 years old. Renter needs to have had their drivers license for at least 1 year.	You just need a valid drivers license.	Renter has to be at least 20 years old, up to 25 for larger cars. At least 1 year of driving experience.	Valid drivers license. Personalized requirements by the the car owners.	Must have had a valide drivers license for at least a year.
Insurance	The rent is insured and includes hull, liability and roadside assistance. Through "Tryg insurance".	Liability and hull insurance is included in all TADAA! subscriptions. The insurance is valid in Denmark, Germany and Sweden. Up to 3,000DKK in deductible	Liability and hull insurance with a deductible up to a certain amount.	Liability and hull insurcance with a deductible up to a certain amount.	Liability and hull insurance with most of excess covered. An insurance package is available which entirely covers excess, with the requirement of being at leat 23 years old.

While it is clear our project shares a lot of ideas with competitors, such as the hourly rate, the key difference is in where the project combines features of different competitors. While relying on private users providing the cars for the rental service like GoMore, the project offers a more flexible deal with the hourly rate of Share Now. Pricing is also an important feature, as it is believed that the car owners should decide on the price of their car. While no price regulation may be odd, it is assumed that the market would quickly stabilize due to supply and demand.

2.7.1 Why flexible?

What is meant by flexible? When comparing to other car rental companies where the car owner rents out their car, how is the competition not flexible?
When looking at GoMore, their renting intervals is determined by the owner of the car. Though, when researching their website, it was only found that the minimum amount of hours a user can rent is 4 hours. There is a catch though. Lets say that a user wants to rent a car for 4 hours, the price would most likely be what is paid when wanting to rent for 24 hours. That is a problem, if the user only wants to rent a car for 4 hours the user should not be paying for 24 hours. The price could be a bit less, but the research showed that a user most likely pays for 24 hours when wanting to rent for less than a day.
Tadaa is more flexible with their renting interval since a user can rent their cars per day or with 30 minute intervals. Their catch is that they only have a limited amount of cars, and these cars are located specific places, and that might not be convenient for most people.
ShareNow is incredibly flexible in terms of their renting interval. Here a user can rent a car per minute, that means that the user pays for every minute the car is rented. Even though ShareNow's cars are only limited to the Copenhagen area, they are scattered all around, so the chance of the user being near one is high. Their catch is that the cars are mostly smaller than a sedan, and if a user is looking to rent it for longer periods, then it will be expensive.
Hertz. Their renting intervals are at the minimum of 1 day [19]. This is not very flexible and not attractive to customers that only want to rent the car for a couple of hours to either go grocery shopping or a trip to the Zoo with the whole family. One of their benefits is that a user can rent a car in one place, and then hand it in at a different spot.
How do we differentiate ourselves from other renting services?
The project is focusing on the renting interval of one hour. This gives the absolute maximum amount of flexibility. This means that no matter how long the car is rented, the price is always determined by the hourly rate. Combined with this is the flexibility of the placement of cars. Since it is the car owner that rents out the car, the car could be located everywhere around Aalborg, since the project is focused on Aalborg Municipality. When dropping off the car a radius should be determined by the owner to where it is allowed to be dropped off, as it would be the most flexible for the user. The hopes for the program is to be so widely used, that you should not have to walk very far to find a car that can be rented, and that lives up to the user's requirements. Since the program appeals to car owners that own all sorts of cars, this gives the car renter the possibility to find a wide variety of cars, should it be a hatchback, sedan or a van.
With the renting interval, the location of the cars and the different kinds of cars, a lot of flexibility is included for the car renter.

2.8 Peoples need and desire for cars

We will now look at what people are travelling for. In figure 2.11 we can see the distribution of distance travelled by people from 11 different EU countries for urban mobility. We can see that there is a difference from country to country on the distance travelled for different purposes. In Denmark for example we can see that the distance travelled for work is comparatively low, while the distance travelled for leisure is higher.

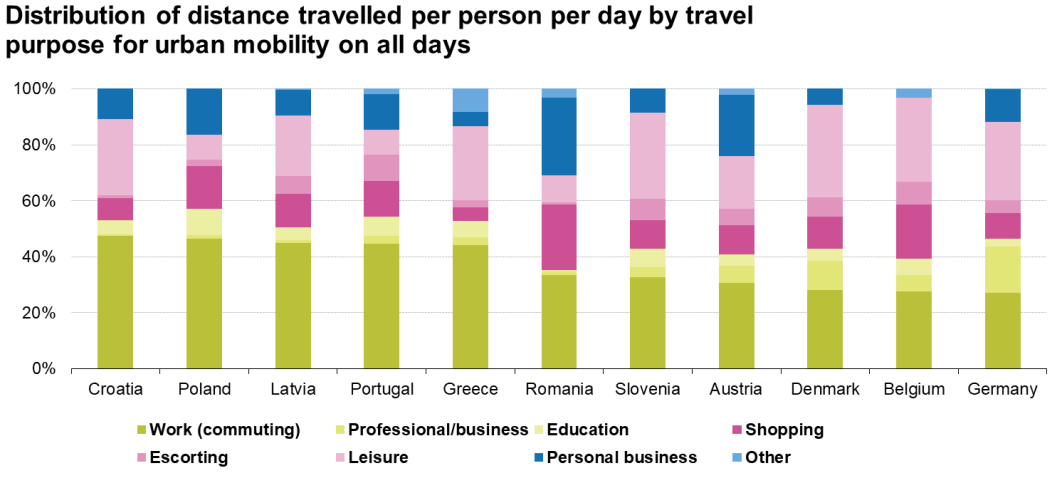


Figure 2.11: Data from eleven EU Member States.[6]

Now let us focus on our main city of interest, Aalborg. As we have mentioned earlier the number of cars is increasing in the world and in Denmark. This is also the case for Aalborg, as the number of cars has significantly increased here as well. The number of cars in Aalborg municipality have increased from around 72.000 cars in 2010 to 93.000 cars in 2021. This is an increase of nearly 30% in 11 years [26]. If we look at the increase of people in Aalborg municipality, we see an increase from 197.007 in 2010 to 219.509 in 2021, which is still a significant increase, but does not follow the increase in cars. The increase in the number of people in the 11 years is just over 11%, while the increase in cars was 30% [2].
So, people in Aalborg municipality clearly sees an increase of cars. One of the reasons for this could be high number of people commuting long distances for work. In Denmark people commute further and further for work, and in 2018 the average Dane commuted 42.5 kilometers to and from work, and this upward trend is going to continue. So, people seem to be willing to live further away from their workplace and use a bit more time on commuting [14].
This could also include the people commuting to and from Aalborg municipality, as this number also increased in recent years. In 2019 the number of people com-

muting to Aalborg municipality was about 30.000 and people commuting out of the municipality was around 24.000. These numbers only include people commuting out of or into the municipality, and not all the people commuting inside the municipality, which is a quite big municipality [17].
A lot of Danes are satisfied about the public transportation system in Denmark.[34] There seems to be an overall willingness among Danes to transport themselves more environment and climate friendly. This includes buying a more environment friendly car, using the bicycle and public transportation more frequently [34].
There is already a big focus on improving the public transportation system in Aalborg. One of the examples of this could be the big project Plusbus, which aims to improve the bus system significantly in Aalborg. This will be made possible with big infrastructural changes such as dedicated bus lanes and special stations [18].
So, the focus of reducing cars should be placed on people living in the city of Aalborg or near Aalborg, as people living far away from Aalborg will have a hard time finding alternative means of transportation. People living in or near the city will probably still need access to a car for certain purposes such as transportation of larger items and to go visit family living far away or with bad access to public transportation.

Chapter 3

Problem statement

As citizens of Aalborg municipality continue to buy an increasing amount of cars every year and the normality of owning a car has risen, the issue is never ending. By looking at figure 3.1, it is seen that in 2007 there was 67465 passenger cars in Aalborg municipality, and now in 2021 there is 92989 passenger cars, that is an increase of 37.83% [25]. With this increase of 38% combined with cars that on average are parked 95% of the time, there is a problem when looking at the 12th world goal of responsible consumption & production [1].

So here is the problem statement:
How do we ensure with a software based solution, that the citizens will be more inclined to use the cars that are already there, thus slowing down the increase of cars in Aalborg, leading to a decrease in car production.

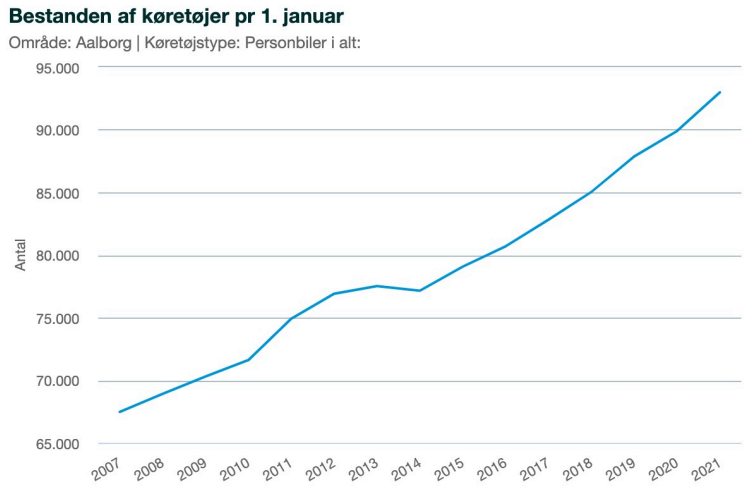


Figure 3.1: Amount of passenger cars as of January 1st from 2007 through 2021 [25].

3.1 Stakeholder analysis

The different stakeholders
Government, the team, bank, customers, shareholders/owners, IT support, media, society, future generation.

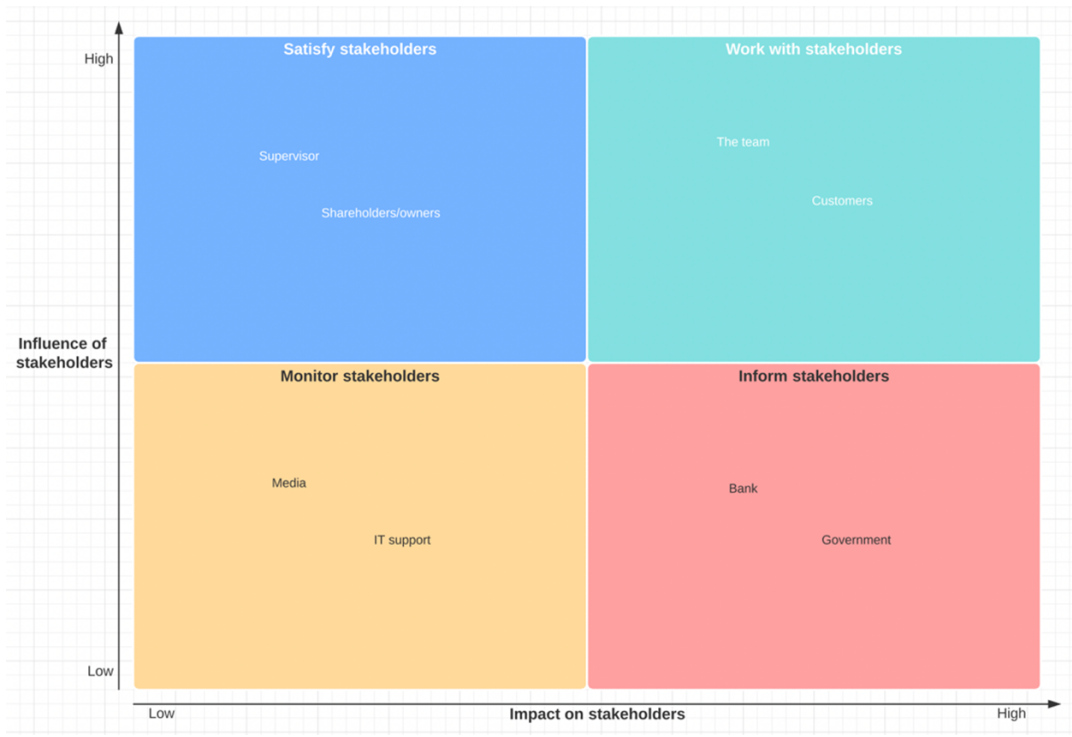


Figure 3.2: Stakeholder analysis.

Goverment

Entrepreneurs are good for the economy and employment. The government supports innovative startups and helps them grow. New and improved products, services, or technology from entrepreneurs enable new markets to be developed and new wealth to be created. Therefor, the goverment would want to always see ney changes and progress, which is why having an young group of people to create a product, to solve a problem [21].

The team

The team's interest is high, because this is first time they will be working on a project and creating a product in order to solve it. The overall theme for this project is interesting as well for them, which gives them higher interest and letting

them put more effort into the project. Involvement from the team is a central part of this. Their impact is high, since the problem will be solved by them. The whole operation and its functions will be controlled by the team itself. The team will become satisfied with the project if they achieve a learning experience, and manage to successfully solve the demanding problem, with a creative product that they are happy about.

Bank

Banks have huge impact when it comes to letting ideas become reality. They are able to provide a company with credits, in order to create something. Their involvement also means they want to earn extra money in return. Being a bank, and having the important resources is an important part of a project, trying to create a product requiring credit. Therefore their part of the process is important as well. By giving the project the right amount of loan, the expectation is to having the team earn money from their project, which would lead to paying the bank back, which is what would keep them satisfied, making them happy about the business [20].

Car renters/Car owners(Customers)

The interest and the impact customers have is sizeable. People in Aalborg are looking for an efficient way to complete some tasks for themselves, with a car, in the best way possible, which would be something that is easy, and inexpensive for them. They have a hope that there will be a system, easy to use and efficient for each individual person, who either does own a car, or don't own one. Customers have a huge impact on this project, since they are the ones the project team are aiming to please. They are the ones to use the product, and their opinion of it all is important. Their needs will be important for the project team's direction toward the goal. The way to handle the customers, is to create a product they would like to use and be happy about. If they are happy about the product, then the team will become successful.

Shareholders/owners

Future Shareholders and the owners of this project have a huge interest in this, since they decided to invest their time, and credit into this idea, expecting the team to perform well, in creating an interesting solution to the problem. Shareholders helps the team realize their idea, and are here to provide credit. Therefore they would like to be satisfied as well, with what they invested in. To handle the Shareholders, the team needs to be able to create profit out from this project, in order to make the Shareholders satisfied [24].

Figure 3.3: Is this a service that you would like to use?

IT support

Having a product made of software, will not always be flawless. Therefore the right team is needed to support for such. But its not always the case that problems occur, which is why IT support does not have that much interest or impact on the project, but there is still a role for them to have. IT support will always be a part of a software product, and they will always be beneficial in case anything would occur. By having a project team to create a product, a support team will be adding the extra volume for the project, so that everything would be smooth as possible. The way IT support functions is by helping and solving technical problems. If that is possible, they will become satisfied, same as the customers getting the help [22].

Media

The media have a special relationship between them, and the companies that are talked about. It could either be beneficial for the company they are writing about, or be a downside for them, since the role of the media is to talk about the result of the product, made by the companies. Therefore, the media has interest, and have an special impact [23].

As the problem analysis and problem statement is done, we know there is a problem we have to solve. The next part of the study is the software solution.

Chapter 4

A Software Solution

As the problem is now more specified, it is time to work on a software based solution, that can positively impact the problem at hand. The main goal of the program is to provide a surface for renting and renting out cars, which has potential to become a global service. Though as the project is of the size that it is with a limited amount of time, the program will for now be focused on the municipality of Aalborg. What this means for the program specifically is that the process of renting and renting out cars will be based on locations in the municipality of Aalborg.

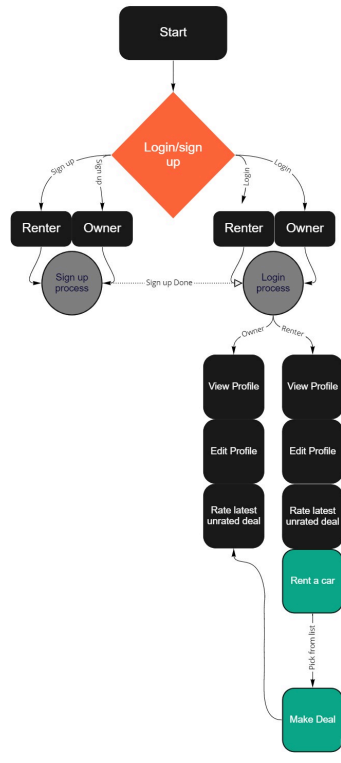


Figure 4.1: A flowchart visualizing the desired program

4.1 Program requirements

The project will now start the creation of a program, but to know how the program will work, there has to be made some requirements. As seen in figure 4.1, a flowchart has been designed, describing the process of the program that is expected. The project will now cover the flowchart in more detailed manner, to better understand the outline of the design.
The program should start with a login and sign up process:
When running the program, it required that there is an option to either sign in or sign up. Depending on what the user chooses, the program will then either find an already registered account or make one. If the choice to sign in is chosen, the program will ask the

user if he/she is a renter or an owner. An E-mail will then be required to search for the user. The program will search for the user respectively to what the user has chosen. If there is such an E-mail, the sign in is successful. If no such E-mail is found, the program will repeat the process, in consideration to spelling mistakes.
If the choice of signing up is chosen, the program will ask the user whether to register as a user or an owner, followed by a series of questions that will be stored in a data file. This will be one of the most important parts of the program, because if there are bugs and mistakes in the user registration and data, the rest of the program might not work as intended. As from here, the entire login and sign up process is defined.

The program will contain some User features, with a couple of options to be chosen from. This includes the option for the user to view their own profile, so that the user can see their ratings and check if their user data is as it should be. Another feature is an option to update user data, so that if the user spots a mistake or makes a change that the program should know about, the user has access to edit and update their information.

As for the initiating car rental process, an option will be available for the car renter specifically. If this option is selected, the list of all cars with necessary data included, will be displayed to the user. This list is to be sorted by type first and then afterwards with the price, in descending order.

Next up is a price algorithm. This algorithm should ensure a balanced price pool for a type of car, in respect to brand, quality, gear, condition (production date, km driven, overall cleanness) and the renter's situation.

One of the other requirements for the program is a rating system. This is because of user safety and quality. The purpose of the rating system is to partially ensure that owners and renters treat, maintain and clean the car properly, while also motivating owners and renters to provide a better experience for each other. The rating system will contain a formula to calculate the average rating. When a renter and an owner has made a deal, the owner and renter is able to rate each other afterwards. The data of the transaction is stored in a data file, containing different elements. Inside this data will also be an element that detects if the owner or renter has rated the deal, so that they can not rate the same person more than once per deal.

4.2 Program design

With a rough shape of the program visualized and the requirements for it now defined, it is time to define the program in a code based environment. The process will be roughly the same as in their requirements section, but the explanation of the execution will be explained using programming terms. It is therefore assumed that the reader knows these terms and what they consist of, as the explanation of these terms will take up way too much space and time. The coding will be in the programming language ANSI C, as this is a requirement from Aalborg University in this project. If other programming languages will be used in combination with ANSI C, this will be explained. The program design is very similar to the requirements section, as the program design simply adds some structure to these requirements.
All of the functions should be together in a separate header file, which is then included in the main program file. This list is an illustration of the program as a flowchart. The program should start of by asking if the user wants to sign in or sign up. If the user chooses to sign up, they are given the choice between signing up as a renter or as a car owner. Depending on the user's choice, the program will prompt for some information as seen in the list below. If the user chooses to sign in, they will be prompted for their email. Depending on whether the user is a renter or a owner, they will be met with two different menus containing different features. The renter can choose to rent a car, which jumps to the renting process, witch can also be seen in the list below. After a car has been rented both the renter and the owner will have the ability to rate each other.

Sign in

- Ask for email
- Prompts user to type in email
  - Renter
    - Menu
      - 1. Rent a car
      - 2. Show your information
      - 3. Edit your information
      - 4. Rate your last deal
  - Car owner
    - Menu
      - 1. Show latest rentals

· 2. Show account details
· 3. Edit account details
· 4. Rate your last deal

Sign up

- Sign up as renter
  - Enter name
  - Enter phone number
  - Enter E-mail
  - Enter age
  - Enter postcode
  - Enter preferred type of car (a: city car/hatchback, b: sedan/station car, c: SUV/Van)
  - Enter preferred transmission type (a = automatic, b = manual, c = both)
    - Account successfully created, and is stored in the database
    - User will have to run the program again in order to login as renter
- Sign up as owner
  - Enter name
  - Enter phone number
  - Enter E-mail
  - Enter age
  - Enter postcode
  - Enter how much your car should cost per hour
  - Enter name of your car
  - Enter car type (a = city car/hatchback, b: sedan/station car, c: SUV/-Van)
  - Enter your car's model year
  - Enter your car's mileage
  - Enter your car's transmission type (a = automatic, b = manual)
  - Enter small description of car
    - Account successfully created, and is stored in the database
    - User will have to run the program again in order to login as owner

Renting process

- List of all cars - sorted after type first, and in the types they are sorted after price
  - Every car has a number - program prompts you to choose a number
  - When you have chosen the car, you can see
    - Price of the car
    - Car name
    - Car type
    - Model year
    - Odometer
    - Transmission type
    - Car decription
    - Car is located in
    - It asks if you would like to rent the car
    - If yes:
      - It asks how many hours you would like to rent it
      - It tells you how much it will cost to the rent car for the amount of hours
      - You now rent the car
    - If no:
      - Then it goes back and displays the cars in the sorted list

Chapter 5

Program Implementation

Now that we have taken a look on the programs requirements and the programs design, we will take a look at how we implemented these requirements and designs into code.
The most vital parts of the program will be discussed and explained, it will start off by discussing and explaining the vital functions in CarRentalFunctions.h, afterwards it will discuss and explain how these functions are used and how the whole program works in the file main.c, and at last a discussion will take place about the parts and features that the group wanted to implement, but did not either have time or the knowledge to do so. The following code will be attached as files, and in the caption for each listing of code there will be a number between 1 and 778, besides that the file name is written.
The variables in the program will be written in *italics*. It is possible to find "//" in our code, these two forwards slashes is seen as a comment by the compiler, that translates the code from C language to machine language that the computer understands, and therefore will not be run in the program. A .h file is a header file, this is to be seen as a library that the main program can read and use. In our case the main file gets all of the functions that we have made from the CarRentalFunctions.h file. This way we can make the main file clean and more easy to read. A .c file is a source code file written in C programming language. In our case the main.c file is the file that runs the whole program, code wise.
The program is built upon different structures, where each structure contains different elements associated with their type. An essential part of the program solution is the handling of data in these structures and therefore requires some sort of database system to store this data. This can be done a various of ways, but it was concluded that the best for this size of program would be storing in some sort of data file.
The data can either be stored as plain text or written to the file as binary. The binary solution seems most appealing to us as it can write each structure as a stream

of binary in which we know and control the size. This makes the writing and reading of binary files much easier as the size of each type of structure is the same. We chose to store this data in .dat files, which is simply a file extension that can be used for all kinds of purposes. The disadvantage of using this method is that the files are unreadable for humans as the files simply contain binary streams. This means checking the files for data must happen through the program itself. This is however not that big of a problem for the program as it should contain a way of showing a users information anyway.
The program has three different binary "database" files, one for each type of structure. This is done to keep the different types separated and grants a lot easier access to what is being searched for. For example if a user is searching for information about a renter, the program knows exactly which file to find it in, instead of having to search through one big data file containing different types of structures, also meaning different sizes of data.

5.1 Structures

At the very beginning of CarRentalFunctions.h (see attached files) the programs structures are located. It is located from line 2 to 52. These structures are used throughout the whole program, and is used to store different information and variables into the specific car owner, car renter or to store the different parts of transaction data.

This function will from now on be referenced with the line numbers in listing 5.1. Below we have 3 structures, one called *transactionData*, another one called *carOwnerData* and the last one called *carRenterData*. A structure is a user defined data type and a structure creates a data type that can be used to group items of possibly different types into a single type. In our situation these structures are very important and are the backbone of the whole program data wise. In these structures there are 4 different data types. There is integer, character, double and boolean.

transactionData:
The *transactionData* structure contains all of the valuable information that could be connected to a transaction. The structure use integers, characters and booleans. The transaction ID that is an ID number that is linked to every transaction there is, *renterID* is the individual ID number for the renter, *ownerID* is the individual ID number for the owner these are all integers. Then we have *renterName*, *ownerName*, *renterEmail*, *ownerEmail* which stores the name and email for either the renter or owner, at the last character type we have is the *carName* which stores the name of the car. Then we have the last two boolean data types which are *ownerRated* and *renterRated*, they both store a true or false if the owner/renter has been rated or not.

carOwnerData:
The *carOwnerData* structure contains all of the valuable information that could be connected to the car owner. This structure use integers, characters and a double data type. The integers are ID that contains the car owners ID number, the *age* of the car owner, the *postcode* of the car owner, the *price* of the car, the model year of the car, *odometer* and the rating amount. When taking a look at characters we have *name* of the car owner, we have the *phonenumber* of the car owner, the email of the car owner, the name of the car, which type of car they have, the *transmission* of the car and at last a short description of the car. There is one double data type left and that is the *rating* of the car owner.

carRenterData:
The *carRenterData* structure contains all of the valuable information that could be connected to the car owner. Most the of the information is the same as the *carOwnerData* structure, though we have *prefCarType* that stores the preferred car type of the renter and we also have *prefTransmissionType* that stores which preferred transmission type they would like to use, instead of the information about the car.

```
typedef struct transactionData
{
  int transactionID;    // An ID number that is linked to every
    transaction
  int renterID;         // Individual ID number for the renter
  char renterName[50];  // Renters name
  char renterEmail[50]; // Renters email
  int ownerID;          // Individual ID number for the owner
  char ownerName[50];   // Owners name
  char ownerEmail[50];  // Owners email
  char carName[50];     // Name of the car
  bool ownerRated;      // Boolean data type on whether the owner has
    rated
  bool renterRated;     // Boolean data type on whether the renter has
    rated
} tranDet;

typedef struct carOwnerData
{
  // values set by user
  int ID;            // Car owners ID number
  char name[50];     // Name of Car Owner
  char phoneNum[9];  // Phone number of CO
  char Email[50];    // Email of CO
  int age;           // age of CO
  int postCode;      // postcode of CO
  int price;         // Price / h (dkk) of CO
```

```
25   char carName[50];  // Name of CO's car
26   char carType;      // Car owners type of car (a = city car/hatchback
       , b = sedan/station car, c = SUV/Van)
27   int modelYear;     // Model year of CO's car
28   int odometer;      // kilometers driven by CO's car
29   char transmission; // CO's car's transmission | a = auto, b = manual
30   char CarDescription[100];
31   double rating;
32   int ratingAmount;
33   // values set by program
34   // double  rating[]; // rating of CO
35 } carOwner;
36
37 typedef struct carRenterData
38 {
39   int ID;                    // Car renters ID number
40   char name[50];             // name of Car Renter
41   char phoneNum[9];          // Phone number of CR
42   char Email[50];            // Email of CR
43   int age;                   // Age of CR
44   int postCode;              // Postcode of CR
45   char prefCarType;          // CR's preffered car type | (a = city
       car/hatchback, b = sedan/station car, c = SUV/Van)
46   char prefTransmissionType; // CR's Preferred transmission type | can
       be a,b or c, a = auto, b = manual, c = both
47   double rating;
48   int ratingAmount;
49   // values set by program
50   // double  rating[]; // rating of CR
51 } carRenter;
```

Listing 5.1: Program structs - 2-52 - CarRentalFunctions.h

5.2 Enter Car Owner

One of the first things through the flowchart that the user meets are this function enterCarOwner, this function is in CarRentalFunctions.h (see attached files). It is located from line 131 to 186. If we take a look at our renting process and our flowchart, this function is the one that when you choose you want to sign up, and then afterwards choose "2. Sign up as owner". The program also contains another function called enterCarRenter that gets the information for the renter. The report will only cover the enterCarOwner function since it is the most comprehensive of the two, and on the basics they are very similar.

This function will from now on be referenced with the line numbers in listing 5.2. The function starts off by using the *carOwner* structure and changing its reference to *tempCarOwner*. In line 6 the *ID* for *tempCarOwner* gets randomized with large

numbers, that way the program makes sure that the chances of the owners or renters getting the same number is very low. From line 7 to 16 it saves the basic information as name, phone number, email and age gets stored in the *tempCarOwner* structure. A do-while loop gets started from line 18 to 21, this makes sure that whenever the user types in their postcode it only accepts the postal codes called 9000, 9200, 9210 and 9220. The reason why the program does this is since it is only meant to be used in Aalborg at first. Line 23 to 30 is the part where the user types in how much their car should cost per hour. If the user enters that it should cost 95kr the program continues to the next part, but as aforementioned the maximum hourly renting price is 100kr, so with the do-while loop from line 26 to 30 we make sure that the price is beneath 100kr. The user gets prompted to enter the name of the car and afterwards prompted to enter the car type, here is again a do-while loop from line 36 to 40, since the program wants to ensure that the user only types in either a, b or c. The last part of the function is basic information gathering from line 41 to 51, where the program asks the user for the car's model year, mileage, transmission type and a small written description about the car containing less that 100 characters. Variable *tempCarOwner.rating* and *tempCarOwner.ratingAmount* both gets assigned the value 0, since both of these variables should be equal to 0 when creating a new user.
Once all of these information's has been entered and the function is done, it saves the information stored in the *tempCarOwner* structure to a file called Owner.dat, which is a database file containing every single owner in the program.

```
1  carOwner enterCarOwner(void)
2  {
3    carOwner tempCarOwner;
4    char name[50];
5
6    tempCarOwner.ID = ((rand() % 1000000) + 100000);
7    printf("Enter name: ");
8    fgets(tempCarOwner.name, 50, stdin);
9    getName(name);
10   strcpy(tempCarOwner.name, name);
11   printf("Enter phone number: ");
12   scanf("%s", &tempCarOwner.phoneNum);
13   printf("Enter E-mail: ");
14   scanf("%s", &tempCarOwner.Email);
15   printf("Enter age: ");
16   scanf("%d", &tempCarOwner.age);
17   printf("Enter postcode: ");
18   do
19   {
20     scanf("\n%d", &tempCarOwner.postCode);
21   } while (tempCarOwner.postCode != 9000 && tempCarOwner.postCode !=
       9200 && tempCarOwner.postCode != 9210 && tempCarOwner.postCode !=
```

```
    9220);
22
23  printf("Enter how much your car should cost per hour: ");
24  scanf(" %d", &tempCarOwner.price);
25
26  while (tempCarOwner.price > 100)
27  {
28    printf("\nEnter a maximum of 100 dkk: ");
29    scanf(" %d", &tempCarOwner.price);
30  }
31  printf("Enter name of your car: ");
32  fgets(tempCarOwner.carName, 50, stdin);
33  getName(name);
34  strcpy(tempCarOwner.carName, name);
35  printf("Enter car type (a = city car/hatchback, b = sedan/station
      car, c = SUV/Van): ");
36  do
37  {
38    printf("\na, b or c: ");
39    tempCarOwner.carType = getchar();
40  } while (tempCarOwner.carType != 'a' && tempCarOwner.carType != 'b'
      && tempCarOwner.carType != 'c');
41  printf("Enter your car's model year: ");
42  scanf("%d", &tempCarOwner.modelYear);
43  printf("Enter your car's mileage: ");
44  scanf("%d", &tempCarOwner.odometer);
45  printf("Enter your car's transmission type (a = automatic, b =
      manual): ");
46  fflush(stdin);
47  scanf("%c", &tempCarOwner.transmission);
48  printf("Enter small description of car (<100 characters): ");
49  fgets(tempCarOwner.CarDescription, 50, stdin);
50  getName(name);
51  strcpy(tempCarOwner.CarDescription, name);
52  tempCarOwner.rating = 0;
53  tempCarOwner.ratingAmount = 0;
54
55  return tempCarOwner;
56 }
```

Listing 5.2: enterCarOwner - 131-186 - CarRentalFunctions.h

5.3 User select

The user select function is very important to the program as it performs a few crucial operations for our program to function as intended. The function is located from line 212 to 276 in the CarRentalFunctions.h file, and can be found in listing 5.4. The main feature of the function is to work as a login system. It does this by

searching through the user files and returning whether the user is a renter or a car owner.

This function will from now on be referenced with the line numbers in listing 5.4. From line 3 to 6 the function initializes the different variables needed for the performance of the operations in the function. From line 9 to 21 the function opens both the renter.dat and the owners.dat in read mode with the file pointers. It then checks if the files were opened successfully, and if the files for some reason could not be opened it gives an error message and exits the program.
The code from line 23 to 39 repeats on line 40 to 56 just with different variables and parameters. The two while loops reads through renters.dat and owners.dat respectively and for each loop stores the respective struct in the variables *tempCarRenter* or *tempCarOwner*. It then compares the email entered by the user and the email in the struct on line 30 and 47 respectively. If the emails match, one of the global struct variables *carRenter1* or *carOwner1* seen in 5.3 is set to be the struct containing that email. This way one of these two structs now contain the user signed in and can be used as such for the rest of the program. One of the two global boolean variables *RenterLoggedIn* or *OwnerLoggedIn* gets set to 1 depending on, if the user logged in is a renter or a owner.

```
1 carRenter carRenter1;
2 carOwner carOwner1;
3 bool RenterLoggedIn = 0;
4 bool OwnerLoggedIn = 0;
```

Listing 5.3: Global Variables - 55-58 - CarRentalFunctions.h

The while loop reads through the entire file and then breaks out of the loop at the end of the file, which can be seen on line 26 to 29. If the email entered by the user can not be found in both files the function will return the message "Sorry no record found" as seen on line 59. The two files are then finally closed on line 62 and 63 to update the files.

```
1  int userSelect(char Email[])
2  {
3    FILE *fp, *fp1;
4    carRenter tempCarRenter;
5    carOwner tempCarOwner;
6    int found = 0;
7
8    // Open renters.dat for reading
9    fp = fopen("renters.dat", "rb");
10   if (fp == NULL)
11   {
12     fprintf(stderr, "\nError opening file\n");
```

```
    exit(1);
  }
  // Open owners.dat for reading
  fp1 = fopen("owners.dat", "rb");
  if (fp1 == NULL)
  {
    fprintf(stderr, "\nError opening file\n");
    exit(1);
  }
  // read file contents till end of file
  while (1)
  {
    fread(&tempCarRenter, sizeof(tempCarRenter), 1, fp);
    if (feof(fp))
    {
      break;
    }
    if (!strcmp(tempCarRenter.Email, Email))
    {
      found = 1;
      RenterLoggedIn = 1;
      carRenter1 = tempCarRenter;
      fclose(fp);
      fclose(fp1);
      return 1;
    }
  }
  while (1)
  {
    fread(&tempCarOwner, sizeof(tempCarOwner), 1, fp1);
    if (feof(fp1))
    {
      break;
    }
    if (!strcmp(tempCarOwner.Email, Email))
    {
      found = 1;
      OwnerLoggedIn = 1;
      carOwner1 = tempCarOwner;
      fclose(fp);
      fclose(fp1);
      return 0;
    }
  }
  if (found == 0)
  {
    printf("Sorry no record found\n");
  }
  // close file
  fclose(fp);
  fclose(fp1);
```

```
64   return -1;
65 }
```

Listing 5.4: userSelect - 212-276 - CarRentalFunctions.h

5.4 Edit profile

The program contains two functions for editing profiles, one function to edit a renter and another to edit a car owner. The functions are practically similar except for some variable names and which files are getting read and written to. Therefore only one of the functions are going to be highlighted and explained. We will focus on the function that edits the owner and this function can be seen in listing 5.5.

This function will from now on be referenced with the line numbers in listing 5.5. The start of the function initializes two file pointers and a temporary struct *tempCarOwner* to contain the structs getting read in the file. Line 6 opens the owners.dat file in read mode and line 7 opens a temp.dat file in write mode. This temporary data file will be used to copy the contents of owners.dat. From line 8 to 25 a while loop runs through the entire file till end of the file controlled by the if statement on line 11. Through each iteration of the loop the structs in owners.dat are stored in the *tempCarOwner*, and the email in this struct is compared with the email of the user logged in. If the email matches the user is asked to enter the new information through the function enterCarOwner, which has been described earlier and can be seen in listing 5.2. This new information is then written to the temp.dat file as seen on line 19. If the emails do not match it will simply copy over the existing structs from owners.dat to temp.dat seen on line 23.
The temp.dat file now contains all of the structs including the new edited struct. The owners.dat is now opened in write mode while the temp.dat file is opened in read mode on line 34 and 35. The while loop from 36 to 45 now copies everything from temp.dat back into owners.dat by reading each struct on line 38 and writing them on line 43. Line 46 and 47 finally closes the files and they are then updated with the new information.

```
1  carOwner carOwnerEdit(void)
2  {
3    FILE *fp, *fp1;
4    carOwner tempCarOwner;
5    int found = 0;
6    fp = fopen("owners.dat", "rb");
7    fp1 = fopen("temp.dat", "wb");
8    while (1)
9    {
10     fread(&tempCarOwner, sizeof(tempCarOwner), 1, fp);
```

```
    if (feof(fp))
    {
      break;
    }
    if (!strcmp(tempCarOwner.Email, carOwner1.Email))
    {
      found = 1;
      tempCarOwner = enterCarOwner();
      fwrite(&tempCarOwner, sizeof(tempCarOwner), 1, fp1);
    }
    else
    {
      fwrite(&tempCarOwner, sizeof(tempCarOwner), 1, fp1);
    }
  }
  fclose(fp);
  fclose(fp1);
  if (found == 0)
  {
    printf("Sorry, no record found");
  }
  else
  {
    fp = fopen("owners.dat", "wb");
    fp1 = fopen("temp.dat", "rb");
    while (1)
    {
      fread(&tempCarOwner, sizeof(tempCarOwner), 1, fp1);
      if (feof(fp1))
      {
        break;
      }
      fwrite(&tempCarOwner, sizeof(tempCarOwner), 1, fp);
    }
  }
  fclose(fp);
  fclose(fp1);
}
```

Listing 5.5: carOwnerEdit - 341-388 - CarRentalFunctions.h

5.5 Car select

Car Select is one of the vital function in CarRentalFunctions.h (see attached files). It is located from line 797 to 849. If we take a look at our renting process and our flowchart, this function is the one that prints out the list, and enables the user to pick a car, calculate the price and finally rent it.

This function will from now on be referenced with the line numbers in listing 5.6. From line 1 to 22 the function opens the file called owners.dat, makes sure with an if statement that the file can be opened. If it opens successfully the program reads everything in owners.dat file into an array of structures called *arrCars[i]*. This means that every car has its own structure, with the individual information from all the owners in owners.dat. This enables us to have each car for itself in a structure that we can sort, and we can also print every single car on its own.
In line 23 we use qsort to sort the array of structures called *arrCars[]*, where we also call the function compare_type_and_price that is in listing 5.7 which will be spoken about later.
From line 24 to 48 the program starts a do-while loop. This do-while loop continues as long as *rent_car* is not equal to 1. The first thing the do-while loop does is printing all the cars in a for loop. It prints a number for every car so it can be identified and the number is increased by 1 since the first car in the array of structures is *arrCars[0]*, it also prints the individual cars price, car name, car type, model year, odometer, transmission type, a description of the car and where it is located. The for loop increases *i* every loop until *i* is equal to *number_of_cars*. This enables the program to enter *i* in *arrCars[i]* to go through the different cars in the array of structures.
Once all the cars has been printed and the for-loop is done, it prompts to user to answer which car they would like. The user can type any of the numbers corresponding to the number that is printed over the car that they would like. Their answer is stored in *&choice* and afterwards *choice* is decreased by one again so it corresponds to the correct car in the array of structures, which enables the program to print the chosen car with *arrCars[choice]*. The program asks the user if they want to rent the car, where 1 is = Yes and 2 is = No, this answer is saved under the variable named *rent_car*. This is where the do-while loop comes in to play, if the answer is 2, the while condition: (*rent_car* != 1) is true, and therefore begins the do-while loop from the start again. Though if the answer to *rent_car* is equal to 1, it asks the user how many hours they would like to rent the car. The user enters how many hours they would like to rent it, it tells you that you successfully chose the car, and it then tells you how much it will cost to rent the car for the amount of hours you chose. After it has printed this, two functions will run, makeDeal(*arrCars[choice]*) and findTransaction(). These two functions are explained in listing 5.8 and in listing 5.9.

```
1 void carSelect(carOwner arrCars[])
2 {
3   FILE *fp;
4   carOwner tempCarOwner;
5   int i = 0, number_of_cars, choice, rent_car, number_of_hours;
6   fp = fopen("owners.dat", "rb");
```

```
7   if (fp == NULL)
8   {
9     fprintf(stderr, "\nError opening file\n");
10    exit(1);
11  }
12  while (1)
13  {
14    fread(&tempCarOwner, sizeof(tempCarOwner), 1, fp);
15    if (feof(fp))
16    {
17      number_of_cars = i;
18      break;
19    }
20    arrCars[i] = tempCarOwner;
21    i++;
22  }
23  qsort(arrCars, number_of_cars, sizeof(carOwner),
      compare_type_and_price);
24  do
25  {
26    for (i = 0; i < number_of_cars; i++)
27    {
28      printf("\n%d.", i + 1);
29      printf("\nPrice: %d\nCar name: %s\nCar type: %c\nModel year: %d\
    nOdometer: %d\nTransmission type: %c\nCar description: %s\nCar is
    located in: %d\n",
30              arrCars[i].price, arrCars[i].carName, arrCars[i].carType,
     arrCars[i].modelYear, arrCars[i].odometer, arrCars[i].transmission
    , arrCars[i].CarDescription, arrCars[i].postCode);
31    }
32    printf("\nChoose which car you would like to rent: ");
33    scanf(" %d", &choice);
34    choice--;
35    printf("\nPrice: %d\nCar name: %s\nCar type: %c\nModel year: %d\
    nOdometer: %d\nTransmission type: %c\nCar description: %s\nCar is
    located in: %d\n",
36           arrCars[choice].price, arrCars[choice].carName, arrCars[
    choice].carType, arrCars[choice].modelYear, arrCars[choice].
    odometer, arrCars[choice].transmission, arrCars[choice].
    CarDescription, arrCars[choice].postCode);
37    printf("\nWould you like to rent this car? (1 = Yes, 2 = No): ");
38    scanf(" %d", &rent_car);
39
40    if (rent_car == 1)
41    {
42      printf("\nHow many hours would you like to rent this car? ");
43      scanf(" %d", &number_of_hours);
44      printf("You successfully chose %s\n", arrCars[choice].carName);
45      printf("It will cost you %d dkk to rent this car for %d hours\n"
    , arrCars[choice].price * number_of_hours, number_of_hours);
46      makeDeal(arrCars[choice]);
```

```
47        findTransaction();
48      }
49    } while (rent_car != 1);
50
51    fclose(fp);
52  }
```

Listing 5.6: carSelect - 797-849 - CarRentalFunctions.h

What did we want to implement?

To the function carSelect we wanted to implement a feature, before the list where all of the cars gets printed. This feature should have given the user an opportunity to sort the cars in different ways. As you know when signing up, the program asks for a lot of different information, as a renter you get asked what your preferred type of car is, and what your preferred transmission type is. This is very useful information when wanting to sort, and in this case, we wanted to implement so that the user could either sort for price, type, their preferred type of car or their preferred transmission type. One of the challenges we had with this is when choosing which car you would like to rent, the *arrCars[choice]*, the array would be different from the owners side, than the renters. There would also be an issue with sorting the cars in a way that the user chooses, as qsort sorts the array of structs called *arrCars[i]*, and this *arrCars* array of structures contain all of the cars, not just a couple of them, if you would sort for a specific type as an example. If we wanted to sort for a specific type, we would need a whole new array of structures with those cars in them. This we did not have time to do, and would take up a lot of space in the program.

5.6 Compare type and price

compare_type_and_price is the sorting algorithm from CarRentalFunctions.h (see attached files). It is located from line 436 to 454. This function is directly and only used in the function carSelect, that is explained in listing 5.6.

This function will from now on be referenced with the line numbers in listing 5.7. The function compare_type_and_price is used in the call of qsort on line 21 in listing 5.6, the function compares two array elements and returns an integer value specifying their relationship. The call to qsort calls this function one or more times during the sort, passing pointers to two array elements on each call. The compare_type_and_price function compares the elements and return one of the following values: -1 if p1 *carType* is less than p2 *carType* or p1 *price* is less than p2 *price*, 1 if p1 *carType* is greater than p2 *carType* or p1 *price* is greater than p2 *price*,

as we can see from line 5 to 19. If two elements are equal, their order in the sorted array is unspecified, and therefore the qsort does not do anything with these two elements and continues with the next two elements.

This sorts the elements in the array in increasing order, first by their car type, and then afterwards in their price. This is essentially what gives the do-while loop in listing 5.6 from line 24-48 the ability to print out a list that is at first sorted after the car types, and in the respective car types they are sorted after price.

```
1  int compare_type_and_price(const void *v1, const void *v2)
2  {
3    const carOwner *p1 = (carOwner *)v1;
4    const carOwner *p2 = (carOwner *)v2;
5    if (p1->carType < p2->carType)
6    {
7      return -1;
8    }
9    else if (p1->carType > p2->carType)
10   {
11     return 1;
12   }
13   else if (p1->price < p2->price)
14     return -1;
15   else if (p1->price > p2->price)
16     return 1;
17   else
18     return 0;
19 }
```

Listing 5.7: compare_type_and_price - 436-454 - CarRentalFunctions.h

What did we want to implement?

One of the things we wanted to implement as mentioned in the section about carSelect was the different sorting methods. They would have looked about the same as this one in listing 5.7, but with different parameters.

5.7 Make deal

This function has the purpose of establishing a connection between the renter and the owner. The function is located on line 390 to 434 in the carRentalFunctions.h header files (see attached files).

This function will from now on be referenced with the line numbers in listing 5.8. First off, a temporary *tranDet* variable is initiated, which is from the *tranDet* struc-

ture. The functions starts with assigning all values in the Transaction Data structure to the desired value. The first one being the *ID*, which is done by using the rand() function from the c library. The number is chosen between 1.000.000 and 1.000.000.000. Thereafter, the logged in car renter's ID is assigned to the *renterID* in the variable, followed by the car owner the renter chose, which is a part of the function parameter. The structure also contains two booleans, as seen in listing 5.8, the name of these two are *renterRated* and *ownerRated*. Both are assigned to 0 which equals false, because none of the two has rated the deal in the instant it is established. Next up is the strcpy function, which copies a string into another. The car Renter, the car owner and the name of the car are all copied into the *tempTrans* variable, each at their specified locations. The now completed *tempTrans* variable is stored in its own data file. The file is opened on line 23 in listing 5.8, followed by a check, that determines if the file managed to open. If it fails to open the file, the program stops. If the file is successfully opened, the data is written into the end of the file. This is because of the "ab" on line 23, which means append mode in a binary file.

```
1  void makeDeal(carOwner tempCarOwner)
2  {
3
4    FILE *fp;
5    tranDet tempTrans;
6
7    // ID's
8    tempTrans.transactionID = ((rand() % 1000000) + 100000);
9    tempTrans.renterID = carRenter1.ID;
10   tempTrans.ownerID = tempCarOwner.ID;
11
12   tempTrans.renterRated = 0;
13   tempTrans.ownerRated = 0;
14   // emails
15   strcpy(tempTrans.renterEmail, carRenter1.Email);
16   strcpy(tempTrans.ownerEmail, tempCarOwner.Email);
17   // names
18   strcpy(tempTrans.renterName, carRenter1.name);
19   strcpy(tempTrans.ownerName, tempCarOwner.name);
20   strcpy(tempTrans.carName, tempCarOwner.carName);
21
22   // open file
23   fp = fopen("transactions.dat", "ab");
24
25   if (fp == NULL)
26   {
27     fprintf(stderr, "\nError opened file\n");
28     exit(1);
29   }
30
```

```
31    fwrite(&tempTrans, sizeof(tempTrans), 1, fp);
32
33    if (fwrite != 0)
34    {
35      printf("Transaction completed\n");
36    }
37
38    else
39    {
40      printf("Error writing to file !\n");
41    }
42
43    // close file
44    fclose(fp);
45  }
```

Listing 5.8: makeDeal - 390-434 - CarRentalFunctions.h

What did we want to implement?

For this function, we wanted to add some more details to the transaction system. This would be the date that the deal was created and a receipt for both parts containing proof that the transaction took place. This is because of user safety and insurance policies.

5.8 Find transaction

This function finds the latest unrated transaction for a user, enabling it for rating. The function is located on line 776 to 864 in the carRentalFunctions.h header file(see attached files). 5.6.

This function will from now on be referenced with the line numbers in listing 5.9. This function first off initiates several different variables that will come into play. The function checks if a renter or an owner is logged in, and makes this value a local value, as global variables are a lot more exposed to unforeseen issues. Since the transaction of the logged in user is to be found, the transaction data file is opened, followed by an open-success check. As of line 35 in listing 5.9 a while-loop is created. This loop runs until the end of the transaction data file is reached. At line 39 an if statement checks whether the *ans* char variable is 'a' or not, as well as if the logged in renter's Email fits a renter E-mail in the the transaction data file. The last check confirms if the transaction has already been rated by the renter, and if all these conditions are met, the transaction is saved in a local variable and the found variable is set to 1. The underneath else if statement on line 51 does the same as the above, except that the statement is meant for a logged in car owner instead. If

no deal is found, also expressed if (*found* == 0) on line 59, a message will appear concluding that no unrated deal was found. On line 67, another if statement is created, checking if any unrated transaction was found. If there was, two if statements on line 71 and 76 will be checked for. These are as seen before, a chekc for whether it is a renter or an owner that is logged in. The statements will then print out the counter part information in the transaction meaning that if the renter is looking to rate a deal, the owner will be displayed, as for the owner vice versa. A question will be asked the user, if they would like to rate the displayed user, followed by a scanf on line 84, storing the answer in the *ans2* char variable. The function now enters a do-while loop on line 81 to 102, checking again for which user is logged in and whether they wished to rate the other user or not, if an answer different from 'y' or 'n' is entered, the functions ask for another answer. This is in consideration to pressing the wrong key.

```
1  void findTransaction(void)
2  {
3
4    carOwner dealOwner;
5    carOwner tempOwner;
6    carRenter dealRenter;
7    carRenter tempRenter;
8    FILE *fp;
9    tranDet tempTrans;
10   tranDet foundTrans;
11   bool found;
12   char ans;
13   char ans2;
14
15   if (RenterLoggedIn == 1)
16   {
17     ans = 'a';
18   }
19
20   else if (OwnerLoggedIn == 1)
21   {
22     ans = 'b';
23   }
24
25   fp = fopen("transactions.dat", "rb");
26
27   if (fp == NULL)
28   {
29     fprintf(stderr, "\nError opening file\n");
30     exit(1);
31   }
32
33   // read file contents till end of file
```

```
34  while (1)
35  {
36
37    fread(&tempTrans, sizeof(tempTrans), 1, fp);
38
39    if (feof(fp))
40    {
41      break;
42    }
43
44    if (ans == 'a' && (!strcmp(tempTrans.renterEmail, carRenter1.Email
      )) && tempTrans.renterRated == 0)
45    {
46
47      found = 1;
48      foundTrans = tempTrans;
49    }
50
51    else if (ans == 'b' && (!strcmp(tempTrans.ownerEmail, carOwner1.
      Email)) && tempTrans.ownerRated == 0)
52    {
53
54      found = 1;
55      foundTrans = tempTrans;
56    }
57  }
58
59  if (found == 0)
60  {
61    printf("\nNo unrated deals found");
62  }
63
64  // close filef
65  fclose(fp);
66
67  if (found == 1)
68  {
69
70    fflush(stdin);
71    if (ans == 'a')
72    {
73      printf("\nYou have made a deal with the transaction details:\nID
      : %d\nCounter Part Name: %s\nCounter Part Email: %s\nCar name: %s\
      nDo you wish to rate this deal?\ny: yes\nn: no\n",
74              foundTrans.transactionID, foundTrans.ownerName,
      foundTrans.ownerEmail, foundTrans.carName);
75    }
76    else if (ans == 'b')
77    {
78      printf("\nYou have made a deal with the transaction details:\nID
      : %d\nCounter Part Name: %s\nCounter Part Email: %s\nDo you wish to
```

```
    rate this deal?\ny: yes\nn: no\n",
79              foundTrans.transactionID, foundTrans.renterName,
    foundTrans.renterEmail);
80    }
81    do
82    {
83
84      scanf("%c", &ans2);
85      printf("\n");
86
87      if (ans == 'a' && ans2 == 'y')
88      {
89        userRating(foundTrans.ownerEmail, ans, foundTrans);
90      }
91
92      else if (ans == 'b' && ans2 == 'y')
93      {
94        userRating(foundTrans.renterEmail, ans, foundTrans);
95      }
96
97      else if (ans2 == 'n')
98      {
99        exit(1);
100     }
101
102   } while (ans2 != 'y' && ans2 != 'n');
103 }
104 }
```

Listing 5.9: findTransaction - 692-795 - CarRentalFunctions.h

5.9 Main.c

Now that all of the functions has been explained and discussed, we know need to know what they are being used for. Everything that happens when running the program is situated in the file main.c.

Lets take a look at the first part of main.c. This is where the rand() function from the c library gets seeded in line 8. It is seeded in the time function from the time.h header file, because the seed will change every second when using this. Otherwise the same sequence of pseudo random numbers will generate. The first thing you will meet when opening the program is the choice between "1. Sign in" or "2. Sign up". A do-while loop starts in line 12 to make sure that it gets an answer that is either 1 or 2. We will start off by covering the situation that you entered "1 = Sign in", so let's say that you have an account. When you enter "1. Sign in" the if statement in line 17 is true and you enter it. A do-while loop start from line 20 to line 26 that prompts you the enter your email. If the function userSelect see listing 5.4, which searches for your account with the email by first searching in the renter database and then in the owners database, comes back with the variable *isRenter* as one, it means that the email found is a renter, if it comes back as 0 it means that the email found is an owner and continues through the program. If it cannot find the email it prompts you to enter it again.

```
1  int main(void)
2  {
3    int answer = 0;
4    int isRenter = -1;
5    char Email[50];
6    carOwner arrCars[10];
7
8    srand(time(NULL));
9
10   printf("1. Sign in\n2. Sign up\n");
11
12   do
13   {
14     scanf("%d", &answer);
15   } while (answer < 1 && answer > 2);
16
17   if (answer == 1)
18   {
19
20     do
21     {
22       printf("Please enter your email: ");
23       scanf("%s", &Email);
```

```
24
25      isRenter = userSelect(Email);
26    } while (isRenter < 0);
```

Listing 5.10: Getting email - 8-33 - main.c

Renter

After you have entered the email and it has found you as a renter which means that *isRenter* is equal to 1, you get a welcome back message that also displays your name. You are now getting presented with 4 different choices. First one is if you want to rent a car, second is if you want to display your profile so you can see all the information that is stored on your account, third is if you want to edit this information that is stored on your profile and fourth choice is if you want to exit out of the program.
These choices are made possible by a switch case from line 12 to 31, and takes *answer* as an input. If the *answer* is "1" it calls the function carSelect in listing 5.6. If the *answer* is "2" the case calls carRenterDisplay which is a function that just prints out your information stored in Renters.dat. If the *answer* is "3" it calls carRenterEdit this gives the user an option to edit their information, this is as aforementioned almost the same as the function discussed in listing 5.5. If the *answer* is "4" it returns "EXIT_SUCCESS" and quits the program.
After every case it breaks out of the switch case and closes the program.

```
1  if (isRenter == 1)
2      {
3        printf("Welcome back %s!\n", carRenter1.name);
4        while (1)
5        {
6          printf("\n1. Rent a car\n2. View your profile\n3. Edit your
       profile\n4. Exit\n");
7          do
8          {
9            scanf("%d", &answer);
10         } while (answer < 1 && answer > 4);
11
12         switch (answer)
13         {
14         case 1:
15           carSelect(arrCars);
16           break;
17
18         case 2:
19           carRenterDisplay();
20           break;
21
```

```
22          case 3:
23            carRenterEdit();
24            break;
25
26          case 4:
27            return EXIT_SUCCESS;
28
29          default:
30            break;
31          }
32        }
33    }
```

Listing 5.11: Renter - 35-67 - main.c

Owner

If *isRenter* comes back equal to 0 it means that you are an owner. The program welcomes you back and you get a list containing your 5 next options. The first one enables you to see your past rentals, this gives you an overview of the transaction ID, name and email on the renter that rented your car. Second one enables you to view your profile. This just displays your profile information. Third choice is to edit your profile, this is exactly the same as the process of setting up a new one, you just keep your ID and gets the opportunity to change everything else on the profile. Fourth choice is to rate your latest rental, whenever a renter rents your car you get the opportunity to rate the person. Fifth case is just to exit out of the program. These five opportunities are encased in a switch case just like the renter part, where case 1 calls show_rentals, case 2 calls carOwnerDisplay, case 3 calls carOwnerEdit, case 4 calls findTransaction and the last case exits the program.

```
1  else if (isRenter == 0)
2      {
3        printf("Welcome back %s!\n", carOwner1.name);
4
5        while (1)
6        {
7          printf("\n1. View your car rental history\n2. View your
       profile\n3. Edit your profile\n4. Rate your latest Rental\n5. Exit\
       n");
8          do
9          {
10           scanf("%d", &answer);
11         } while (answer < 1 && answer > 5);
12
13         switch (answer)
14         {
15         case 1:
```

```
          show_rentals();
          break;

        case 2:
          carOwnerDisplay();
          break;

        case 3:
          carOwnerEdit();
          break;

        case 4:
          findTransaction();
          break;

        case 5:
          return EXIT_SUCCESS;

        default:
          break;
        }
      }
    }
```

Listing 5.12: Owner - 68-105 - main.c

Sign up

If the answer is "2. Sign up" and you want to create a new account, then you will be met with two options, that is if you would like to sign up as a renter or an owner, a do-while loop makes sure that you either enter 1 or 2. If the answer is "1. Sign up as a renter" the function carRenterData gets called and ran. If the answer is "2. Sign up as owner" then the carOwnerData will be called, and in carOwnerData the function spoken about earlier is called in listing 5.2.

```
  else if (answer == 2)
  {
    printf("1. Sign up as renter\n2. Sign up as owner\n");
    do
    {
      scanf("%d", &answer);
    } while (answer < 1 && answer > 2);

    if (answer == 1)
    {
      carRenterData();
    }
    else if (answer == 2)
```

```
14      {
15        carOwnerData();
16      }
17    }
18    return 0;
19  }
```

Listing 5.13: Sign up - 107-125 - main.c

Main.c

This is the whole program that is run from start to end. The main.c file acts as the director of the whole program, this is where the user enters the decisions and the flow of the program is determined.

5.10 Testing

Testing of our program was done while we programmed it. This was done to prevent having a wrongly made code that all group members was coding on. If this was the case that a wrongly made code ended up on our individual computers, everyone would be adding to the wrongly made code, and the risk of having all the code being filled with bugs if very large.
The next part will touch on the different kinds of testing we did throughout the process of coding the program, and also when the code was all done.

Input validation

Input validation is a used throughout the functions where the user inputs any information. In the 3 listings below, 5.14, 5.15, 5.16, we take use of an input-validation loop. This means that a do-while loop keeps looping as long as the expected answer is not true. In listing 5.14 we expect the user to type in either a, b or c to choose the type of car they have. A do-while loop starts in line 2, prints "a, b or c:" and waits for the user to respond with getchar(). getchar assigns its value to *tempCarOwner.carType*, the "while" that ends the loop in line 6 validates the users input and continues if the value of *tempCarOwner.carType* is not equal to "a", "b" or "c".

Login and sign up

To test whether or not our sign up and login features worked, the program was run 4 times. 2 times each logging in and signing up. They then branch out to each of the 2 user types. The test was successful for the sign-up feature if the user created was stored in the user data file and was presented the options for either a renter

or car owner. A successful test for the login feature would be if the user data was able to be retrieved from the user data file. User options being presented for the logged in user is also a requirement.

Edit profiles

The editing of profiles can be tested by checking the profile before and after editing. The new input information can simply be compared to the profile after the editing has occurred.

Rating system

The testing of the rating system is done by checking the rated profiles after the rating has occurred. After the transaction has been made between the renter and the owner, both parties have the option to rate each other. After the rating process we can check both parties' profiles and see if the rating has been added to their respective profiles.

```
printf("Enter car type (a = city car/hatchback, b = sedan/station
  car, c = SUV/Van): ");
do
{
  printf("\na, b or c: ");
  tempCarOwner.carType = getchar();
} while (tempCarOwner.carType != 'a' && tempCarOwner.carType != 'b'
  && tempCarOwner.carType != 'c');
```

Listing 5.14: Car type validation - 165-170 - CarRentalFunctions.h

```
printf("Enter how much your car should cost per hour: ");
scanf(" %d", &tempCarOwner.price);

while (tempCarOwner.price > 100)
{
  printf("\nEnter a maximum of 100 dkk: ");
  scanf(" %d", &tempCarOwner.price);
}
```

Listing 5.15: Car price validation - 153-160 - CarRentalFunctions.h

```
printf("Enter postcode: ");
do
{
  scanf("\n%d", &tempCarOwner.postCode);
} while (tempCarOwner.postCode != 9000 && tempCarOwner.postCode !=
  9200 && tempCarOwner.postCode != 9210 && tempCarOwner.postCode !=
  9220);
```

Listing 5.16: Post code validation - 147-151 - CarRentalFunctions.h

Chapter 6

Discussion

As the program is now done and working as intended, it is time to discuss what could have been done different, maybe even better, while also discussing what was in the project's range of programming, what the project would need to succeed completely and what the future could hold for the program.

Things that could have been done different:

One of the things that could have been done different, was instead of making the rating system, we could have made a send request in the makeDeal function. What this would do was when the renter would have selected the car he/she would want to rent, a request was sent to the owner of the car. We would make a database for the requests, and the way this could work, was that the database would search through the email of the renters request and the owners email. It would search through the database until it would find the correct pair of email addresses of both the owner and the renter. Then when the pair was found, the owner would get the option if he/she wanted to accept the request or deny it, if they accepted the request then both parts would have made a deal, otherwise not.
Looking at our survey we could have been more willing to make the answers from people more open, e.g. those who do not live in Aalborg Municipality. Our survey was also only published on our own Facebook profile, instead of a variety of different Aalborg Facebook groups. The reason we did this was to only get answers from people within the municipality as our main focus city is Aalborg, but it could have helped to get more answers and see what people outside Aalborg municipality thought about this program. But since our main focus city is Aalborg, it would have been a good idea to publish the survey on a variety of different Aalborg Facebook groups, in our own Software 2021 group and in some Aalborg University groups, this could have given us a lot more answers and maybe the survey would have been more comprehensive. Another aspect would also be the credibility of the

respondents, as Facebook is a place where everyone can partake in almost everything. Judging from the group's own members, there is also a tendency to "troll" on Facebook, meaning that people on purpose provide an unserious or untrue answer, with the sole purpose of being in the way or spoiling the true value of the survey.
We could have made our database of logins in standard text files instead of the binary system. The binary system is much easier to setup for your program to start reading and writing but once your are far into programming it can be annoying to use binary system. The file made by a binary system can only be directly changed by the program itself, otherwise the file will become corrupted. This will mean that when testing the program you will have to rely on making changes with the program as you can not make simple changes in the data file like manually adding a user, slowing down testing. Another downside to binary is that once you have many users saved, it can be hard for you to understand what is actually in the file, if your program has stopped working properly. The database is hard to read by anything but your program.

What could the project also have made in the given time frame:

As for what the program could have included, maybe instead of something else or as a last option would be a precise price algorithm that could determine the price sealing for individual cars. This would be based on given information like brand, production year, features, L/km. It would probably take some time and feedback to develop the algorithm to properly fit the great majority (95%), but it would be a nice feature to have from the start, so that it could be adjusted as early as possible. The program would need a huge database from already existing cars to determine the car type just by the name of the car. Furthermore it would from the car type be able to suggest a price of the car, just by going through the cars of the same type in the database. There may be an API that could provide this for the program, so it may not have been troublesome, but as the group of the project does not yet have the capabilities for this, it was not prioritized. An accept option for car owners, so that they have the possibility to view the renter that would like to rent the car, and evaluate whether they are fit for the owner's requirements or not. This would be a top priority feature if the program is to undergo further development. The renter would select which car he or she wanted to rent, and when they have selected the car they found ideal for them, they would be able to send a request to the owner, and he/she would then be able to accept or deny the request from the renter.

What would the program need except for what it has, to succeed?

It is obvious that when someone rents something from another, there has to be some insurance. As the project is based on cars, this is very vital. People would

most likely never rent a car out to a stranger without an insurance policy that guarantees compensation if something were to happen to the car. This is also evident in the survey that the project published. So for the program to work properly and safely, there has to be a lot of legal administration implemented in the software. Front end developing would also be at the top of the priority list. This is because the program need to be enjoyable and manageable for the user. This would also visualize the renters different options of sorting through price or rating of car owners. The renter would also be able to select a different car type from the beginning instead of just going through all the different cars in the system to find the ideal type of car they would want to rent such as, city cars, station cars or SUV's.

What would the future be for the program? What is the potential?

The future of the program would be firstly to start implementing a GPS/location tracker for the cars. Secondly we would need some kind of insurance policy so that if something happens to cars when they are rented, the renters would have to pay for the damage. Thirdly we would need some a lot of front end, this would help the users and it would be much easier to look through the program and all the options given to the user. The potential of the program could be huge, because of the flexibility in the renting periods, and because the goal of the program is to help with the increase of using the resources that are already being used.

6.1 What we did not have to time or skill to implement/program

The program would obviously need some sort of GPS/location of the car, such as where it is parked and in what range they should park the car when the renting period is over. This part the group did not have the time or the skills to implement, therefore it is not in the program, but it would be a high priority going forward with this program. Another thing we did not have time to implement was the accept or deny a rent request for the owner and a send request to rent for the renters, what this means is when the renter has selected the desired car her/she can send a request to the owner, then the owner get to decide whether he/she want to accept or deny the rent request. It would also need some sort of way for the renter to explain where they parked the car to the owner, so that the owner easily can find it. This could be solved with a GPS tracker in the car, so that the owners could find the exact location when the renting period is over. We did not have time to implement which hours the car would be up for renting, meaning that the car owners could choose what hours it could be rented. E.g: The car is available from 9-15 and again from 17-23.59. This would make sure that the renter

could only rent the car within those hours, all from 1 hour to 6 hours. Another thing we did not have the time to implement was a sort of price algorithm that would allow the renters to buy more kilometers in every hour they rent the car. The program would give the first 25 kilometers within every hour for free, but if the renters would like to drive more than that, they would be charged for every extra driven kilometer. This would be connected with the GPS tracker, so for every extra driven kilometer more that 25, the renter would be charged the same amount of money. We would also need to give the renters an option to pay beforehand for more kilometers if they would already know that they were going to drive further than the 25 given kilometers. This is done because the car owner is the one responsible for the gas in the car. So some renters need is to only drive 5 km to get a sofa at Ikea, and another renters need is to go visit a friend 12,5 km away, and then has 12,5 km back. The different needs of the renters is different, and therefore the given 25 km is to make sure that the part where the car owner pays for the gas themselves does not get exploited. Furthermore it would need something as a 'Bluetooth key' for the renters so when a deal has been made only the renter of the car and the owner would be able to open the car. This would be an ideal way of also making the solution more flexible for the renters, because they do not need to find the key or get it from the owner beforehand. The 'Bluetooth key" would allow the renter to open the car and to start the car, the renter would receive the key on an app when the deal has been accepted.

Chapter 7

Conclusion

It is concluded that there is a real problem at hand, regarding the use of cars, meaning that cars are used inefficiently, especially compared to the process of producing them and how much they are actually used afterwards. It can also be concluded, that when reading the SDG 12, there is a connection between the CO2 emission problem and the inefficient use and production of cars. When looking at the first, second and fifth target, it is evident that the problems with cars are very relevant when compared to the issues explained.
One of the aspects that makes cars a world wide problem is convenience. As cars are not just a tool designed for specific work related purposes. Cars are luxuries that are sought after in different shapes and forms and used for many miscellaneous purposes, which makes it more about making life easier for the individual than it is about environmental health. This also points towards that making the use of cars come to a halt and using public transport instead is unrealistic. Therefore it is concluded, that a solution for cars that involves cars are necessary.
It is concluded, by a survey based on the municipality of Aalborg, that a majority believes that there are too many cars in Aalborg, though this could have been misinterpreted from the discussion. What more is that a majority is interested in a software, that enables short term renting of private cars.
It is concluded that the software of this project is indeed related to the problem statement. The program is somewhat fulfilling in terms of the time frame and the requirements that were set thereafter, though it lacks several implementations and components to be a complete and usable application. A majority of these implementations and components being out of the time and skill frames of the students. It can also be concluded by the relation between the survey and the program, that the program grabs the problem statement by its core and is a direct possible solution to inefficient use of cars in Aalborg.

Bibliography

[1] Paul Barter. *Cars are parked 95% of the time. Let's check!* URL: https://www.reinventingparking.org/2013/02/cars-are-parked-95-of-time-lets-check.html.

[2] *Befolkningsudvikling i Aalborg kommune*. URL: http://apps.aalborgkommune.dk/statistik/webaarbog/Befolkningstilvaekst/Struktur/Data_2020/indexlevel1/Data2020Befolkningstilvaekst.html.

[3] *CO2 emission performance standards for cars and vans*. URL: https://ec.europa.eu/clima/eu-action/transport-emissions/road-transport-reducing-co2-emissions-vehicles/co2-emission_en.

[4] *CO2 emissions from car production in the EU*. URL: https://www.acea.auto/figure/co2-emissions-from-car-production-in-eu/.

[5] Ann Ann Deiterich. *Metal as a Renewable or Nonrenewable Resource*. URL: https://sciencing.com/metal-as-a-renewable-or-nonrenewable-resource-5192262.html.

[6] *Distribution of distance travelled per person per day by travel purpose for urban mobility on all days*. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=File:Distribution_of_distance_travelled_per_person_per_day_by_travel_purpose_for_urban_mobility_on_all_days_Feb_2021.png#filelinks.

[7] *Energy efficiency in transport*. URL: https://www.wikiwand.com/en/Energy_efficiency_in_transport/.

[8] *Growth within: a circular economy vision for a competitive Europe*. URL: https://emf.thirdlight.com/link/8izw1qhml4ga-404tsz/@/preview/1?o.

[9] *History of car rental: what has changed?* URL: https://www.arnoldclarkrental.com/latest-news/275-history-of-car-rental-what-has-changed.

[10] *How Much Does It Actually Cost Manufacturers to Make a Car?* URL: https://www.mtvehicles.com/post/how-much-does-it-actually-cost-manufacturers-to-make-a-car.

[11] The World leaders. *THE GLOBAL GOALS*. https://www.globalgoals.org/. 2015.

[12] J.B Maverick. *What Raw Matierals do Auto Manufactures Use?* URL: https://www.investopedia.com/ask/answers/062315/what-types-raw-materials-would-be-used-auto-manufacturer.asp.

[13] Søren Dalbro Michael Berg Rasmussen Fenja Søndergaard Møller. *Har adgang til offentlig transport betydning for om man har bil?* URL: https://www.dst.dk/Site/Dst/Udgivelser/nyt/GetAnalyse.aspx?cid=45984.

[14] *Mobile danskere: Nu rejser vi 42,5 kilometer til og fra arbejde*. URL: https://www.danskindustri.dk/di-business/arkiv/nyheder/2018/2/mobile-danskere-nu-rejser-vi-425-kilometer-til-og-fra-arbejde/.

[15] Otua. *What is your car made of?* https://autobarncollision.com/2016/02/09/what-is-your-car-made-of/. 2016.

[16] *Passenger car registrations*. URL: https://data.oecd.org/transport/passenger-car-registrations.htm.

[17] *Pendling efter område*. URL: https://www.statistikbanken.dk/PEND101.

[18] *Plusbus*. URL: https://plusbus.dk/.

[19] *Renting requirements - Hertz*. URL: https://www.hertz.com/rentacar/reservation/reviewmodifycancel/templates/rentalTerms.jsp?KEYWORD=RENTAL&EOAG=SINC61.

[20] *stakeholder bank*. URL: https://mlsdev.com/blog/app-development-cost.

[21] *stakeholder goverment*. URL: .https://www.regeringen.dk/media/1219/styrketinnovationivirksomhedern.

[22] *stakeholder it support*. URL: .https://satvasolutions.com/importance-maintenance-support-app-development/.

[23] *stakeholder media*. URL: https://business.gov.au/marketing/advertise-your-business.

[24] *stakeholder Shareholders/owners*. URL: .https://www.projectmanager.com/blog/stakeholder-vs-shareholder.

[25] Danmarks Statistik. *Amount of passenger cars as of January 1, from 2007 through 2021*. URL: https://www.statistikbanken.dk/statbank5a/Graphics/MakeGraph.asp?interactive=true&menu=y&maintable=BIL707&pxfile=20211113124231351800937BIL707.px&gr_type=0&PLanguage=0.

[26] Danmarks Statistik. *Bestanden af køretøjer pr 1 januar efter område og køretøjstype*. URL: https://www.statistikbanken.dk/BIL707.

[27] *Stor vækst i antal familier med flere biler*. URL: https://www.dst.dk/da/Statistik/nyt/NytHtml?cid=29118.

[28] *The problem of automotive inefficiency*. URL: https://www.mobilitydisruptionframework.com/the-problem-of-automotive-inefficiency.

[29] *The Top 5 Car-rental Companies*. URL: https://www.travelandleisure.com/worlds-best/car-rental-agencies.

[30] *TOURISM VEHICLE RENTAL MARKET - GROWTH, TRENDS, COVID-19 IMPACT AND FORECASTS (2021 - 2026)*. URL: https://www.mordorintelligence.com/industry-reports/tourism-vehicle-rental-market.

[31] National Geographic Tyson brown. *Automobile*. https://www.nationalgeographic.org/encyclopedia/automobile/. 2020.

[32] *UN - facts*. URL: https://www.un.org/sustainabledevelopment/sustainable-consumption-production/.

[33] *UN 12th goal - Ensure sustainable consumption and production patterns stats*. URL: https://unstats.un.org/sdgs/report/2019/goal-12/.

[34] *Undersøgelse: Danskerne, kollektiv transport og grøn omstilling*. URL: https://passagerpulsen.taenk.dk/bliv-klogere/undersoegelse-danskerne-kollektiv-transport-og-groen-omstilling.

[35] *What Are Cars Made Of? 10 Of The Top Materials Used In Auto Manufacturing*. URL: https://maycointernational.com/blog/what-are-cars-made-of/.