

---

---

# P1 Project

- Automatic Bargain Hunting -

---

---

Project Report

cs-22-SW-1-P1-13



# AALBORG UNIVERSITY

## STUDENT REPORT

Department of Computer Science

Aalborg University

<http://www.aau.dk>

**Title:**

P1 - Project 1st semester

**Theme:**

Automatic Bargain Hunting

**Project Period:**

Fall Semester 2022

**Project Group:**

cs-22-SW-1-P1-13

**Participant(s):**

Emil Suphi Doganci

Julia Lindharth Ekinci

Kristoffer Højmark Jensen

Mathias Mosgaard Larsen

Merete Kaldahl Andersen

Natasja Rosengren Jensen

Tobias Tjørnelund

**Supervisor(s):**

Gabriela Montoya

**Copies:** 1

**Page Numbers:** 65

**Date of Completion:**

December 21, 2022

**Abstract:**

This project discusses the limitations and possibilities of a programming project that aims to help students in Denmark afford groceries amid rising prices and inflation. The program allows users to create a shopping list and find the cheapest prices from the listed stores. We reflect on potential improvements to the program, including the implementation of dynamic and flexible programming techniques to address its reliance on premade data and improve its adaptability. We also discuss the decision to exclude certain features, citing time and resource constraints and a focus on delivering a functional and useful product within the available time frame. The project concludes by considering the potential for future development and the program's potential to be useful for other groups beyond Danish students.

# Contents

<b>Preface</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
<b>2 Problem analysis</b>	<b>4</b>
2.1 The rising inflation and why it's relevant for this project . . . . .	6
2.1.1 The development in user habits regarding grocery stores . . . . .	6
2.1.2 Population groups affected by inflation . . . . .	8
2.1.3 Advise on saving money . . . . .	8
2.2 What differentiates our project from the existing solutions? . . . . .	9
2.3 Simulation and handling of data . . . . .	10
2.3.1 Data scraping . . . . .	10
2.3.2 Accessing data using API . . . . .	11
2.3.3 Scraping - By using HTML . . . . .	13
2.3.4 Our own program . . . . .	15
2.4 Recommender systems . . . . .	16
2.4.1 Main approaches to developing recommender systems . . . . .	17
2.4.2 Main challenges faced by recommender systems and how to overcome them . . . . .	22
<b>3 Problem statement</b>	<b>24</b>
<b>4 A software based solution</b>	<b>25</b>
4.1 Program requirements . . . . .	26
4.2 Program design . . . . .	28
<b>5 Implementation of the program</b>	<b>30</b>
5.1 Structs . . . . .	30
5.1.1 main.h . . . . .	31
5.2 Distance function . . . . .	33

5.3	User profile . . . . .	35
5.4	Store management.c . . . . .	37
5.4.1	Validate file pointer and store_db . . . . .	37
5.4.2	Create store database . . . . .	38
5.4.3	Collect store information . . . . .	40
5.4.4	Products, list of groceries and prices . . . . .	41
5.4.5	Collect product prices and names . . . . .	42
5.4.6	Find product names . . . . .	43
5.5	Sorting algorithm and comparator . . . . .	45
5.5.1	Sorting algorithm . . . . .	45
5.5.2	Comparator . . . . .	47
5.6	Random_sale_decider and set_on_sale . . . . .	48
5.6.1	Random function . . . . .	48
5.6.2	Set products on sale . . . . .	49
5.7	Load_shoppinglist and sum_of_products . . . . .	50
5.7.1	Load user shoppinglist . . . . .	50
5.7.2	Sum of user products for each store . . . . .	52
5.8	Output . . . . .	54
5.8.1	Print . . . . .	54
5.8.2	Print promotions . . . . .	56
<b>6</b>	<b>Discussion</b>	<b>58</b>
<b>7</b>	<b>Conclusion</b>	<b>61</b>

# Preface

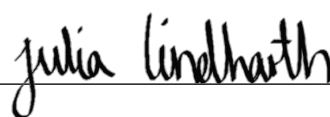
Aalborg University, December 20, 2022



---

Emil Suphi Doganci

<edogan22@student.aau.dk>



---

Julia Lindharth Ekinci

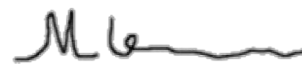
<jekinc22@student.aau.dk >



---

Kristoffer Højmark Jensen

<khje22@student.aau.dk>



---

Mathias Mosgaard Larsen

<mmla22@student.aau.dk>



---

Merete Kaldahl Andersen

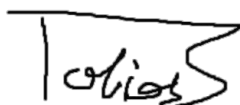
<mkan22@student.aau.dk>



---

Natasja Rosengren Jensen

<nrje22@student.aau.dk>



---

Tobias Tjørnelund

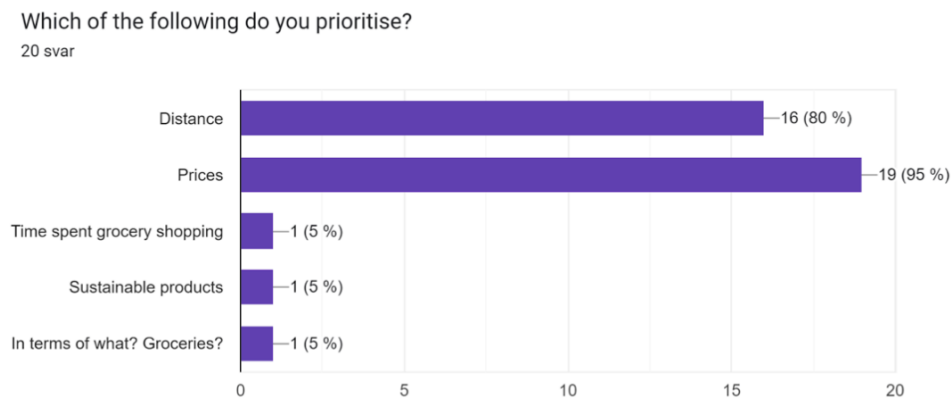
<ttjorn22@student.aau.dk>

# Chapter 1

## Introduction

When 2022 began, no one could have predicted events like the Russo-Ukrainian war, which significantly increased energy costs and, later, led to massive global inflation. Inflation occurred in the blink of an eye and continued to rise. Although some people may continue to buy groceries despite rising prices, many students have a different mindset and budget. Inflation has had an impact on prices in countries all over the world, including Denmark. Many everyday items, particularly groceries, have become more expensive.

In Denmark, the price of butter and beef both saw significant increases from July 2021 to July 2022. Butter prices rose by 50%, and beef prices rose by 28.7%[22]. This can be a significant burden for students in Denmark who rely on SU, a financial aid program provided by the state, to cover their expenses. For many students, SU may not be enough to cover their bills and other expenses, especially with the rising cost of food. We started our project by conducting a survey that our fellow students answered at Aalborg University. Some key points from the survey we would like to highlight is that 95 percent of the students that answered prioritize prices when shopping for groceries, and 80 percent prioritize the distance to grocery stores. This is also shown in figure 1.1. Since we wanted to create a program that suits students' preferences, we included these data in our program solution.



**Figure 1.1:** The table shows what students prioritize regarding grocery shopping

In September 2015, the United Nations adopted 17 global goals, one of which is responsible consumption and production. Our program aims to address this issue by helping consumers shop more responsibly by allowing them to create and follow a shopping list. This helps reduce irresponsible consumption and waste by ensuring that consumers only purchase what they need and do not make unnecessary or impulsive purchases.

Due to time and resource constraints, we have chosen to focus on Danish students for this project. Nevertheless, our solution may be helpful for other groups as well. Our project aims to address inflation's challenges and provide affordable grocery options.

## Chapter 2

# Problem analysis

To begin this project, it is necessary to conduct thorough research using various methods, sources, and approaches. As a starting point, it is important to understand the UN's 12th global goal, which serves as the foundation for this project. By analyzing this goal, we can gain a better understanding of the motivations behind the project and its significance. This will provide us with a strong foundation on which to build our analysis and approach to addressing the problem at hand.

The 12th global goal of the United Nations is responsible consumption and production. By addressing unsustainable ways in which our planet's resources are currently used, this goal is intended to promote economic growth and sustainable development. We can ease the pressure on the environment and support the implementation of the 10-year sustainable consumption and production framework by shifting to more sustainable consumption and production patterns. There are several sub-goals under this goal, including reducing food waste in particular [13].



**Figure 2.1:** Global goal 12: responsible consumption and production

### Target 12.1

*“Implement the 10-Year Framework of Programmes on Sustainable Consumption and Production Patterns, all countries taking action, with developed countries taking the lead, taking into account the development and capabilities of developing countries.” [13]*



Within 10 years, all countries are expected to achieve more sustainable consumption and production patterns. The 10-year sustainable consumption and production framework should be implemented, with developed countries taking the lead and considering developing countries' development and capacity.

### **Target 12.3**

*"By 2030, global food waste at retail and consumer level per population is halved and food loss in production and supply chains, including loss of crops after harvest, must be reduced."*[13]

Retail and consumer food waste, as well as food loss in production and supply chains, are the focus of this target. "A calculation from Statistics Denmark showed from May 2021 to May 2022, general food prices had risen 10.1 pct, a 330 DKK increase a month for an average household"[10].

As of now, 1.3 billion tons of food is wasted each year, and the target aims to double this amount by 2030. Food waste has become a particular issue in Denmark due to inflation-induced high food prices. Platforms like "etilbudsavis" and "minetilbud.dk" address this issue by providing offers and bargains, but do not take distance to supermarkets into account or overall savings.

These websites also provide deals and special offers on products. They do not focus solely on making the user create a shopping list. Creating a shopping list can help prevent consumers from making impulse purchases and buying products they do not need. When consumers create a shopping list, they take the time to think about the items they actually need and plan out their purchases in advance. This can help them to be more focused and avoid making impulsive decisions while they are in the store, which can lead to buying unnecessary items.

Additionally, having a shopping list can help consumers avoid being tempted by other products that they may not have originally planned to purchase. The risks of the consumer being tempted by attractive displays or special offers, on products they don't actually need, is a lot higher than if they make a shopping list. Even though the consumer has to actually enter the supermarket in order to buy the products on their shopping list, it can help them save money by preventing them from overspending on items they don't need. By sticking to their list and avoiding impulse purchases, consumers can avoid purchasing items that they may not have budgeted for and avoid overspending.

To provide students with the most accessible and cost-effective options, our project takes into account both shopping lists, offers/bargains as well as distance to supermarkets. Our survey identified distance and price as priorities when grocery shopping. Bargain hunting and shopping lists can ultimately be useful tools that can eventually help students with overspending and prevent irresponsible consumption habits, which can contribute to reducing food waste in Denmark and contributing to the 12th global goal.

## 2.1 The rising inflation and why it's relevant for this project

Inflation refers to an overall increase in the price of goods and services, as opposed to an increase in individual prices. The European Central Bank aims to keep inflation at 2 percent per year, but not negative. A decrease in general prices is called deflation. According to the Danish National Bank, Denmark's inflation is expected to reach 8.6 percent by 2022. In addition, the National Bank estimates that inflation will decrease to 4.3 percent in 2023 and 1.7 percent in 2024. During the past couple of years, the world situation has changed dramatically, including the outbreak of covid-19 and the invasion of Ukraine. This has resulted in high inflation as a result of the large difference between supply and demand[21].

### 2.1.1 The development in user habits regarding grocery stores

The effect of inflation on Danish spending can be seen in their choice of grocery stores. According to Danish statistics, discount stores have seen an increase in turnover, while supermarkets have decreased.

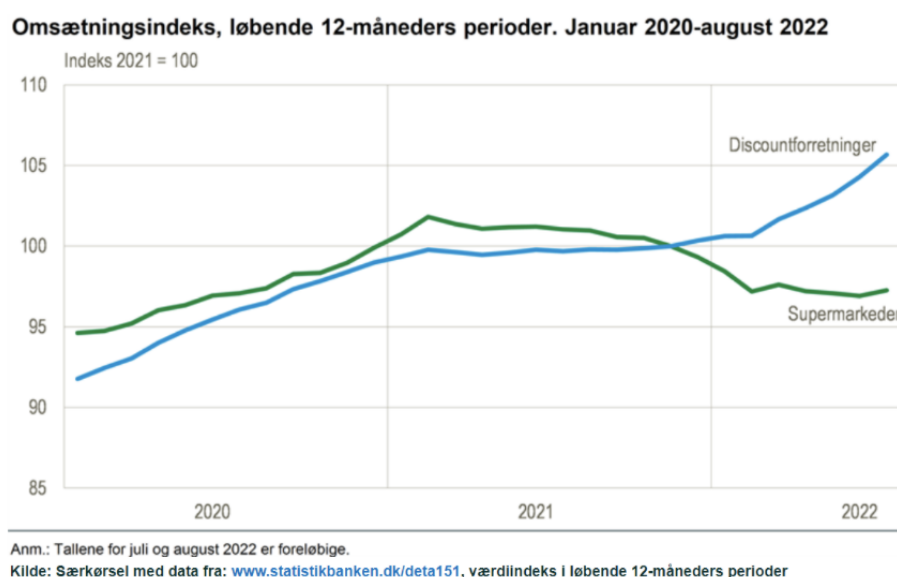


Figure 2.2: The graph shows the turnover index from January 2020 to August 2022[33]

There is a growing awareness among consumers about their consumption, and they want to save money when it comes to food purchases. Figure 2.2 illustrates this, showing that the two types of grocery stores both increased turnover fairly evenly until late 2021, when the two lines started moving in opposite directions, with discount stores gaining in popularity. Based on Danish statistics, this trend may continue in the next few months[33].

As shown in Figure 2.3, beef and dairy products have been among the most affected by

**Her er de fødevarer, der er steget *mest* i pris**

Vare	Juli 2021 - juli 2022
Spiselige olier undtagen olivenolie	105,0%
Smør	50,7%
Margarine	31,0%
Okse- og kalvekød	28,7%
Babymad	28,4%
Pastaprodukter og couscous	27,7%
Andre mejeriprodukter	27,7%
Frisk mælk	26,9%
Frisk mælk med lavt fedtindhold	26,7%
Fersk fisk	24,8%
Æg	24,6%
Chips mv.	24,2%
Frossen frugt	23,3%
Ost	23,0%
Kaffe	21,7%

Kilde: Danmarks Statistik

**Figure 2.3:** The graph shows the products that has increased the most in price[22]

the sudden rise in inflation. From July 2021 to July 2022, the prices of different goods increased. The price of beef products has increased 28.7%, butter by 50.7% and other dairy products by about 27%. Farmers' expenses are rising due to rising electricity, transportation, and feed prices. This rise in feed and electricity prices which is caused by the ongoing war in Ukraine and Russia, have had a big impact on food and other grocery prices all over the world. Ukraine is responsible for 13% of the global trade of maize, 10% of rapeseed, 11% of barley, 9% of wheat and 36% of sunflower oil and seeds, while Russia is responsible for 1% of maize, 6% of rapeseed, 12% of barley, 18% of wheat and 17% of sunflower oil and seeds [1]. Agricultural products such as chicken, pork, beef, eggs, and dairy products are priced higher because these grains are used as feed for cows, chickens, and pigs. Electricity and gas prices for transportation further increase these products' prices[39].

As a result of their perceptions of how consumption of goods and services will develop, 24% of 1895 respondents in a GFK survey plan to reduce their grocery expenditures in the coming years. When asked how they would reduce their grocery expenditures, 71% of respondents stated that they would go for more bargains when asked how to do so[25]. Students, especially, can benefit from the program by finding bargains that can help them save on food.

### **2.1.2 Population groups affected by inflation**

People outside the labor market are most affected by inflation, according to Danish statistics. Among these groups are pensioners, unemployment benefit recipients, students, early pensioners, and cash assistance recipients. Inflation has had the greatest impact on these things because they spend the largest proportion of their consumption on food, electricity, and heating[14].

It is our goal to develop a program for users to find the best overall price for all products they need. As an additional assistance, the program will also help users find individual bargains. We are targeting students because they are the most vulnerable group to inflation. In the past, Ida Marie Moesby, a Nordea consumer economist, said it was difficult for students to set budgets before inflation, but now it is even more difficult in 2022. In Denmark, all students receive state assistance called SU, which doesn't keep up with inflation. Therefore, creating a program that helps students make the most out of their small budget would be an interesting idea[24]. The program could also benefit other groups of people, but we choose to focus on students since we ourselves and our friends are students.

### **2.1.3 Advise on saving money**

Price increases are at an all-time high. You can save money in a number of ways, but making a food plan may be a good way to save money when you go grocery shopping. Statistics from "Forbrugerrådet Tænk" showed that you can save 15% of your food budget by making a plan. Additionally, you might want to consider only grocery shopping once a week and ensuring you have all the items you need for the following week. "Financer.com" says that doing so can save you up to 6000 DKK annually, along with helping you avoid impulse purchases and daily temptations. One of the main focuses of our program was to help users reduce impulse buying by developing a grocery list. You can also save money by going bargain hunting and buying only on sale items, and not buying items that could have been cheaper elsewhere. The program can also solve this problem, in that it allows users to enter grocery lists, and the program automatically returns the retailer with the best bargains according to the input[20].

## 2.2 What differentiates our project from the existing solutions?

There are already a few solutions to the problem that our project will be looking at. However, these solutions only solve a portion of the problem. All of these solutions resolve the same problem, which is to provide discounts only. Services and websites like "etilbudsavis" and "minetilbud.dk" only offer discounts. It is not necessary to sign up to make a user profile, set preferences, or offer suggestions regarding distance to different supermarkets. Neither do they show the total sum of products or which products are discounted at the same shop. To illustrate how we will differentiate, we have created a chart depicting the core elements of bargain sites.

Bargain websites	eTilbudsavis	Minetilbud.dk	tilbudsaviseronline.dk	Our project
Sign up/Log in?	They do not require a user profile, it is optional - you can still use the website without a profile.	They do not require a user profile, it is optional.	They do not require a user profile, it is optional.	Our program requires you to make a user profile, as the given inputs are to help you with finding bargains in your shopping list, and show you the lowest total sum of products in your shopping list.
Similarities - Shopping list - Sign Up - Discounts	This website has; - Shopping lists and sign ups available - It does not find the discounts of the products you have in your shopping list - you have to search for them yourself. If you want to make a shopping list, you have to search for each individual product, and find the bargain yourself, manually. - Discounts for all possible stores and supermarkets - can be really unmanageable to search for bargains on foods.	This website has; - Shopping lists and sign ups available - On this website, you have to sign up for an account, to make a shopping list. If you do not, the function is unavailable. - They also have discounts for all kinds of stores, not just supermarkets - that would make searching for promotional foods more unmanageable as well.	This website has; - Shopping lists and sign ups available - They also do discounts on all kinds of stores exactly like the other two websites. - Practically does the exact same thing as "etilbudsavis" and "minetilbud.dk".	We want to require users to make a user profile. Ideally, the users can add their shopping list to the program. But in this project, we have premade a shopping list. When the shopping list is added, the program will provide the lowest total sum of products, also showing which of the products are discounted. The program will display the distance to the supermarkets as well, based on the maximum distance that the user provides.
Distance?	This website has a distance feature, where you can change the minimum and maximum, and you can enter your address as well.	At this website, you can not enter an address, you have to find and select the city you want to find discounts in.	Here you cannot enter an address, you need to select a city as well.	We want to implement a distance feature, as well as when signing up, you will have to enter your address, in order for the program to calculate how far the supermarkets are from your exact location - and based on your inputs and distance, it will provide the closest nearby supermarket, that has lowest total sum of products in your shopping list.
Location	Denmark, Norway, Sweden	Denmark	Denmark, Norway, Sweden, Belgium, USA, Netherlands, France, and many more.	Aalborg
Discounts in your shopping list?	Does not show, you have to search for the bargains yourself.	Does not show, you have to search for the bargains yourself.	Does not show, you have to search for the bargains yourself.	We will display which of the products in your shopping list are discounted at each shop.
Target group	All groups	All groups	All groups	Students

As you can see, our project shares some similarities with the other competitors. Although all the other competitors are quite similar with only a few different features, our project is a bit more user based. We will help the individual user to find the best bargains and provide the distance to the stores, based on user-given inputs. We want to make it as simple and manageable as possible, with providing the discounts directly to the user, instead of having the user search for the bargains themselves. Overall, the idea of the program/app has the

same core values as the competitors, but we have chosen to implement a user profile, and provide a functionality where we will show you which of the products in your shopping list are discounted at the same shop - as we believe it is more manageable and more personal, so that the bargains and distance fit the users exact preferences and priorities.

## **2.3 Simulation and handling of data**

### **Introduction to data scraping**

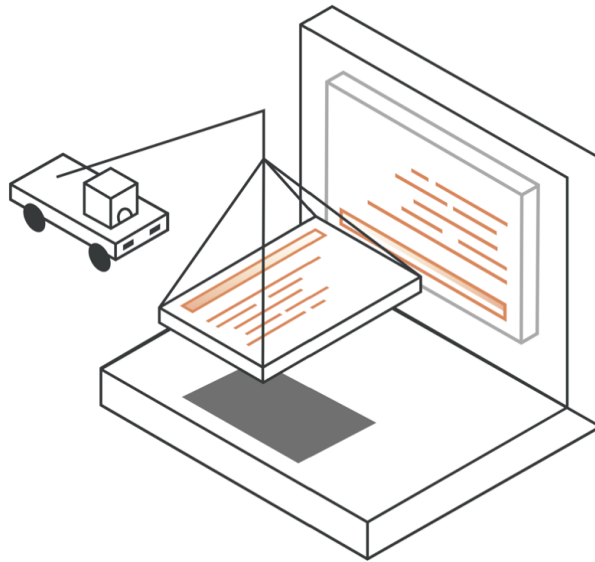
Data scraping is also called web scraping. It is being used to draw out content and data from a website and it uses bots to complete this procedure. Data scraping takes out stored data in a database by also taking out underlying HTML code. The scraper is then capable of taking the scraped data and recreating the content of a website in another place.

#### **2.3.1 Data scraping**

To completely understand the following section we are giving some words explanations. A bot is an abbreviation for robot, and it is a program for a computer that works as an intelligent agent for a user, a program or to simulate an activity for a human. Some tasks are usually being automated by bots, which means the bots can work without certain human commands[12].

An intelligent agent (IA) is an independent computer software system. It can reach desired targets and interact with people or events if it happens around the agent. IA is being programmed with the use of machine learning, which is a subfield of artificial intelligence (AI), and it is capable of observing and adapting to different situations because it is provided with sensors that makes this possible. It is being utilized in situations where people interact because IA can demonstrate basic social skills[37].

API is short for Application Programming Interface, and will have a further explanation under that specific topic.



**Figure 2.4:** Illustration of data scraping [38]

Figure 2.3 illustrates the data scraping process in a very simple way. The vehicle portrays a digital program that extracts an output of data from an already existing program into another program.

Data scraping is being used in a broad variations of digital businesses that data harvest and the legal ways of using data scraping is:

1. "Search engine bots crawling a site, analyzing its content and then ranking it".
2. "Price comparison sites deploying bots to auto-fetch prices and product descriptions for allied seller websites".
3. Market research companies using scrapers to pull data from forums and social media (e.g., for sentiment analysis)."[36]

### **2.3.2 Accessing data using API**

API's, which are an acronym for Application Programming Interface, are used in order to transfer data between two computer programs in real time, regardless of whether the programs are written in the same language or not. APIs act as an intermediary between two different pieces of software, where you can have either an open or closed API. Open APIs can be used to quickly share necessary data that can be used by others in their own software, whereas a closed API is used in a private manner, for instance to transfer sensitive data that needs to be encrypted during the transfer[8].

A web API uses HTTP to request data, which is often returned in the form of a JSON or XML file. This JSON or XML file can then be processed as desired by your own program, and presented with a good user interface. As a result of APIs, companies are not forced to

manually update and process the relevant data every time, so they are able to save both time and money[18].

Companies do not usually want others to extract their data content and use it for something else therefore they do not make all their data easily visible or as an API. Scraper bots can be used indefinitely to extract all the hidden data, and they do not care if the data is easily accessible or limited. So even though companies hide and protect their content the scraper bots compete with the protection and try to outmaneuver it. Data scraping have three steps:

1. "First the piece of code used to pull the information, which we call a scraper bot, sends an HTTP GET request to a specific website."
2. "When the website responds, the scraper parses the HTML document for a specific pattern of data".
3. "Once the data is extracted, it is converted into whatever specific format the scraper bot's author designed".

Different types of illegal data can be collected by the bots, such as content scraping, price scraping, and contact scraping. Content scraping is all or next to all content that is being drawn from a website onto another. The content can be everything that is online, such as pictures, text, CSS code, HTML code etc. This can be done with or without the consent of the owner. The majority of content scraping occurs illegally because people copy someone elses material and claim it as their own. Price scraping is when the bots check a competitor's prices, and track and monitor the prices. They are always up to date on what they should sell the same product for and therefore give them a huge advantage at getting customers to buy from their website instead of the original. In this project price scraping could be relevant to use legally, so the app always is updated with the new prices, and the variation of prices throughout every supermarket in Denmark.

Contact scraping can be used when a website holds personal information about for instance their employees. The data is being drawn to get the personal information such as a phone number or email address for bulk mailing lists, robo calls, or for an evil purpose. This is the way scammers and spammers get their information to get in contact with people.



### 2.3.3 Scraping - By using HTML

In order to pick out specific tags, such as the price or name of an item, in large quantities of data from a HTML website, it is important to understand the structure of the webpage. It is possible to import the HTML code of a website, but it is also an option to look at the structure by using your web browser's developer tools. The developer tool is accessed by pressing Ctrl+Shift+I. This tool is used to display a website's elements, and it is accessible for the user by clicking the different HTML elements in the structure.

Some websites are easier to scrape than others. With static websites, there is a resemblance between what the user views on the website, and what is written in the HTML code[4]. Dynamic websites are trickier to read by only viewing the HTML code since java scripts are often implemented in the code. Dynamic websites are often large and have multiple pages. The information on the website is often stored in databases, and therefore also being updated through the database and not through the HTML code, as with static websites[16].

Figure 2.4 illustrates where the price of a product can be found in the HTML code of REMA 1000's website. In this case, the price can be found in a <div with the class "price". and is set to be 14 danish kroner.

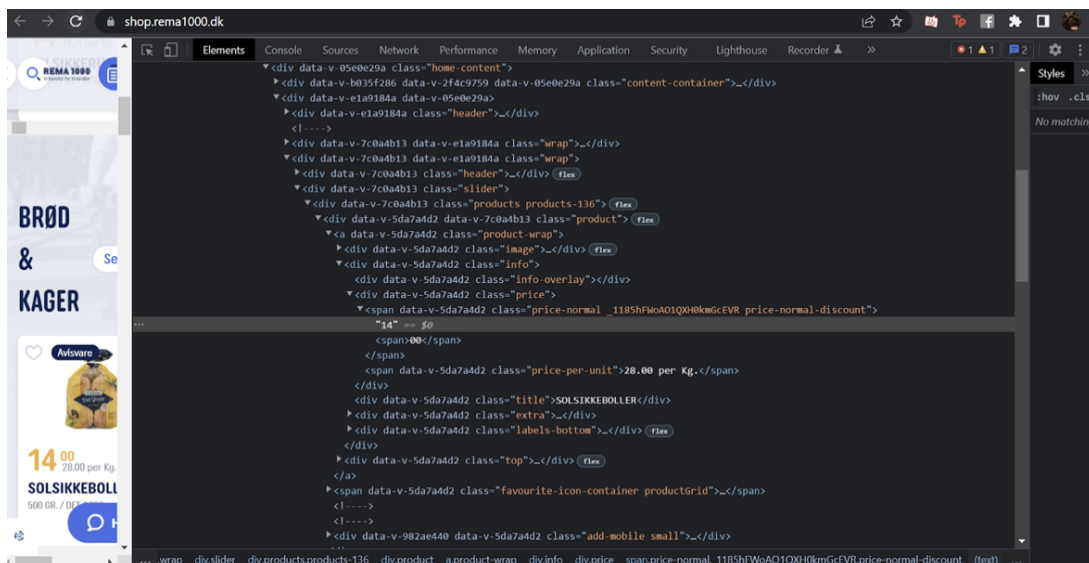
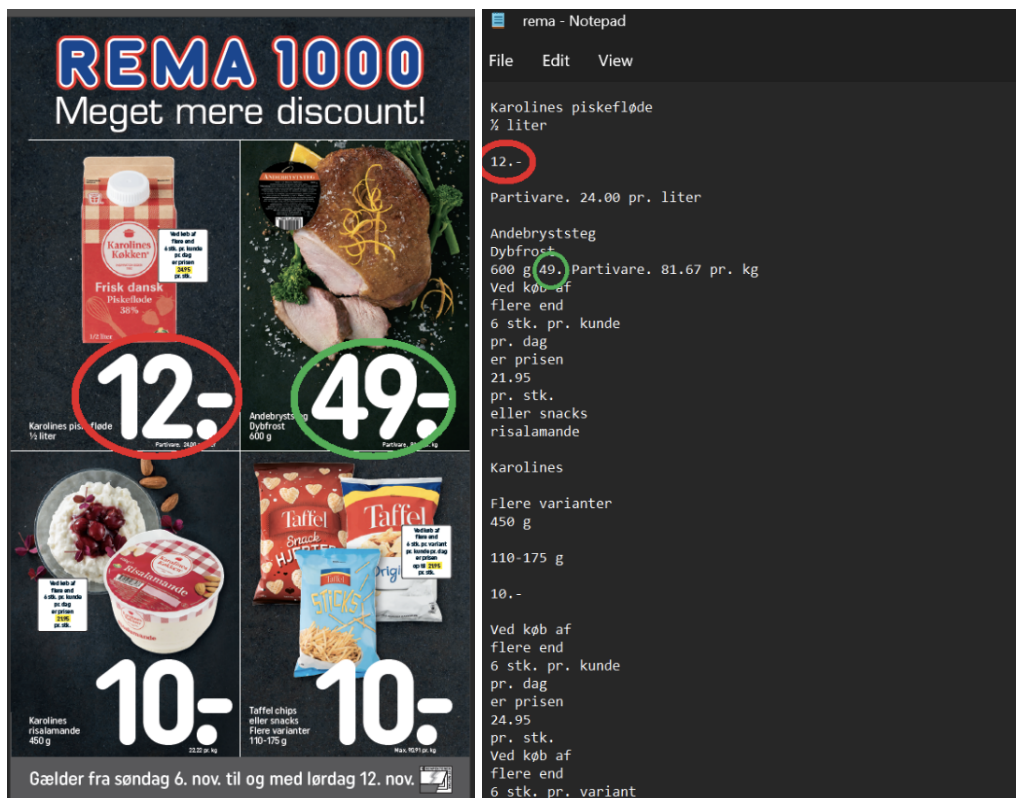


Figure 2.5: HTML elements of REMA 1000's website. Accessed by using the developer tool of a web browser [31]

If you are only looking to find a few specific pieces of information on a website, it is completely doable to do the scraping of a website manually, but in case of pulling large amounts of data, it would be very time consuming. This procedure can be automated by using scraping bots . A scraping bot can parse through a website to find the desired information. This only requires some preset parameters, so that the bot knows what to look for[23].

Another method for extracting information from a grocery store and in this case, a leaflet with bargains, is to download the leaflet from a store's website, and then convert the PDF-file to a text-file. It could also be a possibility to automate this process, but there are some issues with this method. As seen in Figure 2.5 and 2.6, the structure of the information is not consistent in all of the bargains. The bargain price for the cream is highlighted with a red circle, and the price of the duck is highlighted with a green circle. The duck's price is almost concealed between all of the other information. Therefore it would be hard for a bot to identify and extract these informations.



**Figure 2.6:** A leaflet from REMA 1000 converted to a PDF-file [27]

**Figure 2.7:** The leaflet from Figure 2.5 converted to a text-file using Adobe Acrobat [27]

During the course of this project, APIs can be used in order to get information about the prices of the product on sale and the store in which the product is sold. An API can save us a lot of time, since it does not require storing data locally, nor does it require manual updating of data each time a supermarket adjusts their weekly offers. The API can be utilized after the desired shopping list has been created. It is at this point that the desired items will be given to the application, which is then able to send the information to the APIs of the various stores, which can then process this information and return the prices of the desired items, as well as whether they are currently on sale and available. As a result, consumers can then digest this information more easily by presenting it with a well designed user interface that makes it

easier for them to understand.

#### **2.3.4 Our own program**

Since there are many difficulties with scraping different websites for bargains and prices, we decided to simulate our own data for our program. One way to create data using C, is to use structures. Structures, also called structs, are used to group multiple variables, also called members of the struct. Structs can contain numerous data types such as integers, characters, doubles and many more. This purpose is relevant to our program since we desire to have multiple members for each of our structures. An example of a member could be “Name”, which could be an array of characters. Another example is coordinates, which would most likely be of the data type, double[35].

## 2.4 Recommender systems

### Introduction to recommender systems

Anyone who has ever browsed through an online shop for some items or youtube and netflix for some videos, movies and tv-shows may already be familiar with what these recommender systems are and how they function but we are going to enlighten you with more information. According to a recent report from the Statista Research Department, global data creation is expected to reach 97 zettabytes in 2022 and exceed 180 zettabytes by 2025. A zettabyte is equivalent to approximately a trillion gigabytes, which is equivalent to around a billion devices with an average storage capacity of 1,000 gigabytes each. Think about the situation if there were no engines, algorithms, systems or any other tools available to help navigate through such a vast amount of information. How many days, weeks, months or years would you estimate to find what you're looking for?[32]

Search engines provide a good effort to address this problem, however search engines are designed for the purpose of finding precise data that correspond to a query of keywords, while recommender systems are specifically designed for the purpose of determining relevant content for a specific user.

According to dictionary.com, the exact definition for a search engine is the following [30];

*"a computer program that searches documents, especially on the World Wide Web, for a specified word or words and provides a list of documents in which they are found."*

Where as according to a publication by researchers from the National Institute of Technology, a recommender system is described as the following [28];

*"An RS is an intelligent computer-based technique that predicts on the basis of users' adoption and usage and helps them to pick items from a vast pool of online stuff." RS being a recommender system.*

## **What exactly are recommender systems, and how do they work?**

A recommender system or engine is a subclass of an information filtering system which is a system that removes unwanted information for various purposes. Unlike the information filtering system, a recommender system will not focus on removing unwanted information but rather specifically provide wanted information. Recommender systems are typically used for decision-making processes such as purchasing items, looking for entertainment etc. It will look for relevant items for a particular user, based on common traits or patterns in data that it received through various techniques and methods[3].

The main approaches for these recommender systems consist of collaborative based filtering, content based filtering or a hybrid of these two approaches.

The differences between recommender systems and search engines may now be apparent as search engines do not provide personalization of data, i.e. They do not search through data to identify what is more relevant for each individual user, but instead search for data that is relevant to general searches. By contrast, recommender systems provide a way to analyze data in order to determine relevance based on the user's behavior. This difference makes it evident that it is of great significance to explore. So let's look deeper into the different approaches and currently existing recommender system technologies.

### **2.4.1 Main approaches to developing recommender systems**

There are mainly three approaches to developing recommender systems as mentioned previously; collaborative filtering, content based filtering and a hybrid approach.

Collaborative filtering(CF) is based solely on past interactions between users and items in order to produce new recommendations. *"This includes the user's online activities and predicting what they will like based on the similarity with other users."*[29]

A collaborative filtering recommender system is generally divided into two subcategories called memory based and model based approaches. Memory based recommender approaches do not use a model but use past recorded interactions to produce new recommendations. *(Example: A recommender system suggests a person to buy a specific lawn mower. The recommender system looked at the most popular lawn mowers among the person's neighbors and made the best recommendation from those lawn mowers to the person.)*

Model based recommender approaches use an underlying "generative" model that explains the user to item interactions and try to discover it in order to make new predictions[29].

There are a lot of advantages by using a collaborative approach. The main advantage is that they do not require information about new users or new items, and therefore can be used in many situations. For every user interaction the system records new information and becomes

more accurate generating better recommendations for the end user. The drawback from using only past interactions to generate new suggestions is that the approach can not recommend anything to a new user/item because the system does not have any data to compare with (also called cold start problem). This typically leads to random or most popular recommendations to a new user which is not very accurate and could lead to a lost sale in a business as an example[29].

Unlike collaborative methods, content based(CB) approaches use additional information about users and/or items. This can include age, sex, job or any other personal information.

The idea is to build a model based on the available “features”, that explain the observed user-item interactions. By looking at users and movies, typically young women tend to rate some movies better than young men will do because they like other types of movies. If such a model would be determined to be relevant for determining relevant movies for a user it would be very easy to make new predictions. In this specific model only age and sex is main information to determine relevant movies but more information about a user could make the system even more accurate.

Unlike collaborative approaches, content based methods do not suffer from a cold start problem because we already have some characteristics to build a model from. If we go back to the example about movie ratings an interesting complication could happen if no ratings from women ever were captured by the model. What would the model do? In its first attempts it would try recommending the movies men like and sooner or later it will learn from the women’s ratings. The more users registering their data to the model makes it even stronger. This means new users or items with previously unseen features will almost never suffer from a cold start if the system is old enough with plenty of data to compare with[29].

There are some key differences in the filtering methods. Memory based collaborative methods require no latent model. The algorithm used works directly with the user to item interaction. Example: Users will be recommended suggestions by their interactions with items and a neighbors interactions. These methods have a theoretical low bias but high variance[29].

Hybrid approaches are a combination of two or more filtering methods in order to gain better performance over CB and CF approaches when they are used alone. The overall structure of hybrid approaches can be illustrated with figure 2.10. More accurately this figure illustrates one of the ways of implementing a hybrid recommendations system. This implementation revolves around training two models independently, one collaborative filtering model and one content based model and then combining their suggestions.

A research team from the Department of IT, Maharashtra Institute of Technology in Kothrud, Pune, India, have illustrated an example of the difference of performance based on different factors on these three approaches using the same dataset. In addition to the two recommendation systems mentioned, demographic attributes are also considered for forming the hybrid union in the examples below, as described in the research. It can be observed that the hybrid approach outperforms the other two methods when they are used alone[7]. Demographic attributes are used as pre-existing knowledge of demographic information about the users and their opinions regarding the recommended items[15].

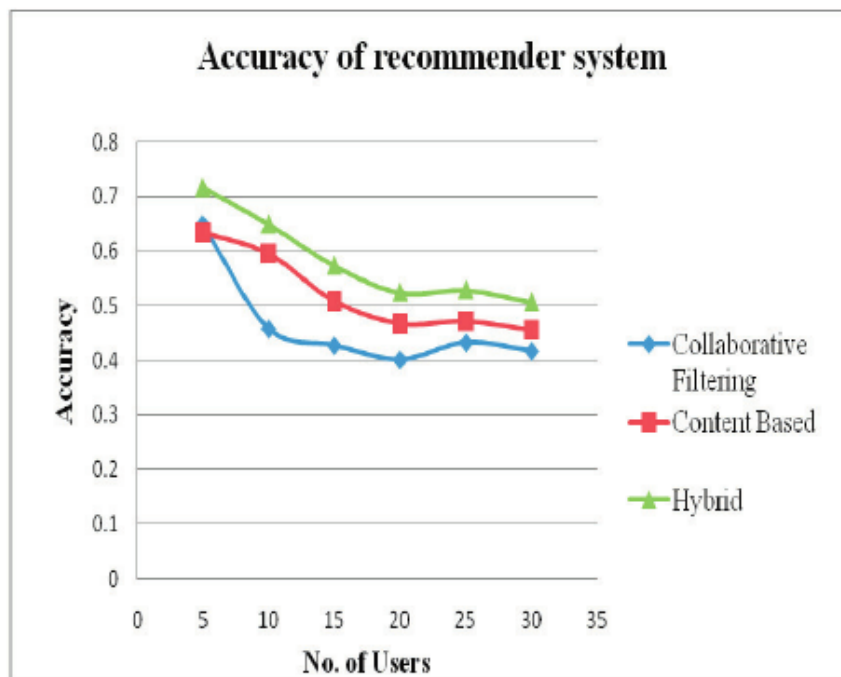
### **Metrics used for analysis:**

Accuracy is defined as the fraction of relevant recommendations made by the system out of all possible recommendations. By using this metric, it is possible to determine the overall effectiveness of the system in making relevant recommendations.

Precision, on the other hand, refers to the percentage of relevant recommendations made by the system relative to the total number of recommendations. By using this metric, we can determine to what extent the system is providing recommendations that are actually relevant to the user. Last but not least, recall measures the proportion of relevant recommendations made by the system as compared to the total number of relevant recommendations that could be made. By analyzing this metric, you can determine how complete the system's recommendations are, or how many relevant items it is capable of recommending [7].

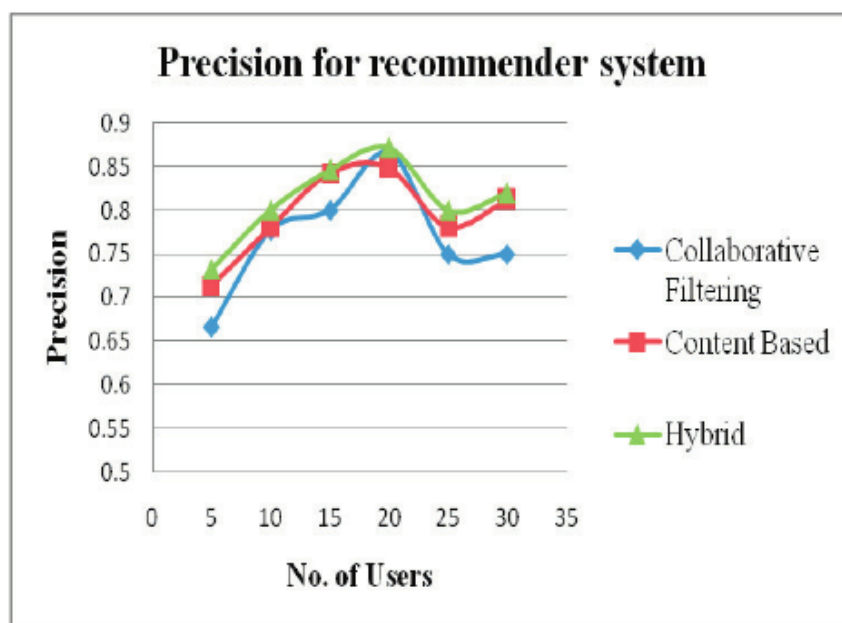
The collaborative filtering accuracy decreases as the number of users increases and then gradually recovers. As explained above in the collaborative filtering explanation, this is a demonstration of the 'cold start' problem. Because there are more users, there will be a larger gap in the initial data consisting of "user-to-item interaction", which will reduce the accuracy of the recommendation output as there will be an insignificant amount of data to work with.

In theory, content-based filtering should perform better as more users appear, however, this can be difficult to observe in this figure. At around 20 users, the content-based recommendation becomes more accurate. Overall, it is more accurate than collaborative filtering in this example since it uses additional information about users and items. However, this raises the question of whether or not collaborative recommendations may outperform content-based recommendations as the number of users increases beyond the size of the dataset used. Due



**Figure 2.8:** Comparative analysis of recommendation techniques on the basis of Accuracy [7]

to the combination of these two approaches, it is not surprising that the Hybrid outperforms them, as illustrated in figure 2.11.



**Figure 2.9:** Comparative analysis of recommendation techniques on the basis of Precision [7]



Combining collaborative filtering with content-based filtering increases the likelihood of correct recommendations, since collaborative filtering recommends based on ratings while content-based filtering recommends based on item characteristics. Consequently, the Hybrid recommender technique has resulted in an improved level of precision. Additionally, in the source of this example, it is suggested that unusual tastes or preferences may affect the recommender system, which could explain why the collaborative filtering spikes out of a stable precision at a certain number of users.

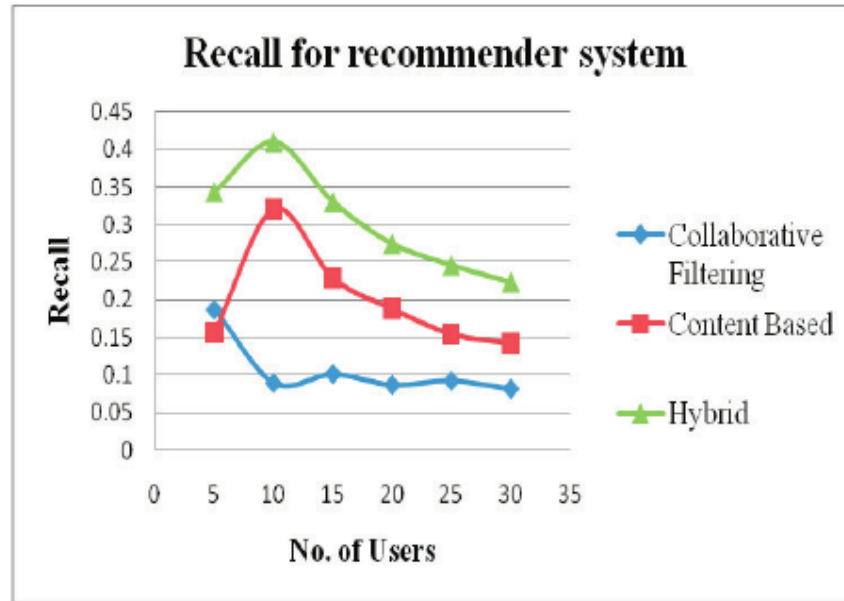


Figure 2.10: Comparative analysis of recommendation techniques on the basis of Recall [7]

As the number of users increases, the recall decreases and then stabilizes. This is particularly true in the case of collaborative filtering recommendations. We can explain this by assuming that there is less demand for knowing "user-to-item" interactions as the user count increases, which results in this stabilization as the ratio of relevant recommendations to total relevant recommendations decreases.

### Conclusion on the best recommendations approach in the example and in relation to our project

A conclusion can be drawn that combining collaborative filtering and content-based techniques with demographic attributes results in the best recommendations based on the examples reviewed. Using a hybrid form also overcomes the "cold start" problem for new users[7]. However, our proposed solution does not use user-to-item interactions or user comparison as data points. If we were to implement a proper solution, the hybrid form would be the preferred approach.

## 2.4.2 Main challenges faced by recommender systems and how to overcome them

The challenges could be divided into many different types, including cold start problems, synonymy based problems where recommender systems are unable to distinguish between synonyms of item names, interventional issues such as privacy concerns and shilling attacks, scalability issues, latency issues, etc. In light of the conclusion made earlier, we will shortly discuss some of the most relevant problems that may arise if we were to attempt to implement a recommender system[17]. We will discuss one mechanical problem and two interventional problems that may arise.

**Cold Start Problem** As mentioned previously this problem occurs when new users enter the system or new items are added to the catalog. In these cases recommendations cannot be made without pre-existing knowledge consequently leading to less accurate recommendations which was also demonstrated and explained in figure 3. It should be noted that this type of problem usually happens for collaborative filtering. This issue can be solved in multiple ways as demonstrated throughout the examples above. There can be implemented surveys to build knowledge or if there is pre-existing knowledge like a collection of user demographic information. In general these issues are solved with content-based techniques such as using user personalization[17].

### Shilling Attacks

It is possible for malicious users or competitors to enter into a system and feed false ratings on some items in order to increase or decrease the popularity of those items. As a result of these attacks, recommender systems may experience a decrease in performance and quality. In order to mitigate damage or prevent these attacks, there are some existing detection approaches and methods. Detection approaches may include generic and model-specific attributes, prediction shifts, and hit ratios. The extent of these types of attacks can be determined by a number of factors, including the intent of the attack, the size of the attack, and the level of knowledge required to launch the attack. This could turn out to be a very real threat to our project under the assumption that it was a fully developed application and implemented in a business oriented industry [17].

### Privacy

As always when considering human related information, there is the concern of privacy leaks. The most common way to minimize these situations, would be by implementing cryptographic methods, such as encryption and decryptions of data and data transfer[17].

## **Conclusion**

To conclude this short section, our main concern with building recommender systems would be involving user data and assessing the necessary risks and mitigation techniques to prevent unfortunate situations. With the short amount of time we have in order to find a solution, it would be in our best interest not to focus too much time on building an advanced version of this portion of the solution and pay our attention to other parts of the overall problem in order to demonstrate a part of the solution. However, if a proper recommendations system were to be implemented to the solution, an initial idea could consist of a hybrid between CF and CB to prevent cold start problems, with advanced cryptographic methods to prevent data interventions.

## Chapter 3

# Problem statement

Throughout the problem analysis, we have learnt more about the problem at hand, and we now have a better understanding. We have investigated other competitors, and found similarities and differences on the bargain sites that are already existing. Here is the problem statement we are going to examine:

**How can a software-based solution help students with saving money on grocery shopping by providing the cheapest option and displaying discounts from stores, based on user-given inputs?**

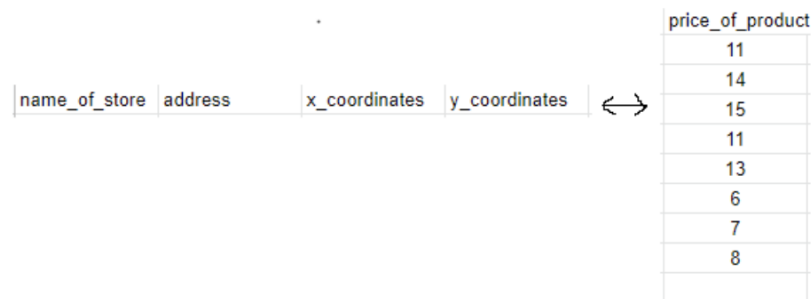
## Chapter 4

# A software based solution

In this section, we will describe the requirements for our program, which is designed to find the cheapest supermarket within a certain range for a given shopping list. The program will use a user profile with variables such as name, location, and transportation options to make recommendations. The program is session-based and does not save any user data. A text file containing the shopping list will be imported during the simulation. The program will use offline, self-made data for testing purposes, including information about the groceries and prices at each supermarket. The program will calculate the distance between the user and each supermarket using coordinates and the user's chosen mode of transportation. All data will be stored in a database that can be accessed and processed by the program. When the program is launched, the user will be prompted for their name, location, and transportation options, and the program will calculate the cheapest supermarkets within the specified range. The program will output the name of the store, the total price of the groceries, and the distance from the user's location.

## 4.1 Program requirements

In order to solve the problem, we must establish some requirements for the program. Ultimately, the application will find the cheapest supermarket within a certain range for the sum of products specified. A user profile containing variables such as name, location, and transportation options will be used in combination with a shopping list to recommend the cheapest supermarkets. As the program is session-based, no user data will be saved. A text-document containing the user's shopping list will be fetched into the program during simulation. Because all data used by the program is offline and self-made, it can only be used for testing purposes. It includes information about the groceries that are available in each supermarket as well as their prices. To calculate the distance between the user and the supermarket, coordinates (latitude/longitude) will be used to set the supermarket's real location. The distance will be the length of the straight line between the two coordinates. In order to facilitate simulations, all of this data will be stored in databases that can be accessed by the program. There will be two text-based databases for each store included in the program, one for providing store information and one for providing grocery prices.



**Figure 4.1:** Representation of our database

Upon launching the program, you will be asked to enter a name and a location in coordinate format. Once you have selected your transport option, the program will ask you how far you are willing to travel. In the program, four modes of transportation will be offered: on foot, by bicycle, by motor vehicle, or by bus. Having calculated the distance between the user and each store, the program will eliminate all stores outside of the perimeter you have set. Upon retrieving the shopping list from the `src` folder, the program begins searching for prices at each supermarket. In all of the stores, the same products are offered and the prices for all items are listed in the same order. Once this is completed, the system will calculate the total of all your groceries in each store and search for the specific products that are on sale that match the user's shopping list. The following information will then be printed to the console:

%name\_of\_store% | %total\_price% | %distance%

%product\_name%: %product\_price%

%product\_name%: %product\_price%

A user named Bob, who lives somewhere in Aalborg, has chosen walking as his method of transportation. He has set the maximum distance to one kilometer since he does not wish to walk more than one kilometer. His shopping list is ready to be imported. The program will produce the following output:

*Hi Bob! The following are the stores with the lowest prices within a 1-kilometer radius of your residence:*

Rema 1000 | Price: DKK 395 | Distance: 0.9 km

Tomato: 5 kr

---

Netto | Price: DKK 410 | Distance 0.3 km

Carrots: 10 kr

---

LIDL | Price: DKK 425 | Distance 0.6 km

## 4.2 Program design

As shown in the flowchart, the process begins with the "Program starts" box, followed by the "Getting user data" box, which enters the data into the system. A variety of methods are used to process the data, and finally, a printout of the user's shopping list is generated. The flowchart illustrates the sequence of steps and the flow of the process with arrows.

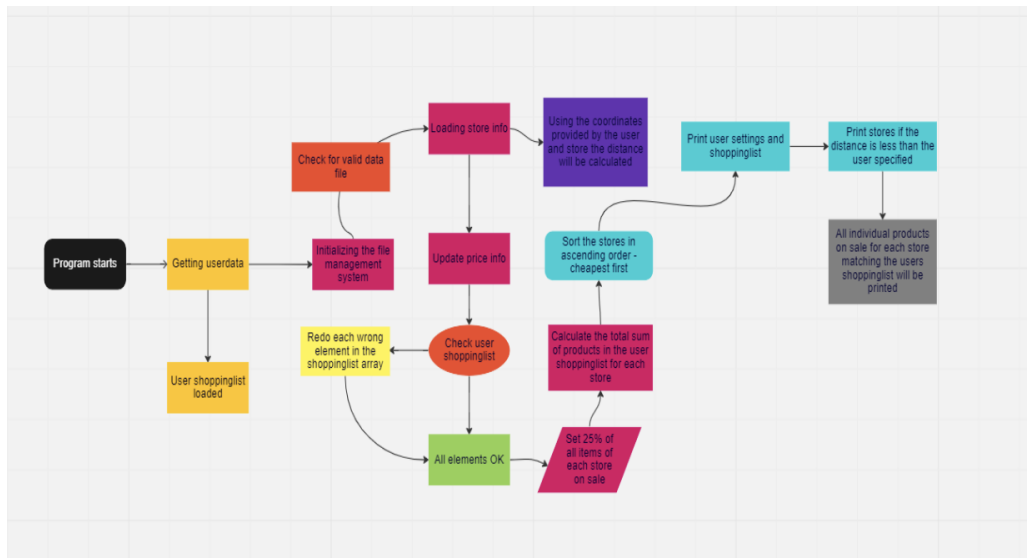
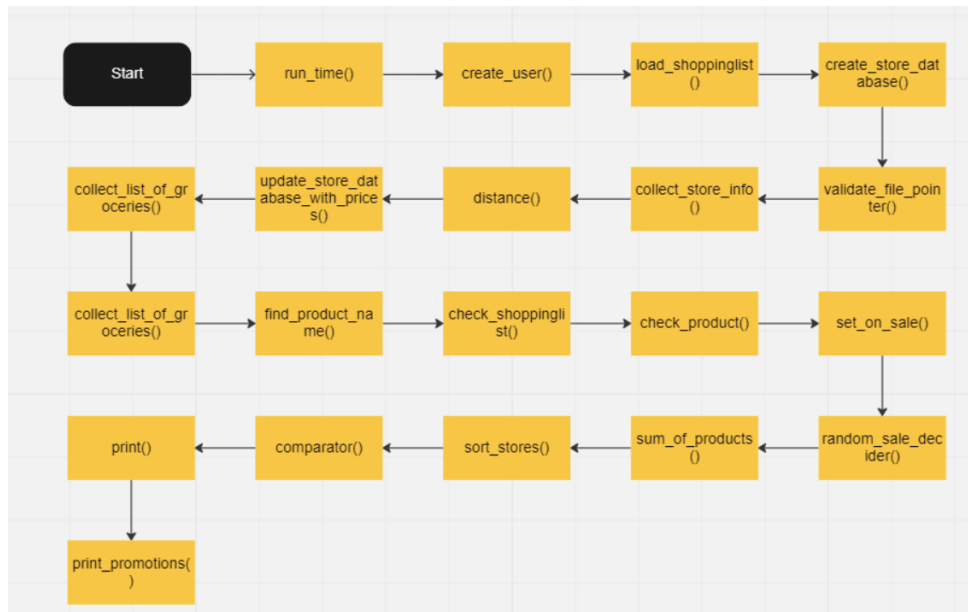


Figure 4.2: Flowchart of our program design

It is much easier to design and document a program if you visualize the flow of the program with a flowchart, as well as helping you to understand the program's flow. In addition, it facilitates communication between team members, provides a greater overview of the program, and helps identify problems and suggest improvements when necessary.





**Figure 4.3:** Flowchart of the functions and their connections in our program

As with the previous flowchart, this chart can also be used to get a better understanding of our functions and their connections, which is particularly useful for spotting errors and simplifying the overall workflow.

## Chapter 5

# Implementation of the program

A detailed explanation of each function will be provided in this section. We will also discuss the input and output data the functions operate on, as well as the specific tasks they perform. As part of this section, we will provide examples of how the functions are used and how they interact with each other to accomplish the program's objectives. We will also examine the functions in detail and how they are implemented

### 5.1 Structs

Structures are used to group several variables into one place. Each variable in the structure is known as a member of the structure. Unlike an array, a structure can contain many different data types. Structures are a vital part of the program and they're seen as the central part of the program's data handler.

Structure variables can be declared with structure declaration by naming a variable after the curly brackets, or as a regular data type without naming any variable.

It's important to note that structure members cannot be initialized with declarations, as memory only gets allocated when variables of the structure are created.

Structure members are accessed using the dot operator (.) or through pointers with (→) operator[34].

In the following section, we will go through our designed structures for the solution of the project. These are located in main.h. It should be noted that we are using typedef keywords for both structures to customize the data type name with the declared structure variable. This also minimizes keystrokes when creating a variable with a datatype of our defined structures.

### 5.1.1 main.h

The store structure is defined inside “main.h”. It consists of 9 members of 3 different data types. We have information related to the store such as name, address, sum, distance and coordinates. We also have information stored in arrays regarding groceries or products, such as product cost, product name and a sales variable determining whether a product is on sale or not. The array members of the structure have sizes defined by a macro definition or a standard value of 30 describing the amount of characters it can consist of. The macro definition MAX\_PRODUCTS consists of a value, representing the max amount of products available.

```
1  typedef struct store
2  {
3      double    longitude, latitude;           //Coordinates of store
4      char      name[30];                     //Name of store array
5      char      address[30];                  //Phyiscal address of store
6      double    sum;                          //Sum of products in shoppinglist
7      double    distance_from_user;           //Distance from user
8
9      double    product_cost[MAX_PRODUCTS];   //Product cost array
10     char      *product_name[MAX_PRODUCTS];   //Product name array
11     int        product_onSale[MAX_PRODUCTS]; //Product sale array
12 } t_store_db;
```

The userdata structure is defined inside “main.h”. It consists of 6 members of 3 different data types. The structure consists of information related to the user such as name, coordinates, amount of products in the cart, transport option and the max traveling distance.

```
1  typedef struct userdata
2  {
3      char    name[30];           //Name of the user array
4      double  longitude;          //Longitude coordinate of the user
5      double  latitude;           //Latitude coordinate of the user
6      int     amount_of_products_in_cart; //Amount of products in shoppinglist
7      int     transport_mode;      //Transport option of the user
8      double  max_traveling_distance; //Max traveling distance
9  } t_userdata;
```

## 5.2 Distance function

This function takes two sets of coordinates (user and store location) and calculates the distance in kilometers between the two coordinates.

This function is located in "utilities.c" from line 37 to 59

```
1  double distance(double lat1, double lon1, double lat2, double lon2)
2  {
3      if ((lat1 == lat2) && (lon1 == lon2))
4      {
5          return 0;
6      }
7
8      double theta = lon1 - lon2;
9      double dist =
10          sin(deg2rad(lat1)) * sin(deg2rad(lat2)) +
11          cos(deg2rad(lat1)) * cos(deg2rad(lat2)) *
12          cos(deg2rad(theta));
13
14      dist = acos(dist);
15      dist = rad2deg(dist);
16
17      dist = dist * 60 * 1.1515;
18      dist = dist * 1.609344;
19
20      return dist;
21 }
```

First of all it should be noted that some of the functions mentioned below are from the `<math.h>` library which is a standard library in C.

The function takes four double parameters: `lat1`, `lon1`, `lat2`, `lon2`. These parameters are the x and y values of the coordinates of both the user and the specific store.

The first task of the function is to check if both coordinates are the same. If so the function returns 0.

If not the function will determine the value of theta by subtracting the parameters `lon1` and `lon2` with each other. The haversine formula is then used which calculates the great-circle distance between two points - that is, the shortest distance over the earth's surface[6]. This value is then set to the variable `dist`. Afterwards the function `acos` with `dist` as its input is used to calculate the inverse cosine in radians. The next step is using the function `rad2deg` which converts radians to degrees. This value can then be converted into miles, and then into kilometers.

The return value will be the distance between the two coordinates in kilometers.

## 5.3 User profile

This function creates a user object by prompting the user for input. Information is gathered about the user and their shopping list is collected through another function.

The function is located in "runtime.c" from line 285 to 318.

```
1  t_userdata create_user()
2  {
3      t_userdata session;
4
5      printf("\nEnter name please: ");
6      scanf(" %s", session.name);
7
8      printf("\nPlease enter your location in a coordinate format (x, y): ");
9      scanf(" %lf, %lf", &session.longitude, &session.latitude);
10
11     printf("\nPlease enter the number of your preferred mode of transport:"
12            "\n(1) On foot\n(2) Bike\n(3) Car\n");
13     scanf(" %d", &session.transport_mode);
14
15     printf("\nHow far are you willing to travel %s in kilometers: ",
16            transport_names[session.transport_mode - 1]);
17     scanf(" %lf", &session.max_traveling_distance);
18
19     char filename[30] = "shoppinglist.txt";
20     FILE *shoppingList;
21     shoppingList = fopen(filename, "r");
22
23     validate_file_pointer(shoppingList);
24
25     session.amount_of_products_in_cart = load_shoppinglist(shoppingList);
26     return session;
```

First, the function declares a struct variable of type `userdata`. After that, the function prompts the user to enter different information such as the name of the user, the location, and the preferred method of transportation. Using `scanf`, this information is scanned into the members of the struct variable `session`.

After the user has provided the console with the necessary information, the program proceeds to declare a file pointer that points to the file containing the user's shopping list. Using this file pointer, `load_shoppinglist` will load the shopping list and return the total number of products found in the shopping list. As a result of the return value of the function, the member `amount_of_products_in_cart` of the struct `session` is set accordingly.

The return value of the function is the struct variable `session`.



## 5.4 Store management.c

### 5.4.1 Validate file pointer and store\_db

This function checks whether or not a file is accessible and will exit if it's not accessible.

The function is located in "utilities.c" from line 12 to 19.

```
1 void validate_file_pointer(FILE * file)
2 {
3     if (file == NULL)
4     {
5         perror("Unable to open file");
6         exit(EXIT_FAILURE);
7     }
8 }
```

A file pointer is passed into this function as a parameter and an if-statement is run to determine whether the file pointer equals NULL. In the event that this statement is true, the program will display an error message and exit. Because this function has a type of void, it returns nothing.

### 5.4.2 Create store database

The purpose of this function is to create a dynamically allocated array of each store's information and products and to use sister functions to collect the data and apply it to each member of the struct.

The function is located in "store\_management.c" from line 13 to 36.

```
1  t_store_db * create_store_database(t_userdata session)
2  {
3      char filename[20];
4      FILE *db;
5      t_store_db* arrayOfStoreInfo = malloc(MAX_STORES *
6                                          sizeof *arrayOfStoreInfo);
7
8      for (int k = 0; k < MAX_STORES; k++)
9      {
10         sprintf(filename, "%d_info.txt", k);
11         db = fopen(filename, "r");
12         validate_file_pointer(db);
13         arrayOfStoreInfo[k] = collect_store_info(db);
14         arrayOfStoreInfo[k].distance_from_user = distance(session.longitude,
15                                                         session.latitude, arrayOfStoreInfo[k].longitude,
16                                                         arrayOfStoreInfo[k].latitude);
17         fclose(db);
18     }
19     update_store_database_with_prices(arrayOfStoreInfo);
20     return arrayOfStoreInfo;
21 }
```

The first task of the function is declaring two variables - a char array called filename and a file pointer called db. Then the function malloc() is used to create a dynamically allocated array of type t\_store\_db called arrayOfStoreInfo with the size of MAX\_STORES.

In the standard program, MAX\_STORES is defined as 5, meaning the array can hold up to five groceries\_db structures.

A for loop is initialized to run while k is less than 5. For each loop the function `sprintf()` is used to change the variable `filename`. The format is `“%d_info.txt”` with `%d` being the format tag which gets updated in each loop cycle by the index variable `k`.

The loop now proceeds to make the file pointer `db` point to the file identified by the char array variable `filename` using the function `fopen()` in read-mode. The file is then validated as accessible by the function `validate_file_pointer()`.

The function `collect_store_info()` is then called with the file pointer `db` being its input. This will update each element of the array defined by `arrayOfStoreInfo` and its members: `name`, `address`, `x_coordiantes`, and `y_coordinates` of the array using the count `k`. The updated members of each element of the array is then used to call the distance function which updates the member `distance` with the distance between store and user. At last the function `fclose()` is used to close the file and then the loop reruns until all stores are loaded.

The function's return value will be the dynamically allocated array `arrayOfStoreInfo`.

### 5.4.3 Collect store information

This function scans a file, stores the data into a variable of type `t_store_db`, and returns it.

The function is located in "store\_management.c" from line 110 to 117.

```
1 store_db collect_store_info(FILE *db)
2 {
3     store_db storeInfo;
4
5     fscanf(db, "%s %s %lf %lf", storeInfo.name,
6           storeInfo.address,
7           &storeInfo.x_coordinates,
8           &storeInfo.y_coordinates);
9
10    return storeInfo;
11 }
```

The function takes a file pointer as its parameter. The first task of the function is declaring a variable called `storeInfo` by type `t_store_db`.

The function `fscanf()` is then used to scan data from the file pointer `db` using a specific format into the members of the variable `storeInfo`. In our case the format is `"%s %s %lf %lf"` meaning that it looks for: string, string, double, double.

The function's return value will be the variable `storeInfo` containing the specific store information.

#### 5.4.4 Products, list of groceries and prices

This function updates the given parameter `arrayOfStoreInfo` of type `t_store_db` with each store's grocery prices and names using sister functions.

The function is located in "store\_management.c" from line 43 to 56.

```
1 void update_store_database_with_prices(t_store_db arrayOfStoreInfo[])
2 {
3     char filename[20];
4     FILE *db;
5
6     for (int i = 0; i < MAX_STORES; i++)
7     {
8         sprintf(filename, "%d_groceries.txt", i);
9         db = fopen(filename, "r");
10        validate_file_pointer(db);
11        collect_list_of_groceries(db, arrayOfStoreInfo, i);
12        fclose(db);
13    }
14 }
```

The function has one array parameter of type `t_store_db` called `arrayOfStoreInfo`.

The first task of the function is declaring two variables - a char called `filename` and a file pointer called `db`.

A for loop is initialized to run while `k` is less than what `MAX_STORES` is defined to, and just like the other function `create_store_database()` the same principles of `sprintf()`, `fopen()`, and `validate_file_pointer()` are used here and will therefore not be explained again.

A function called `collect_list_of_groceries()` is called that will update each store's grocery prices and names. The function takes three inputs: the file pointer `db`, the parameter `arrayOfStoreInfo`, and the current count of the loop an integer "`i`" to indicate the store number to the function. Following successful updates of the members of the `arrayOfStoreInfo` by the function `collect_list_of_groceries()`, the file is closed by the function `fclose()`, and the loop is rerun until all the groceries from each store have been loaded.

### 5.4.5 Collect product prices and names

This function will scan price data from a given file into the given array parameter `arrayOfStoreInfo` of type `t_store_db` and call another function to set each product's name.

The function is located in "store\_management.c" from line 95 to 102.

```
1 void collect_list_of_groceries(FILE *db, t_store_db arrayOfStoreInfo[],
2                               int number)
3 {
4     for (int i = 0; i < MAX_PRODUCTS; i++)
5     {
6         fscanf(db, " %lf", &arrayOfStoreInfo[number].product_cost[i]);
7         arrayOfStoreInfo[number].product_name[i] = find_product_name(i);
8     }
9 }
```

This function has three parameters: a file pointer pointing at the specific store, an array of type `t_store_db`, and an integer indicating the store number.

The first task of the function is initializing a loop that runs through all available products. In the standard program, `MAX_PRODUCTS` is defined as 100 meaning that the loop will loop 100 times. The function `fscanf()` is then used to scan each price of a product into the member "product\_cost" in the element of the array `arrayOfStoreInfo` defined by the parameter `number`. Next, the function `find_product_name` is called using the count of the loop `i` as its input. This function uses the same principles as before, updating the member "product\_name" in the element of the array `arrayOfStoreInfo` defined by the parameter `number`.

### 5.4.6 Find product names

The function runs through a datafile containing all of the product names and returns the product on the line specified by the parameter id.

The function is located in "store\_management.c" from line 64 to 86.

```
1  char * find_product_name(int id)
2  {
3      char *temp = malloc(30 * sizeof(char));
4      FILE * file = fopen("productslist.txt", "r");
5      validate_file_pointer(file);
6
7      for (int i = 0; i < id; i++)
8      {
9          fscanf(file, " %*s");
10     }
11
12     fscanf(file, " %s", temp);
13     fclose(file);
14
15     return temp;
16 }
```

This function is a pointer of type char and has an integer called id as its parameter.

In the first step of the function, malloc is called to allocate memory for the char pointer temp. The amount of memory allocated is equal to the amount of characters multiplied by the size of the data type, which in this case is a char. So there would be allocated 30 char data types for the temp pointer. Then we define a file pointer variable with the name "file" and point it to the function fopen() which opens the file productslist.txt, which contains the grocery names. All stores have the same products and use the same order in the program. Using validate\_file\_pointer(), we verify that the file is accessible.

A for loop is initialized to run while i is less than the parameter id. This loop is used to iterate through the lines using fscanf without scanning the strings into a variable until the

loop count reaches id. After the loop, `fscanf()` is used to scan the specific line by the parameter id containing the product name into the variable `temp`.

As an example, the function `collect_list_of_groceries()` needs the product name of item 17, it calls this function with the input of 17. The loop will scan up to 16, and then after the loop `fscanf()` is used to scan line 17 of the file into `temp`. At last the function `fclose()` is called to close the file and `temp` is returned.



## 5.5 Sorting algorithm and comparator

### 5.5.1 Sorting algorithm

The two following functions, `sort_stores` and `comparator`, are integrated with each other. Their purpose is to sort the stores from the `store_info` array according to total price in an ascending order. These functions are located in "run\_time.c"

```
1 void sort_stores(t_store_db store_info[], int stores_amount)
2 {
3     qsort(store_info, stores_amount, sizeof(t_store_db), comparator);
4 }
```

The function `sort_stores` takes the array `store_info` as input. This array contains all the store structs with their variables, which are declared in the struct type, `t_store_db`.

The function is only used to call the `qsort` function from the C library. `Qsort` takes four input parameters. The first parameter is the base, which is the array, `store_info`. The base is a pointer to the first element that needs to be sorted in the array. The second parameter is the number of items in the array that needs to be sorted. We have a macro definition `MAX_STORES` as our number. The third parameter is the size of each element in the array. We have used the C operator "sizeof" to compute the size of an element in our array in bytes. One element in the array is the size of the struct type `t_store_db`. The compare function is a function pointer and is called multiple times until the list is sorted [26].

The compare function, `comparator`, takes two pointers, `p1` and `p2`. These two variables point to two different locations in the array each time the compare function is called. The comparator function returns an integer whenever it is called. Since we would like to sort our array in ascending order, we have assigned the following return values:

- 1 : The qsort function knows that p1 goes after p2 in the array.
- -1 : The qsort function knows that p1 goes before p2 in the array.
- 0 : The qsort function knows that the two elements are equal and will not change their position in the array.

### 5.5.2 Comparator

The two pointers are being declared in two local variables, store1 and store2, whenever the compare function is called. The function contains an if else statement ladder. This statement starts by comparing the sum from store1 with the sum from store2. The function will return 1 if the sum from store1 is greater than the sum from store2. If this is not true, the if-statement will continue to the next comparison, which checks if the sum from store1 is less than the sum from store2. If that is true, the function will return -1. If none of the first two comparisons are true, the if-statement will execute the last step, which is to return 0 [9].

We have chosen to use quicksort as our sorting algorithm because it is the fastest. The compare function is approximately called  $N \cdot \log_2(N)$  times, where  $N$  is the number of elements to be sorted. We have a maximum of 5 elements to be sorted, since our MAX\_STORES variable is set to be 5. Our comparator function can therefore be called  $5 \cdot \log_2(5) \rightarrow 12$  times. In the worst case scenario, the comparator function is called  $N^2$  times  $\rightarrow 5^2 = 25$  times [2].

```
1  int comparator (const void * p1, const void * p2)
2  {
3      t_store_db * store1 = (t_store_db*)p1;
4      t_store_db * store2 = (t_store_db*)p2;
5
6      if (store1->sum > store2->sum)
7          return 1;
8      else if (store1->sum < store2->sum)
9          return -1;
10     else
11         return 0;
12 }
```

## 5.6 Random\_sale\_decider and set\_on\_sale

### 5.6.1 Random function

The purpose of this function is to return 1 with a 25% chance and 0 with a 75% chance.

The function is located in "run\_time.c" from line 151 to 162.

```
1  int random_sale_decider()  
2  {  
3      int x = rand() % 4;  
4      if (x == 1)  
5          return 1;  
6      return 0;  
7  }
```

The function `random_sale_decider` has the purpose of generating a 1 in 4 chance of returning a "1". The `random_sale_decider` uses `rand()`, a C library function, which is a Pseudo Random Number Generator used to return an integer between 0 and `RAND_MAX`. The maximum value "`RAND_MAX`", is a constant defined by the library to be 32767. [11]

The function `srand(time(NULL))` grants the `rand()` function a random seed based on the current time, when called. By using "`rand()%4`" is the random seed generated by the `srand()` function divided by four, and the remainder of this integer division is assigned to "`x`". This will then assign "`x`" with a random number between 0 and 3, and if the number becomes 1, 1 will be returned, otherwise 0 will be returned, making it a 25% chance of returning a 1.

A disadvantage that the `rand()` function has, is that it uses a linear congruential generator, which does not produce completely random results. Because of the usage of this type of pseudo random number generator algorithm, can the result be predicted if the seed value is known [5]. [19] Therefore it is important that the `srand()` is being called, before using the `rand()` function, as it is at the start of `run_time.c`.

### 5.6.2 Set products on sale

The purpose of this function is to set random products on sale by updating the member `product_onSale` to 1.

The function is located in "run\_time.c" from line 132 to 144.

```
1 void set_on_sale(t_store_db store_info[])
2 {
3     for (int i = 0; i < MAX_STORES; i++)
4     {
5         for (int k = 0; k < MAX_PRODUCTS; k++)
6         {
7             store_info[i].product_onSale[k] = random_sale_decider();
8         }
9     }
10 }
```

This function sets 25% of the products available in each store on sale.

An array of type `t_store_db` is provided as a parameter, which is a pointer to all the stores that have been loaded.

A nested for loop is initialized, with the outer for loop's condition set to until all stores have been loaded. The inner for loop runs until every product has been loaded, as indicated by the loop's condition. During the inner loop, the function `random_sale_decider` is called on the parameter `product_onSale`. It determines whether or not a product is listed as a sale product by updating the member to either a 0 or a 1.

This function has no return value since it is of type `void`.

## 5.7 Load\_shoppinglist and sum\_of\_products

### 5.7.1 Load user shoppinglist

In this function, the user's shopping list is scanned into a global array and the number of items found is returned.

The function is located in "runtime.c" from line 326 to 348.

```
1  int load_shoppinglist(FILE *list)
2  {
3      int i = 0;
4      int k;
5
6      while (!feof(list))
7      {
8          fscanf(list, "%s", user_groceries[i]);
9          i++;
10     }
11
12     k = i;
13     while (k < 100)
14     {
15         strcpy(user_groceries[k], "0");
16         k++;
17     }
18
19     return i;
20 }
```

The function takes a file pointer as its parameter which points to the data file containing the user's shopping list. A while loop is used to scan each of the products in the data file into the global array `user_groceries`. The while loop will run until it reaches the end of the file as indicated by the condition of the loop. Since the global array can hold up to 100 arrays of characters with a size of 30 and it is unlikely that the user will have all of the products in the array, a new while loop is initialized to set the remainder of the elements in the array to zero.

The global array `user_groceries` has been updated with the user's shopping list from the data file. The rest of the program can now utilize this information.

At the end, the function returns the integer `i` indicating the number of products contained in the global array.

### 5.7.2 Sum of user products for each store

This function assigns values to the sum variable for each element in the store\_info array.

The function is located in "run\_time.c" from line 91 to 125.

```
1 void sum_of_products(store_db store_info[])
2 {
3     double sum;
4     int j;
5
6     for (int i = 0; i < MAX_STORES; i++)
7     {
8         j = 0, sum = 0;
9
10        for (int k = 0; k < MAX_PRODUCTS; k++)
11        {
12            if (strcmp(user_groceries[j],
13                store_info[i].product_name[k]) == 0)
14            {
15                sum += store_info[i].product_cost[k];
16                k = 0;
17                j++;
18            }
19        }
20        store_info[i].sum = sum;
21    }
22 }
```

The sum\_of\_products function consists of a nested loop. The outer loop iterates through all of the available stores defined by MAX\_STORES. The inner loop iterates through all of the available products in each store defined by MAX\_PRODUCTS. The if statement inside of the inner loop compares each product in the user\_groceries array with each product in the total list of products inside of the store\_prices array using the stringcompare function. The stringcompare function returns 0 if there are no differences between both parameters. If that is the case then the function goes on to increment the sum variable with the price of the matching product in that specific store. It resets variable k and increments variable j by 1, in order to



compare the next product in the shopping list with all the products in the store and continues like this. At the end of each cycle of the outer loop it assigns the accumulated sum value from the inner loop to the sum variable of the current indexed store.

## 5.8 Output

### 5.8.1 Print

This function prints information about a user, their shopping list, and stores within a certain distance from the user's location.

The function is located in "run\_time.c" from line 246 to 278.

```
1 void print(t_userdata user, t_store_db store_info[])
2 {
3     printf("\nYour name is set to: %s "
4           "\nYour location is set to: %lf %lf"
5           "\nYour preferred mode of transport is set to %s"
6           "and your max travel distance is set to %lf km."
7           "\n\nYou have %d item(s) in your shopping list:\n",
8           user.name, user.longitude, user.latitude,
9           transport_names[user.transport_mode - 1],
10          user.max_traveling_distance,
11          user.amount_of_products_in_cart);
12
13     for (int i = 0; i < user.amount_of_products_in_cart; i++)
14     {
15         printf("\n%s", user_groceries[i]);
16     }
17
18     printf("\n\nStores found within %lf km from your location:\n",
19           user.max_traveling_distance);
20     for (int i = 0; i < MAX_STORES; i++)
21     {
22         if (store_info[i].distance_from_user <= user.max_traveling_distance)
23         {
24             printf("\n%s %s | TOTAL PRICE: %.2lf | %.2lf KM AWAY\n",
25                   store_info[i].name, store_info[i].address,
26                   store_info[i].sum, store_info[i].distance_from_user);
27             print_promotions(store_info, i);
28         }
29     }
```

```
30     printf("-----\n");
31 }
32 }
```

This function prints information about a user, their shopping list, and stores within a certain distance from the user's location.

It expects two arguments: a `t_userdata` struct containing information about the user, and an array of `t_store_db` structs containing information about stores.

The function first prints the user's name, location, preferred mode of transportation, and maximum travel distance. Then it prints the items in the user's shopping list. Next, it prints the stores that are within the user's maximum traveling distance. For each store, it prints the store's name, address, total price, distance from the user, and any promotions the store is offering.

Finally, it prints a separator line after each store.

### 5.8.2 Print promotions

This function prints promotions for a given store.

The function is located in "run\_time.c" from line 204 to 238.

```
1 void print_promotions(t_store_db store_info[], int store)
2 {
3     int i, j = 0;
4
5     for (i = 0; i < MAX_PRODUCTS; i++)
6     {
7         if (strcmp(user_groceries[j], store_info[store].product_name[i]) == 0)
8         {
9             if (j > (sizeof(user_groceries) / sizeof(user_groceries[0])))
10            {
11                break;
12            }
13
14            j++;
15
16            if (store_info[store].product_onSale[i] == 1)
17            {
18                printf("%s is on sale for %.2lf DKK!\n",
19                    store_info[store].product_name[i],
20                    store_info[store].product_cost[i]);
21            }
22
23            i = 0;
24        }
25    }
26 }
```

It expects two arguments: an array of `t_store_db` structs containing information about the stores, and an integer representing the index of the store in the array.

The function first initializes two variables, `i` and `j`, which will be used to iterate over the

products in the store and in the user's shopping list, respectively. It then iterates over the products in the store and checks if each product is also in the user's shopping list. If it is, validates that the index of the shopping list is not exceeding the size of the shoppinglist, in which it would cause a segmentation fault and it also increments the index for the shopping list and checks if the product is on sale. If the product is on sale, it prints a message indicating that the product is on sale.

Finally, it resets the product index to 0 and continues iterating over the products in the store.

## Chapter 6

# Discussion

In this chapter we will reflect on potential improvements and consider the limitations and possibilities of the project. We will discuss what would be necessary for it to fully succeed, as well as potential future developments for the program. We will also touch on the topics that we have talked about but not implemented in our solution and reflect on our decisions.

### **Limitations of the program:**

One of the limitations of our program is that it relies heavily on premade data, such as static array sizes and a fixed product list. This can make the program inflexible and less efficient, as it cannot easily adapt to changes in the data.

To address this limitation, we could have implemented more dynamic and flexible programming techniques in our program. For instance, instead of using macro definitions to define the size of the total product list, we could have calculated the list size dynamically. This would allow us to easily add more products to the list in the future without having to modify the program. Similarly, we could have made other aspects of the program more flexible and dynamic, such as the sizes of arrays, in order to optimize its performance.

While we made these choices for demonstration purposes in order to simulate the solution, it would be an improvement to make the program more dynamic in order to optimize its performance and adaptability. This would make it more efficient and able to handle changes in the data more effectively, increasing its flexibility and usability for users. By implementing dynamic and flexible programming techniques, we can address the limitations of our program and enhance its performance, efficiency, and adaptability. This can ultimately improve the user experience and make our program more competitive in the marketplace.

### **Factors influencing the decision to exclude certain features in our project**

We made the decision not to include recommender systems and real-time data collection in our project for a number of reasons. One of the main considerations was time and resources. Building these features would have required additional time and resources that we did not have available due to the limited duration of the project. We had to make trade-offs and prioritize certain aspects of the project in order to deliver a functional and useful product within the available time and resources.

Another reason for not including these features was a skill gap in terms of our programming capabilities and experience. Implementing recommender systems and real-time data collection can be complex and require advanced programming skills. While we had a strong team with a range of skills, we may have determined that building these features would have been beyond our current capabilities. In order to ensure that everyone on the team could contribute and learn effectively, we decided to limit the scope of the project to what was within our current skills and experience.

In our program, we have utilized self-made offline data, which has been helpful in the early stages of development. However, this data type would not benefit real users who want to find the lowest prices in actual stores. Because we are still in the early stages of our programming journey, we have decided to focus on what we know we can accomplish, which does not include creating a real-time database of products available in actual stores. If the program were to be further developed in the future, we would like to utilize a bot to scrape information from each store's website to update the database automatically with the latest prices for each store. It is important to note that our current program contains the same products in all stores, which is not the case in grocery stores. This could cause issues with the accuracy and relevancy of the recommendations made by the program.

Ultimately, the decision to exclude certain features or functionality is a common aspect of any project. By carefully considering the available time and resources, as well as the skills and experience of the team, we were able to deliver a functional and useful product while still allowing everyone on the team to contribute and learn effectively.

### **Improving and enhancing our project**

There are several ways in which our project could be improved and enhanced in the future. One key area to focus on would be the inclusion of recommender systems and real-time data collection, which are features that many competitors use to stay at the top of the food chain. Adding these features would enable our project to provide more personalized and relevant recommendations based on a wide range of user preferences, such as distance, food nutrients, allergies, and other factors. By taking into account these preferences, our project could help users make more informed and tailored purchasing decisions, resulting in a better user experience overall.

Another potential improvement would be to make the project web-based instead of an application made in C. This would make it more accessible and flexible, allowing users to access the program from any device with an internet connection. This could potentially increase the user base and reach of the project, as it would be easier for users to access and use the program from any location.

In addition to the recommendation system and data collection methods, other factors that can contribute to the success and effectiveness of the program include the user interface and user experience, the performance and efficiency of the program, and the accuracy and relevance of the recommendations made. By continually improving and optimizing these areas, our project can provide a more seamless and enjoyable experience for users and help them find the lowest prices for their shopping lists. One potential feature to consider adding to our project is social sharing or community features. By allowing users to share their shopping lists or recommendations with friends or within a community, the project could foster a sense of social interaction and collaboration among users. This could make the shopping experience more enjoyable and rewarding for users.



## Chapter 7

# Conclusion

It is concluded that automatic bargain hunting can provide significant benefits for students by helping them save money on their groceries and prevent overspending. By utilizing technology to identify the best deals and offers at nearby supermarkets, students can make informed decisions about their shopping habits and prioritize sustainable consumption patterns. Additionally, automatic bargain hunting can help students save time and effort by providing a convenient and accessible way to access the best deals and offers. Overall, automatic bargain hunting can support the goals of the United Nations' 12th global goal and contribute to more sustainable consumption and production patterns.

The problem at hand has been identified through a survey, showing the majority of students are interested in searching for bargains through a more time efficient way. Therefore, It is concluded that a software solution to identify bargains and offers is necessary.

It is lastly concluded that the software solution that has been made in this project relates to the problem statement. In terms of the program, considering the time and resources that were invested, we can conclude that the program is slightly fulfilling. If it was to be a useful application in the future, it lacks many complex implementations and components, but given the time frame and our limited skills, it is not realistic for us at the moment. Although we are confident that the program we have developed suits and fulfills the problem statement.

# Bibliography

- [1] Keld Vrå Andersen. *Ukrainsk korn kan få verdenspriserne til at falde igen*. 2022. URL: <https://nyheder.tv2.dk/udland/2022-07-22-ukrainsk-korn-kan-faa-verdenspriserne-til-at-falde-igen-1> (visited on 11/10/2022).
- [2] ayushivadhera. *When to use each Sorting Algorithm*. 2022. URL: <https://www.geeksforgeeks.org/when-to-use-each-sorting-algorithms/fbclid=IwAR3IJYuXVx26QLhxrBSGRvQfLfCE5qqvpKKaG5SBTqTa2RbGUc8Ek> (visited on 12/19/2022).
- [3] Nicholas J. Belkin and W. Bruce Croft. *Information filtering and information retrieval: two sides of the same coin?* 1992. URL: <https://dl.acm.org/doi/10.1145/138859.138861> (visited on 11/07/2022).
- [4] Martin Breuss. *Beautiful Soup: Build a Web Scraper With Python*. URL: <https://realpython.com/beautiful-soup-web-scraper-python/> (visited on 11/13/2022).
- [5] C++ TUTORIAL - RANDOM NUMBERS. URL: <https://www.bogotobogo.com/cplusplus/RandomNumbers.php> (visited on 12/10/2022).
- [6] *Calculate distance, bearing and more between Latitude/Longitude points*. URL: <https://www.movable-type.co.uk/scripts/latlong.html> (visited on 12/16/2022).
- [7] Manisha Chandak, Sheetal Girase, and Debajyoti Mukhopadhyay. *Introducing Hybrid Technique for Optimization of Book Recommender System*. 2015. URL: [https://www.researchgate.net/figure/Comparative-analysis-of-recommendation-techniques-on-the-basis-of-Accuracy\\_fig3\\_274096470](https://www.researchgate.net/figure/Comparative-analysis-of-recommendation-techniques-on-the-basis-of-Accuracy_fig3_274096470) (visited on 11/08/2022).
- [8] Tom Frank Christensen. *Hvad er et API?* URL: <https://www.modified.dk/details/api> (visited on 11/15/2022).
- [9] Cplusplus. *qsort*. URL: <https://cplusplus.com/reference/cstdlib/qsort/> (visited on 12/19/2022).
- [10] Foedevarestyrelsen. *Prisboom på fødevarer: Nye tal viser store besparelser ved at følge kostråd og mindske madspild*. 2022. URL: <https://fvm.dk/nyheder/nyhed/nyhed/prisboom-paa-foedevareer-nye-tal-viser-store-besparelser-ved-at-foelge-kostraad-og-mindske-madspild/> (visited on 11/07/2022).

- [11] Bamdeb Ghosh. *rand() Function in C Language*. 2020. URL: <https://linuxhint.com/rand-function-in-c-language/> (visited on 12/10/2022).
- [12] Ben Lutkevich Alexander S. Gillis. *What is a bot?* 2022. URL: <https://www.techtarget.com/whatis/definition/bot-robot> (visited on 11/12/2022).
- [13] Global Goals. *12. Responsible Consumption and Production*. 2022. URL: <https://www.globalgoals.org/goals/12-responsible-consumption-and-production/> (visited on 11/07/2022).
- [14] Astrid Ildor. *Disse grupper rammer inflationen hårdest – og her kan de søge om økonomisk hjælp*. 2022. URL: <https://www.dr.dk/nyheder/penge/disse-grupper-rammer-inflationen-haardest-og-her-kan-de-soege-om-oekonomisk-hjaelp> (visited on 11/04/2022).
- [15] Sarika Jain et al. *Trends, problems and solutions of recommender system*. 2015. URL: [https://www.researchgate.net/publication/307862659-Trends\\_problems\\_and\\_solutions\\_of\\_recommender\\_system](https://www.researchgate.net/publication/307862659-Trends_problems_and_solutions_of_recommender_system) (visited on 11/08/2022).
- [16] Javapoint. *What is a Dynamic Website?* URL: <https://www.javatpoint.com/what-is-a-dynamic-website> (visited on 11/13/2022).
- [17] Shah Khushro, Zafar Ali, and Irfan Ullah. *Recommender Systems: Issues, Challenges, and Research Opportunities*. 2019. URL: <https://fardapaper.ir/mohavaha/uploads/2019/03/Fardapaper-Recommender-Systems-Issues-Challenges-and-Research-Opportunities.pdf> (visited on 11/08/2022).
- [18] Kin Lane. *Intro to APIs: History of APIs*. 2019. URL: <https://blog.postman.com/intro-to-apis-history-of-apis/> (visited on 11/15/2022).
- [19] *Linear congruential generator*. 2022. URL: [https://en.wikipedia.org/wiki/Linear\\_congruential\\_generator](https://en.wikipedia.org/wiki/Linear_congruential_generator) (visited on 12/10/2022).
- [20] Camilla Lund. *Inflationen er historisk høj: Sådan sparer du penge i supermarkedet*. 2022. URL: <https://minbyaalborg.dk/2022/10/27/inflationen-er-historisk-hoej-saadan-sparer-du-penge-i-supermarkedet/> (visited on 11/05/2022).
- [21] Nationalbanken. *Pressemeddelelse: Danmark og resten af verden: Historisk lav rente i Danmark og faldende rente i USA og Europa*. 2022. URL: <https://www.nationalbanken.dk/da/presse/Sider/2022/09/DNN202230314.aspx> (visited on 11/02/2022).
- [22] Steffen Neupert. *Fødevarer trækker inflationen op – her er dem, der er steget mest i pris*. 2022. URL: <https://nyheder.tv2.dk/business/2022-08-10-foedevarer-traekker-inflationen-op-her-er-dem-der-er-steget-mest-i-pris-0> (visited on 11/02/2022).

- [23] Saheed Opeyemi. *How To Scrape HTML Data For Your Data Needs (And Why)*. 2021. URL: <https://scrapingrobot.com/blog/html-scraping/> (visited on 11/13/2022).
- [24] Laura Skelgaard Paulsen. *Regeringen foreslår huslejestigninger på maksimum 4 procent: Hvad betyder det for studerende?* 2022. URL: <https://www.akademikerbladet.dk/aktuelt/2022/august/regeringen-foreslaar-huslejestigninger-paa-maks-4-procent-hvad-betyder-det-for-studerende> (visited on 11/04/2022).
- [25] Simon Bugge Jensen Nicklas Petersen. *Analyse af effekten af tilbudsaviser*. 2022. URL: <https://www.danskerhverv.dk/siteassets/mediafolder/dokumenter/04-politik/analyse-af-effekten-af-tilbudsaviser---gfk.pdf> (visited on 11/10/2022).
- [26] Tutorials point. *Calculate distance, bearing and more between Latitude/Longitude points*. URL: [https://www.tutorialspoint.com/c\\_standard\\_library/c\\_function\\_qsort.htm](https://www.tutorialspoint.com/c_standard_library/c_function_qsort.htm) (visited on 12/19/2022).
- [27] REMA 1000. 2022. URL: <https://rema1000.dk/avis#catalogs/aSAmbSoH> (visited on 11/13/2022).
- [28] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems: Techniques, Applications, and Challenges*. 2021. URL: [https://link.springer.com/chapter/10.1007/978-1-0716-2197-4\\_1#citeas](https://link.springer.com/chapter/10.1007/978-1-0716-2197-4_1#citeas) (visited on 11/07/2022).
- [29] Baptiste Rocca. *Introduction to recommender systems*. 2019. URL: <https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada> (visited on 11/08/2022).
- [30] *Search engine Definition and Meaning*. URL: <https://www.dictionary.com/browse/search-engine> (visited on 11/07/2022).
- [31] *Shop REMA 1000*. URL: <https://shop.rema1000.dk/> (visited on 11/13/2022).
- [32] Statista. *Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025*. 2021. URL: <https://www.statista.com/statistics/871513/worldwide-data-created/> (visited on 12/15/2022).
- [33] Danmarks Statistiker. *Nedadgående trend i detailsalget fortsætter*. 2022. URL: <https://www.dst.dk/da/Statistik/nyheder-analyser-publ/nyt/NytHtml?cid=39885> (visited on 11/02/2022).
- [34] *Structures in C*. 2022. URL: <https://www.geeksforgeeks.org/structures-c/> (visited on 12/05/2022).
- [35] W3Schools. *C Structures (structs)*. URL: [https://www.w3schools.com/c/c\\_structs.php](https://www.w3schools.com/c/c_structs.php) (visited on 11/13/2022).
- [36] *Web Scraping*. URL: <https://www.imperva.com/learn/application-security/web-scraping-attack/> (visited on 11/12/2022).

- [37] *What is an Intelligent Agent?* URL: <https://www.techslang.com/definition/what-is-an-intelligent-agent/> (visited on 11/12/2022).
- [38] *What is data scraping?* URL: <https://www.cloudflare.com/learning/bots/what-is-data-scraping/> (visited on 11/12/2022).
- [39] Selma Bjørnstad Kvist Åstrøm. *Prisen på mælk har aldrig været højere: Producenter nyder det, så længe det varer*. 2022. URL: <https://www.tvmidtvest.dk/midt-og-vestjylland/prisen-paa-maelk-har-aldrig-vaeret-hoejere-producenter-nyder-det-saa-laenge-det-varer> (visited on 11/10/2022).