COMPUTER SCIENCE
2ND SEMESTER PROJECT
DMAJ0915
GROUP 4

DEVELOPMENT OF A BOOKING SYSTEM

2016

University College of Northern Denmark

Technology and Business

Computer Science

Second semester project

Project members:

Alef Pir

Aurelijus Steigvila

Marius Constantin Untaru

Nils-Asbjorn Frederiksen

Roberta Siaudvytyte

Supervisor :

Henrink Munk Hvarregaard

This project has developed part of a complete calendar system, handling booking, events and orders at a board game cafe. The program has been developed in JAVA, and the UI is using Swing objects. An SQL database system has been used to hold the different data for use in the JAVA program. Apart from the developed system the group has worked in close contact with a local board game cafe to develop a better understanding of business run in real life. Basing the entire business chapter on this small board game cafe business, and associated businesses. This has lead to the development of SWOT analysis, Business cases and other interesting artefacts to use in explaining an unique business model as the one run by the business in this project.

Numer of pages: 53

Numer of characters (no spaces):61.672

Submision date:

07.06.2016

TABLE OF CONTENTS

INTRODUCTION

Nowadays, computer technologies take part of the everyday life of each human being. Most of the technologies are highly developed and available for daily purchase, others are still in process of development and only available for the higher classes. Computer industries keep themselves involved over the past decades in developing newer models for operating systems, for databases or application systems. It is no different in the case of companies which provide services. [1] All these advancements in computer technology has lead to the idea of the paperless office has been sought after, paper being easily eradicated by coffee stains and the like. The pursuit of this new office design is mostly made possible with the further development of computers and in that regard the creation of programs to assist employees without the need of a hard copy. Newer computer inventions like the tablet makes paper even more obsolete because it is now possible to easily carry all necessary documents in such a small device. Now, this is one of the many pursuits of the technology era and one that this project takes on, although on a much smaller scale, in the local game board cafe. [2] A company that is almost working directly against the idea of the paperless reality by taking board games, once the inspiration for computer games, but now a source of income for the company this project is all about. *Aalborg Brætspils Cafe* is indeed pionering the comeback for board games in the Aalborg area, showing the way for other board game cafes like *"Dice 'n Drinks"*. But how is this board game cafe connected to the paperless office, they are certainly not interested in playing on computers instead of with paper and hard plastic, suggesting such a thing might even get you shunned on a bad day. It is the calendar system that needs to be electronic, instead of the current solution with a paper calendar often hit by the odd coffee stain or other liquids.There are many companies that use online booking systems such as restaurants, airlines and more. The main idea behind this is that having an online database which can hold tons of data is simply more advantageous, cheaper and sustainable than having papers to write down bookings. So this project is all about designing and implementing a booking system for the cafe, giving the employees the opportunity to have a more streamlined work day.

Problem statement

How can a java program be developed based on the swing gui objects, which can handle the calendar at a boardgame cafe, as well as handling concurrency and database functions like data handling and transactions, using UP?

1.BUSINESS

1.1 Introduction

The board game café did not just start out as a board game cafe, Michael Christensen, owner of the cafe, started out as a figurine painter, painting warhammer figurines for a fee. He added a few tables to his workshop and people started paying him a small monthly fee to play different figurine games at the tables. This small business startup happened several years ago, and after a while he decided to move to bigger locales to open up the opportunity for more players to join the ranks of figure gamers. He started selling different products for the figure games in the locales where the current cafe is located. These products are also sold in a webshop under the name 9k Gaming. At some point his mom took over the upstairs part of the locales because it was more viable to keep the shop downstairs with the figure gamers. Then she opened an antique store and paid rent each month. When she closed down the antique shop in November 2016 the idea to open a board game cafe in the shop locales was born. The business is currently consisting of three parts, a small webshop, a figure gaming club and a board game cafe.

When the board game cafe started out it had 20 - 35 boardgames, and in the few months of existence it has grown its stock to over 150 board games. A lot of the success the cafe has experienced since its opening in 2015, the owner attributes to a TV2Nord News interview which helped the facebook group get to 1500 likes. Currently it has almost 3000 likes.

1.2 Interview

Since it is this project's main theme, it is important to get information and a better understanding on how *Aalborg Brætspilscafe* operates. To do so, the owner of the cafe was interviewed. The following section will present the interview, discuss the knowledge gained out of it and conclude how it is used throughout this project. For a full transcript of the interview, see appendix 1.

Michael Christensen, the owner of the cafe, is ex-military who opened his business recently. The idea was inspired by his mother, who used to rent the local space from him and introduced him to the idea of a boardgame cafe. Currently he is the one and only owner of *Aalborg Brætspilscafe*, which puts him on top of the hierarchical triangle of the company.

During the interview, Christensen provided some useful information and a number of key points that will be considered:

- The business is increasing. Although he started out with only 20 - 30 games, in a very short period of time the board games owned have increased to over 150. This is good for the company, as it can draw more clients due to the diversity in the products.

- The owner is aware of the main competition (Dice n' Drinks) in the area and based on the interview it seems he handles some things the same way his competition does.

- There are six volunteers currently at the cafe, which are receiving their schedules via Facebook. In his role as a manager of the cafe, he organizes social meetings, where he and his volunteers hang out and eat together. The fact that two of his volunteers suffer from Asperger's disease and him trying to make them feel better at the workplace, illustrates his managerial qualities. Besides the occasional restaurant visits, the volunteers get free soda whenever they are working.

- He owns three businesses at the same time and at the same place. Those are the board game cafe, the gaming club and a webshop which sells miniatures and figures for gaming.

- The board game cafe is sponsored with free games by Geeks and Sundry in exchange for advertising. Besides that, food and drinks are purchased in local stores e.g. Netto, Rema1000, by him or the volunteers.

- When asked how many customers visit monthly, Christensen said he was happy as the business only needs 350 customers a month to stay profitable, and he had 580 at the time of the interview, which was a lot higher than the wanted number.

- Even though the cafe is working efficiently, the owner stated he did not plan expanding in the near future, but possibly after two or more years. The main reason for that would be the small space for the many people who visit the cafe.

- Advertising was a key for his success. It was achieved through conducting a small interview with TV2Nord which led to rapid increase of the Facebook page likes.

As conclusion, the interview gave further insight about *Aalborg Brætspilscafe* which will be used later on when describing important aspects of the business of the company.

1.3 The business trinity of 9k Gaming

Webshop

The owner has made a small website where he is selling models and miniatures for board games such as Warhammer 40k, etc. This was spawned by the gaming club, because a lot of the figure gamers wanted new models, but did not want to pay the high price of the local stores and avoid shipping for small packages, so the webshop was born.

Gaming club

The Gaming club is part of the business trinity of 9K gaming. It is a small club with a few core members paying a monthly fee to be able to play at big tables with several obstacles. For example, ruins, fences, castles so on and so forth, giving the gamers the ability to have exciting terrains for the battle of figurines, without having to invest thousands of kroners into these obstacles. The gaming club started out because of Michael's first business, which was a figure painting company for figure gamers, who wanted the great looking army without the many hours of work. This spawned the gaming club, which delivers a small amount of the annual income for the businesses, but important none the less, because it has been a great resource, both for finding volunteers and for getting the word out about the new board game cafe.

Board game cafe

The owner has his own board game cafe. People can book a table and play around 150 different board games. The customers have to pay 30 kroner per person and then they can play any games they want for the entire day. There are many different types of tables and customers have to use a specific table size, depending on the game they have chosen. The cafe also offers all sorts of drinks and snacks for customers to purchase. As mentioned earlier, food is being supplied by the owner himself. He is trying to avoid buying big quantities of them, due to them having a risk of passing the expiration date. Currently, the cafe has six volunteers that help out people with starting a board game, teaching them the basics and helping with cleaning around the place.

Business structure

The model in Fig. 1 shows the organizational structure with the entire trinity represented. The webshop and gaming club, as explained earlier, are run by the owner himself, but apart from that all three instances can be explained by the traditional hierarchy structure, where power flows from the bottom and ends at the top. During talks with the owner it was made clear that he took in suggestions from his volunteers. But in the end it is totally his decision if something is going to happen or not.

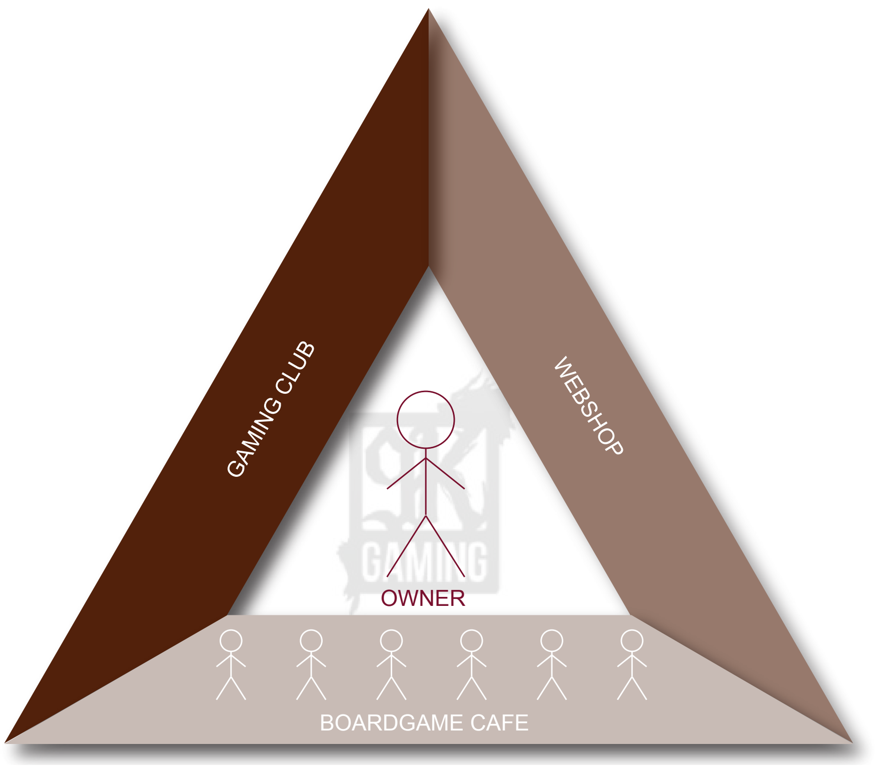


Fig. 1 Organizational structure

Strengths of the traditional hierarchy structure: [3]

- Employees got a clear role in the organisation
- It is easily controlled

Weaknesses of the traditional hierarchy structure:

- Slows down change in idea-rich environments
- Is problematic to change from at a later date

What does this mean?

The company is working with volunteers and a volunteer based employee base is often flimsy at best.[4] The volunteer can suddenly stop working and have no real incentive other than their own interest in the place. The employer has no real hold on employees, this might result in fast exchange in the employee base, so having a clear role for the employee is a good idea, and a simple organizational structure is an easy way to get people into the company. The traditional structure is also easily controlled, so a new volunteer will not do big damage to the organization, by mistake or otherwise, because the volunteer has no real authority. All purchases are made by the owner and the volunteers only got access to a small amount of money in the cashier desk.

When looking at the weaknesses in traditional hierarchy structure and at the board game café, which are an ever changing environment, consistent purchase of new games and different objects for the gaming club as well. There might be wishes from the customer or some of the volunteers that has to go by the owner before it is bought, but as long as the company is at the size it has now it will not be a big problem. But as the interview shows the company is expanding and at some point it might be a problem that everything should be run by the owner, and as seen on the source for weaknesses, one of the problems with this organizational structure is to get away from it once the company is too big for this simple structure.

1.4 Analysis of the competitive situation

This chapter will look into and analyze the competitive situation in Aalborg, thus getting more insight over the main competitor and what they do compared to the *Aalborg Brætspilscafe.* This will be done mainly by looking at the Porter's five forces. It is a great, yet simple tool, which helps understanding the current competitive position of the company this project is working with and its competitors. There is currently two competitors for *Aalborg Brætspilscafe.* A free board game club at Studenterhuset, which is not an actual business, it is a voluntary club, and then there is the main competitor, another board game cafe called *Dice 'n Drinks.*

According to Porter's model, there are five powers that drive the business. Those consist of: [7] [8]

- Suppliers' bargaining power
- Customers' bargaining power
- Threats of new entries
- Threats of substitutes
- Competitive rivalry

These five factors will be looked at and analyzed based on this project's theme.

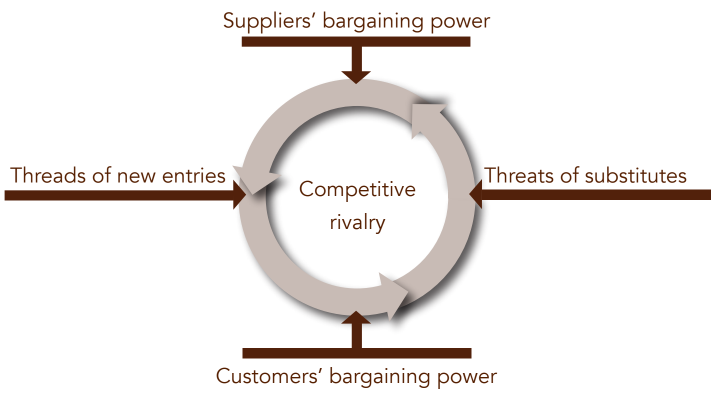


Fig. 2 Porter's five forces

Suppliers Bargaining Power: Suppliers power determines to what extent the suppliers control the business. Those factors vary from uniqueness of the provided supply, the number of suppliers to cost of supplies and more. *Aalborg Brætspilscafe* is sponsored by the YouTube channel *Geek and Sundry* (check section Interview). In exchange for advertising the cafe is getting free board games. However, the suppliers can always terminate the contract or change it in a negative fashion for the cafe. This said, there are many different ways to acquire board games, thus the suppliers don't have big control over the company. Therefore, it can be concluded that the cafe does not rely on one supplier and has the option of many substitutes.

Customers Bargaining Power: This section analyses the customer buying power and the ability of the customer to contribute in bringing the prices in the business down. There are a few factors that can affect the customer power. Such factors that concern the cafe in this project are customer's price sensitivity and knowledge of the product. Based on the interview where the owner stated that he offers discounts for students, it can be concluded that the cafe's target audience is students. For a city like Aalborg which is inhabited by a big amount of students, it is logical that the price is playing a serious factor for the customer. Besides that, knowledge of the product is a serious buyers power. A customer with enough background and research might be aware of the product's benefits, whether it is worth it, will compare its quality and price to the competition and use this knowledge to bargain.

Competitive Rivalry: What this section covers is the analysis of the competition. To get insight it will look into the amount of competitors in the area, the diversity in prices and stock, the difference in the policies and the interaction between the competitors. *Aalborg Brætspilscafe* has one main competitor named *Dice 'n Drinks*. According to their facebook page, *Dice 'n Drinks* opened at the beginning of February 2016 with the goal to create a cozy boardgamecafé in Aalborg city where customers can go to play games, meet new people and have a drink. Their plan is to offer a wide range of board games suited for different number of players. They are opened for suggestions and ideas that might improve the customer experience. Comparing the two businesses it can be observed that they have a similar location in the city, both of them are located in the center, only a couple of blocks from each other. The main difference between the two of them are prices, the variety of board games and the variety of snacks and drinks they offer.

The table in Fig. 3 displays and compares the difference in prices in both cafes. Amongst the most significant and noticeable is the difference in the price of the annual card which is almost two times cheaper in *Dice n' Drinks.* However, *Aalborg Brætspilscafe* offers discounts when it comes to students as well as offers a semiannual card. Aalborg is known for its university and high student life, having lots of both foreign and native students, so having a student discount seems to be a good to have policy. Besides that, both cafes offer food and drinks, where *Dice n' Drinks* has higher diversity. As it can be seen on Fig. 3, there is snacks, food, hot drinks etc., the difference in the price range is quite big and almost everything at *Dice n' Drinks* is more expensive.

	Aalborg Brætspilscafé	Dice n'Drinks
Annual card	450 kr.	200 kr.
Annual card for students	350 kr.	--
Semi-annual card	300 kr.	--
Semi-annual card for students	250 kr.	--
Daily card	30 kr.	20 kr.
Snacks	10 kr.	10 - 15 kr.
Food	--	15 - 45 kr.
Hot drinks	10 kr.	20 - 30 kr.
Soda and water	5 - 10 kr.	15 - 25 kr.
Alkoholic baverages	15 - 20 kr.	20 - 100 kr.

Fig. 3 Table of prices in both cafes [5] [6]

So far there was an 'aggressive' act between the companies, specifically, *Dice 'n Drinks* called the police to inform them that *Aalborg Brætspilscafe* is serving alcoholic drinks to the customers without authorization. The police verified that *Aalborg Brætspilscafes* employees don't actually serve the drinks, they just sell them to the customers, literally speaking they don't pour the drinks in the glasses. And there is no need of an authorization for that.

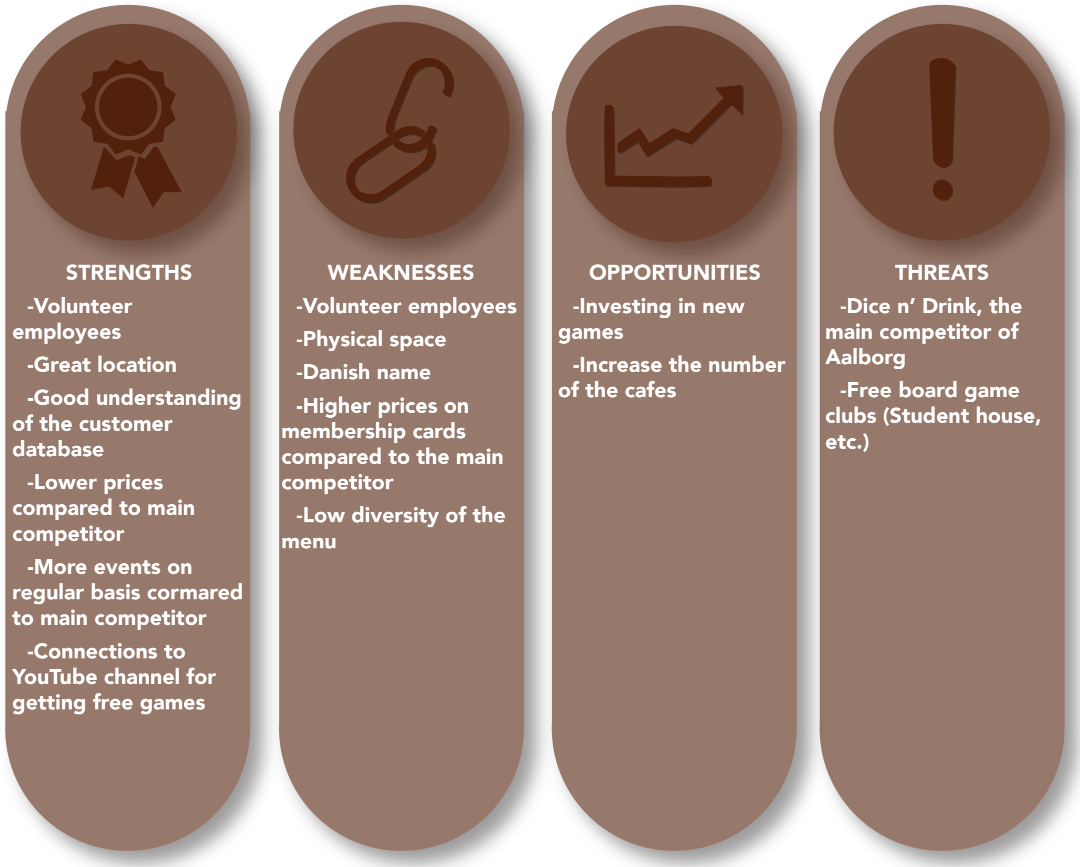
Threats of Substitution: In this part, the opportunities of the clients are reviewed, in particular whether it is easy for the customer to do what the business is doing but in a different way. The chances of substitution vary, depending on some certain factors. For instance, the treats of substitution are high when the substitute products are cheaper, have better quality, better performance or lower costs. On the other hand, the threats of substitution are lower when what was mentioned in the previous sentence is reversed or there are no substitute products. In the case of this project, it can be assumed that the threats of substitution are rather lower than higher. Looking at the boardgaming market,[9] board games prices vary but often their prices go over 400-500 kroner per single game, some even reach 1500 kroner or more. These prices are rather high and the chance to pay 30 kroner per day or 450 kroner annually for a variety of 150+ boardgames seems like a better option, thus lowering the threats of substitution amongst the clients. However, there is the Aalborg Library which has a lot of boardgames available for borrowing for free. This can be considered as a disadvantage for the cafe because the library creates alternate opportunities for clients, especially students.

Threats of New Entries: It is very important to know how easy and cheap it is for other companies to enter the market. Extensive entries of new business which are potentially competitive can definitely threaten the company's economical position. This is why it is very important to acknowledge the importance of increasing the awareness of new entries, understanding what barriers are there that might stop or reduce the amount of new competitors and preserve a favourable position over the potential competition. There are several known barriers for opening a new business. Some of those are barriers created by the government, creating patents, blocking distribution channels and more. The boardgaming business in Aalborg seems to have a high level threat of new entries because it is not restricted by anything aside capital and location. There are various channels of distribution, those being book stores, websites and others, which opens for the possibility of opening a new boardgame cafe.

In conclusion, both companies are at their early stages of development and are in a young market practiced by a few in the city. Fighting for customers, the goal for them is to expand and maintain a profit by improving the marketing approach, the range of board games and the variety of snacks and drinks, all of them leading to a better customer experience. It can also be concluded that both cafes are very similar in many occasions such as location, prices and amount of boardgames which extends their competitiveness even further.

1.5 SWOT analysis

This chapter will cover the SWOT analysis of the company. It is a well known method widely used to help understanding the business's strengths and weaknesses, as well as consider the potential opportunities and threats. Strengths and weaknesses are the factors which are controlled within the company. Opportunities and threats are external factors in the market, which could influence the company's economical state. As following, these four factors and relations between them will be analysed in this chapter and they can be seen in Fig. 4. The content of the figure is based on the customer interview (see sec. Interview), the organizational structure (see sec. Organizational Structure) and competitive analysis (see sec. Competitive Analysis).



STRENGTHS	WEAKNESSES	OPPORTUNITIES	THREATS
-Volunteer employees	-Volunteer employees	-Investing in new games	-Dice n' Drink, the main competitor of Aalborg
-Great location	-Physical space	-Increase the number of the cafes	-Free board game clubs (Student house, etc.)
-Good understanding of the customer database	-Danish name		
-Lower prices compared to main competitor	-Higher prices on membership cards compared to the main competitor		
-More events on regular basis cormared to main competitor	-Low diversity of the menu		
-Connections to YouTube channel for getting free games			

Fig. 4 SWOT analysis

Some of the biggest strengths the company has are having dedicated staff, great understanding of the customer base and wide supply. Accordingly, these bring weaknesses like insecure volunteer labor force and lack of space for the gamers. However, some of these vulnerabilities can be fixed or at least stabilized. Volunteers can be treated well so they would be stimulated to stay and the spaces can be used optimally under effective and creative supervision. Another strength is having lower prices on food and drinks compared to *Dice n' Drinks,* which is especially a factor when most customers are students. Speaking about the prices, the membership is more expensive than competition's, then the only solution for that would be to give equivalent supply, so it would be worth paying. *Aalborg Brætspilscafé* has connections to a very popular gaming channel in YouTube called *Geeks and Sundry* (see Appendix 1), which in exchange of advertising provides the cafe with free board games. Another strength that can be drawn is having regular events, which can draw lots of clients with the condition to keep them in longer term.

The biggest weakness, which was taken into consideration, is the name of the cafe. The cafe has a Danish name, which clearly states the concept of the business. But taking into consideration the number of international students, who could be potential clients, but do not know the language, a Danish name is clearly not a positive factor, for this segment. The physical space of the cafe has limitations and this can cause problems. Board games are often played by more than two people, and when it comes to big events it might lead to a situation where some people cannot play because there is no more space at the table or at the cafe.

The most considerable opportunity the company has currently is to extend the chain of cafes, although the owner stated he didn't plan extension in the next two years (see interview transcript). It was not possible for the project group to find similar businesses in surrounding cities like, Randers, Hobro etc. Even though Aarhus and Copenhagen seem to have multiple as well, it can be concluded that there are room for branching out, even though all the big university cities got similar cafes and clubs. Competition can be dangerous, but it also motivates the company to establish new strategies to gather even more clients. Having a competitor is challenging, forcing the company to always look for improvements, but making a bad step can have severe consequences when it comes to the number of customers. Some threats, like competitors, might be seen as collaborators as well. Places like Student House (Read changed to Nordkraft after the interview) occasionally has board game events, but this is not what they are purely volunteering, and are non-profit, so *Aalborg Brætspilscafe* could supply games in exchange for advertising the cafe during the event. One or another way, all collaborations should give advantages. The more partners the company has, the wider it is known. It is just a matter of choosing the right collaborators, which would help to reach the correct audience.

1.6 Mission and vision

When running a company a clear vision and mission is important. It is created to help all employers pull in the same direction, but having one at the beginning of a business, such as the *Aalborg Brætspilscafe* can be helpful to further developments.

Having these things in mind, and based on the interview with the owner of the cafe, a mission has been formulated. This is a general short statement, that explains the mission of the company. The mission statement was formulated by the project group, and not discussed in further detail with the owner of the cafe.

Mission

Aalborg Brætspilscafé's purpose is to provide a cozy board gaming experience for the customers, no matter the age, where they can also enjoy some drinks and snacks along with their friends.

Based on the SWOT analysis earlier in the report and the interview with the owner, a Vision has been formulated as well, this vision can be seen here:

Vision

In the next two years the company will focus on extending the boardgame count, extending the menu to offer a wider variety of drinks and snacks and also investing in marketing by creating more events. The overall vision would be to offer better services than their competitor and to keep having the number of customers increased by the month.

1.7 Strategic goals

The strategic goals of the company, reflected from the mission, vision and SWOT analysis, are to take advantage of their opportunities, strengths and find solutions for their weaknesses and threats. To be more specific, table in Fig. 5 presents every goal and the strategies chosen to achieve them. The strategies in this section are chosen by the project group, and is not necessarily what the cafe owner would do, these choices are based on the SWOT analysis earlier in the report and the mission and vision, which are also created solely by the project group.

Goal	Strategies
Expand the circle of customers	• Collaboration with other brands • Advertising during events • Active advertising in Facebook,both in English and Danish • Increase the number of events and contests • Offer discounts on membership
Extend the supplier chain	• Collaborate with possible board game suppliers (similar to Geeks and Sundry)
New cafes	• Look for opportunities to open new cafes in other cities. • Pay visits to other board-game cafes in other cities.

Diversity in menu (more snacks & drinks)	• Add diverse products for vast customer choice • Add food to the menu • Alcohol license.
Gain market share over the competitor	• Make a SWOT analysis of the competitor

Fig. 5. Table of strategic goals

Expanding the circle of customers would be one of the main objectives. There are many techniques that can be used to achieve that. Such are, active advertising, both in social media and through events and tournaments, steady membership discounts. As it was realized in the development of the SWOT analysis the current name and advertisements does not hit a certain segment of the Aalborg populace, namely foreigners, for example international students or people who are not fluent in Danish. There are many international students in Aalborg and losing this segment could be vital in the competition against the more English friendly, *Dice & Drinks*. Furthermore, the cafe can always expand in terms of assets. This can be done by finding reliable suppliers, who would supply the cafe with even more boardgames. These collaborations are often beneficial for both sides as the cafe can get free merchandise and supplier gets ad space.

Even though the cafe owner stated that he did not plan on expanding in the next two years(see sec. Interview Q), there is always a possibility of opening businesses in other cities. This however can be risky, as many factors should be taken into consideration such as location, competition in the area, budget, customer base, etc.

Another goal that was thought of was increasing the list of items in the kiosk menu. Looking back at the competitive analysis (see sec. Competitor analysis), it can be observed that *Aalborg Brætspilscafe* lacks diversity compared to their main competitor in Aalborg. The first obstacle to gain more diversity would be the alcohol license, *Dice & Drinks* got tab beer and beer from tab generally fits the student segment. Finally, getting further insight into what the competition does differently, can help the cafe improve and attract more customers. This can be done by creating a SWOT analysis from the competitions point of view, analysing it and then brainstorming countermeasures based on this new SWOT analysis.

1.8 Security plan

In order to measure the security of the entire system, it was decided to create a security plan to cover up the main issues and point out possible solutions. This is a really important subject, because any harm done to the system can be crucial and damage the entire company. It is vital to protect the system in order to save the customer database and the reservation system from being sabotaged by competitors, because any simple competitor can freely book all the tables, thus immobilizing the reservation system. A way to prevent random visitors from accessing the system. Backing up all the reservations is something to be considered, so that no data would be lost if any accident should happen.

One way to protect the system would be by adding a captcha on the website and doing IP checking in order to avoid any spam with reservations. It is important to modify the system, so that in the cafe's local computer, it would logout a user automatically after three minute when not used. Every time a user would book a table, one would have to fill in a captcha and then the system would check the user's IP. If the system would detect the same IP being used multiple times, it would give out a warning to the owner of the system.

The administrator of the website should be the developer. Nobody else should have any right to modify anything in the website, this is due to the possibility of someone else sabotaging the website. The administrators of the cafe's application should only be the owner in order to avoid anyone else in the cafe having full access to the system. It is possible for a random customer to just approach the computer that is left alone without care making the entire system completely vulnerable.

It is a good idea to have meetings every month and go through the entire system and check for any possible leaks. After some time without leaks it would be advisable to change this to once every half year.

1.9 Business case

Introduction and background

Aalborg Brætspilscafe needs a system to handle table and event reservations securely and efficiently. Currently they are using their phone and a website, but all reservations are written into a paper calendar instead of a database. What is suggested in this project, is a system that handles both reservations over the phone and in person with the help of an on site system, and reservations on the website. The idea is that no matter the channel for the reservations, the reservation is handled in the same database, thereby collecting all necessary info in a backupable and secure location, much better than any paper calendar.

Management summary

Holding all reservations in paper is ineffective, an electronic system would be able to announce any upcoming reservations. Thus avoiding employees wasting time in double checking the calendar. Also lowering the human error factor in the reservation systems, given that the computer system would add it as soon as it is written once, and it would not be necessary to read the automated emails the current websites sents employees when a new reservation is added. Thus avoiding the possibility of a mistake happening when the employee copy/pages the reservation into the calendar.

Different options:

1.Stay with the current method.

Cons:

a. Takes a lot of employee time.

b. If the employee at work can't access the email system all newer reservations goes without notice.

c. Manager has to double check if all reservations are added to the paper format, especially after a day off.

Pros:

a. Only good thing, this method is easy to learn and easy to use, the business is using volunteers and have a high possible exchange in employees because of this.

2. Use two integrated systems (A web page and an onsite application - connected to the same database)

Cons:

a. Takes some time to get into. Employees might need longer training.

b. More expensive than current method.

Pros:

a. No double checking, everything is held in one database and can easily be itemize by the system and showed in the onsite application.

b. Able to add several functions to ease every tasks. For example, a reminder for the employee right before a new reservation should be ready.

3. Use a web page (All reservations is made on the website, even those received by phone)

Pros:

a. Only needs one system, and if correctly developed the webpage can be very intuitive, thereby diminishing the amount of training employees need.

b. No double checking, everything would be on the database and should be accessible by all employees.

Cons:

a. Does not fit with the semester's goals.

b. More expensive than current method.

Recommendation:

The group recommends the two system method, some functions are easier implemented on an application than on a webpage. This was chosen because the amount of work going into developing a web page that handles everything is about the same as developing a web page that does 20% of the tasks and then a windows application that does the last 80%.

Cost / Benefit analysis

Cost:

As it can be seen on the cost/benefit model (Fig. 6), the costs of implementing the suggested system is based in the extra training of employees and the development of the system. System will be developed by students, thereby diminishing the cost developing cost. The employee training has no tangible price, because all employees are volunteers, so it can really only be measured hourly, and a quick introduction for about 30 minutes should be enough for most volunteers to learn the basics.

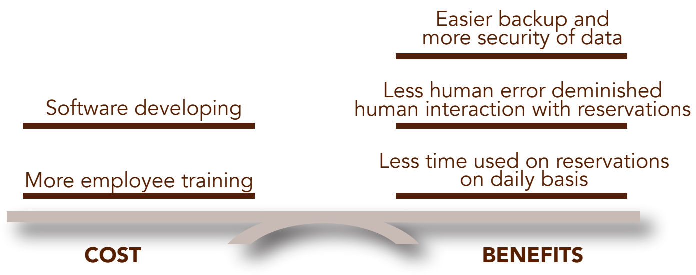

Easier backup and more security of data
Software developing
Less human error deminished human interaction with reservations
More employee training
Less time used on reservations on daily basis
COST
BENEFITS

Fig. 6 Cost/benefit model

Benefits:

The benefits of implementing the system is a bit less tangible than the cost. Less time used on reservations on a daily basis, is caused by several actions. Firstly, when a customer reserves a table on the web page the employee don't need to do anything, the reservation will be added to the database automatically and the employee will be notified if anything needs to be done shortly before the reservation starts. Secondly, there is no need to keep checking the calendar, whether it is updated nor if there is new reservations, the system will notify the employee if anything needs to be done.

Less human error [10], caused by less human interaction. This is added because the employees doesn't need to copy/paste from an automatically created email, thereby no risk of errors during this state in the reservation process. Everything is handled by the system.

As mentioned many times, everything is in a paper calendar, and though a paper calendar is pretty secure against cyber attacks and the like, it is not very secure against accidents. The cafe is often making tea, coffee, opening sodas and the like, while the calendar is on the table. It is really just a matter of time before something hits the pages of the calendar and the entire database is ruined. This won't happen with a digital database, and if one database is lost the backup will be in place.

2.SYSTEM DESIGN

The following chapter of the report will explain all steps of designing the system. It will cover the development of the system through different methods, starting with the very first vision of the system in a flow diagram and mock ups, the progress made during analysis and realisation of the requirements, and concluding it with a class diagram and relational model.

2.1 Flow diagram

A flow diagram was depicted to represent the initial idea of the system. It will help to define and pick the main use cases. It is also a starting point of the system design, which will be developed throughout this whole designing process. The initial idea is that the customer won't be able to interact with the system and this can be done only by the people who work at the cafe and the owner. Firstly, customer needs to make a call which is picked up by a volunteer, who has to login the system. Then the volunteer adds search information, and if there is no match the details has to changed or the call has to end. If the user is pleased, the volunteer proceeds to picking a table and adding a customer to the booking. Finally, the volunteer adds everything to a new booking and the call ends.

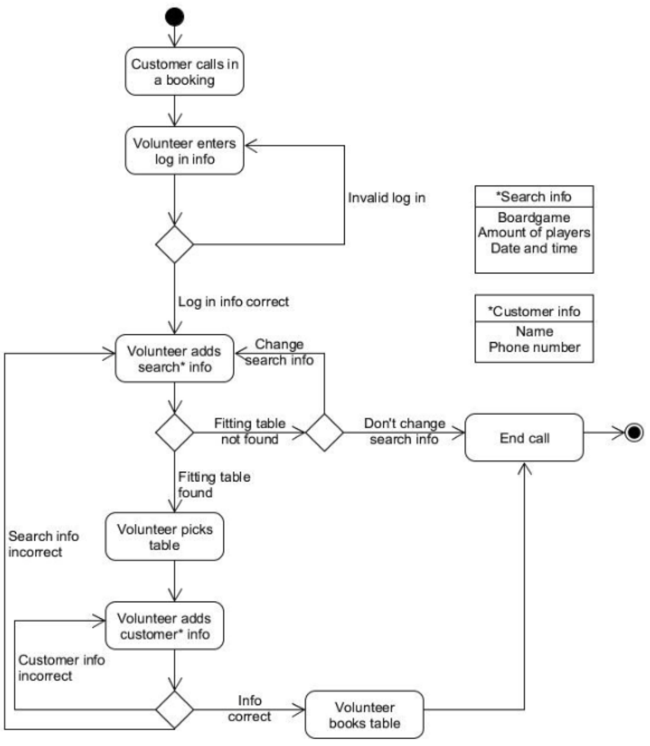


Fig. 7 Flow diagram

2.2 MockUps

When creating a system it is important to make certain that the developer and the customer are on the same page. To do this mockups are designed, a quick small description was made by the customer and an initial mockup of the system GUI is made based on these descriptions and wishes. Later on, when the mockups are done they can give a visual representation of the desired system. This way the customer can easily understand and give valid feedback before an expensive and time consuming development process is done based on misinformation. Furthermore these mockups in detail helps the developers through the entire design process.

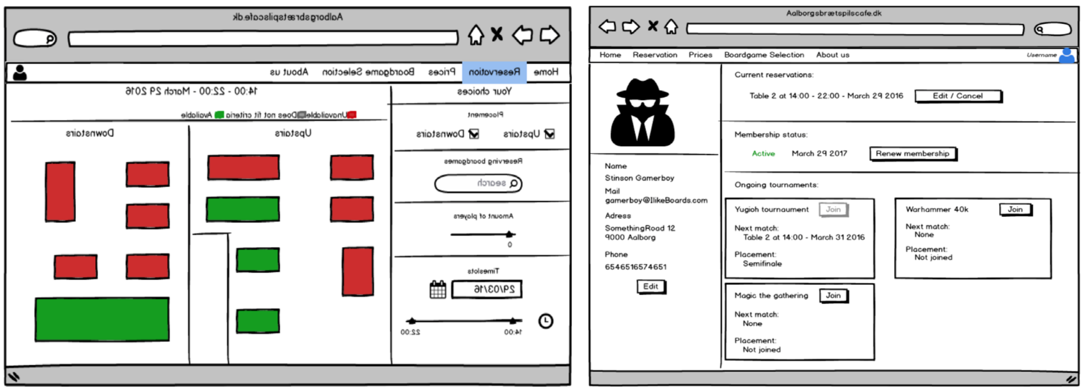


Fig. 8 Two mockup examples of the book table and profile pages

The figure above displays an example of two mockups designed for showing how booking tables should be handled and what the user profile page should consist of. The book table page is divided into two different areas, one showing the available tables and one showing the search inputs. The system will then color the tables according to availability, red is unavailable, as in already taken, green is available and fits criteria for search and lastly grey is for tables that are not taken, but does not fit the criteria of the search for one reason or the other. In the profile page, the user can view and edit it's own details, see or edit current reservations, create or renew membership and join available events. The pages are designed so they are user-friendly and easy to navigate while containing all the necessary information the user needs. All mock-up pages can be found in the Appendix 2.

2.3 Use case

Based on the flow diagram, the mockups and the interview in the previous sections, some use cases have been formulated and will be explained in the following section.

As it can be seen in the use case diagram (Fig. 9) there is a lot of different use cases in the system, most of which are executed by the customer-actor. However, the customer is extended by the game master and the volunteer, who is extended by the owner. It is important to make clear that the diagram shows that all other actors can act as a customer and execute all the use cases. However, while the customer can only change this info in his own profile, the volunteer and owner should be able to change it for all users, mostly because it should be possible for a customer to call in about bookings and the likes as well as doing it on his own. The game master role differs from the volunteer by only one task which is managing events. This choice was done in order to prevent confusion amongst the volunteers, so the person who creates an event would be the only administrator for that event. There is also the owner actor who besides being able to do all use cases, has three additional ones. He is the only one who can delete a user profile, can manage products and boardgames.

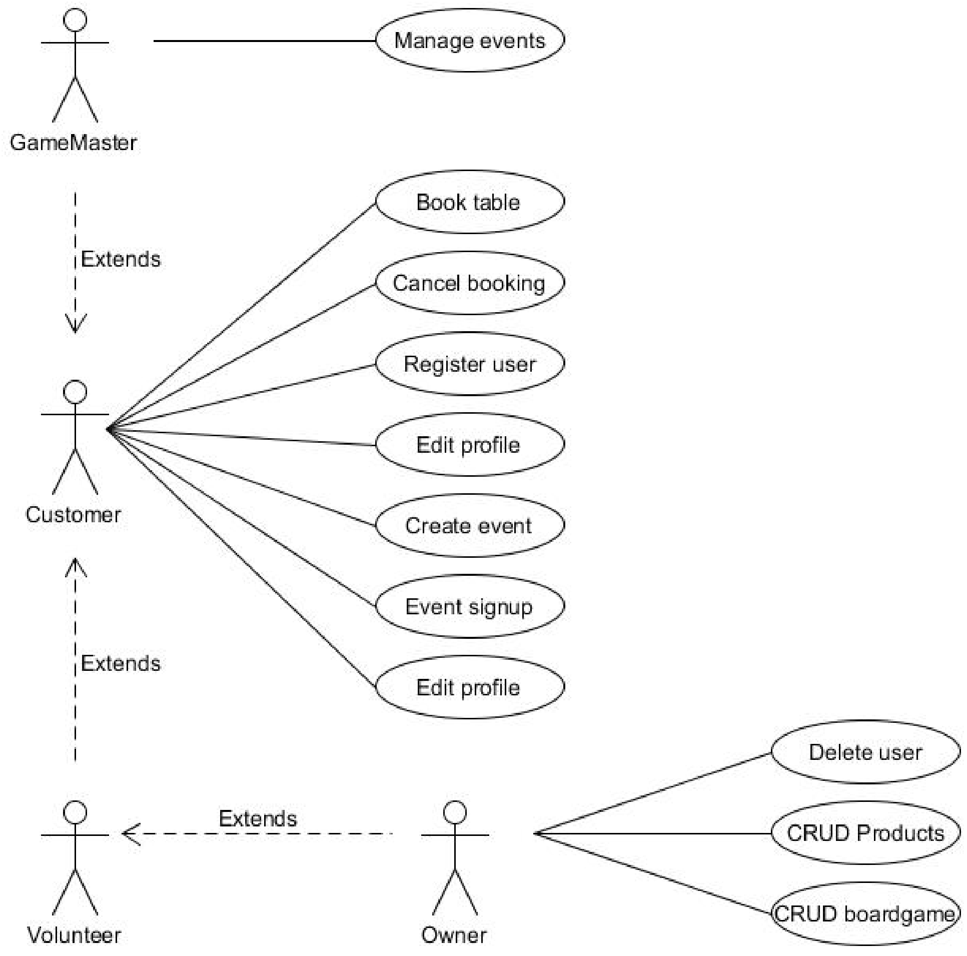


Fig. 9 Use case diagram

During the creation of this use case diagram, it was realised that there would be inheritance problems, namely with the gamemaster because it is necessary for everyone to be able to manage events. In the system this could be solved by adding a customer to an event, and then that customer is the game master, instead of having a game master by itself.

Event table

First thoughts regarding the system design were the establishment of the actors.

The group had a brainstorm about who is going to interact with the system, what are the possible scenarios and what is the purpose of them, which lead to realizing the difference between the actors and a better overall understanding of the system.

The following event table (TO BE) for 'Book table' was made based on the system workflow diagram to present an example of how the system was designed, "Book table" being one of the most important use cases in the system.

Event	Use case	Step in use case	Actor
Registered customer calls to book a table and a certain board game.	Book Table	*Log in *Find boardgame *Find table *Find customer *Book table *Logout	Volunteer Owner

Fig. 10 Event table

Brief use case description

Looking at the 'Book table' use case, a brief use case description was made as a summary of the main scenario. It helps to give a simple idea of what the use case is about and it makes a connection between the workflow diagram, event table and use case diagram.

Use case: Book Table
A registered customer calls to book a table and board game. The volunteer or the owner(if available) logs in the system and finds a board game, a table and the customer. They agree on the settings and the volunteer books the table. Then he logs out.

Fig. 11 Use case description

Fully dressed use cases

Based on the previous sections, the most important use cases have been picked to be developed in detail. These use cases are Book Table and Create Event. The reason to create the following fully dressed use cases are to make certain that the most important use cases are completely agreed upon, with every detail in place, between all the developers.

Fig. 12 describes all steps the volunteer has to undertake when a customer either calls for a reservation or wants to create a reservation in person at the shop with the help of a volunteer. In this use case scenario, the customer is a non-existing customer and the customer's information is manually added by the volunteer instead of the database, which would add the customer info if the volunteer was able to access an already created customer profile, as explained in alternative flow 9.a. The whole booking process is very simple and it can only be done by a volunteer or the owner/administrator. In order to book a table, the volunteer has to access the system with a volunteer account where it is fairly easy to navigate through the GUI(see sec. 2.2 Mock-ups). Furthermore, the volunteer has to find a table by selecting the attributes desired by the customer. As it can be seen in Fig(above) the system is going to show green tables when there is a match, and if not the use case will fail. Finally, the volunteer must always log out from the system for safety measures as mentioned in the security section of the business chapter.

Use case name	Book Table	
Actors	Volunteer	
Pre-conditions	Volunteer has to be registered and there is an available table	
Post-conditions	The volunteer request a booking	
Frequency	--	
Main Success Scenario (Flow of events)	Actor (Action)	System (Response)
	1. The volunteer logs in the system.	2. The system responds to the login request.
	3. The volunteer selects the reservations tab	4. The system transfers to the reservation tab GUI
	5. The volunteer selects all booking parameters in order (floor, amount of players ,tablet time and date)	6. The system includes the different parameters and updates GUI
	7. The volunteer selects a table to be reserved	8. The system updates the color of the selected table in the GUI
	9. The volunteer confirms the reservation with name and phone number	10. The system registers the reservation
	11. The volunteer logs out	12. The system shuts down the connection to the DB
Alternative flows	1.a Volunteer misspells login info. System pops up with an error "Username or password is incorrect" 5.a. If there are no results by the given choices, the system colors all tables red and the volunteer has to reselect the choices again until there are available tables in the system 9.a. The customer is already in the system and the volunteer searches for the right customer in the database and adds him to the booking. 10.a. The system can't register because table has been taken by another user in the meantime. The system makes an error popup and the volunteer can now pick another table.	

Fig. 12 Fully dressed use case for Book Table

Fig. 13 shows the Create Event use case, after discussing the different methods used for managing the game master in this use case, it was decided that the game master should be a customer added to the event. This means that the volunteer would add an event name, a costumer for the game master role, a description and what games are used for the event.

Use case name	Create Event	
Actors	Volunteer	
Pre-conditions	The volunteer must be registered in the system as a volunteer or administrator	
Post-conditions	The volunteer creates an event	
Frequency	--	
Main Success Scenario (Flow of events)	Actor (Action)	System (Response)
	1. The volunteer logs in the system	2. The system responds to the log in
	3. The volunteer selects "Create new event" from the home page.	4. The system responds to the request and redirects the user to the create event page
	5. The volunteer adds all event creation details(event name, GM, description and games)	6. The system displays the information in the GUI
	7. The volunteer selects create event after filling the spaces	8. The system displays the new data and awaits confirmation
	9. The volunteer confirms the event creation	10. The system registers the new data
	11. The volunteer logs out	12. The system closes the connection
Alternative flows	1.a. Volunteer misspells log in info. System pops up with an error "Username or password is incorrect" 9.a. Volunteer realizes some of the input was incorrect and goes back to update the information. 11.a. The volunteer already got a list of people who wishes to sign up, so he adds them before logging out.	

Fig. 13 Fully dressed use case for Create Event

2.4 Domain model

The domain model of the system includes nine classes. Earlier three classes were to be used to hold the different people, involved in the boardgame cafe, the customer, the volunteer and the owner. But because they all hold the same attributes and only have slightly differences in methods, they were combined in one class named Person

The board game cafe got two different experiences that the customer will be able to take use of, events and bookings. Booking is when the customer comes to the cafe and just needs a table once, these bookings can also take use of the board game class to add a specific board game for the booking, in case the customer wants to play a certain board game with his friends.

The event class is for events that might spread over a longer period of time. These can be campaigns in a certain game or 'pen and paper', or it could also be a tournament in a certain game or maybe a figurine game like Warhammer or Warmachines.

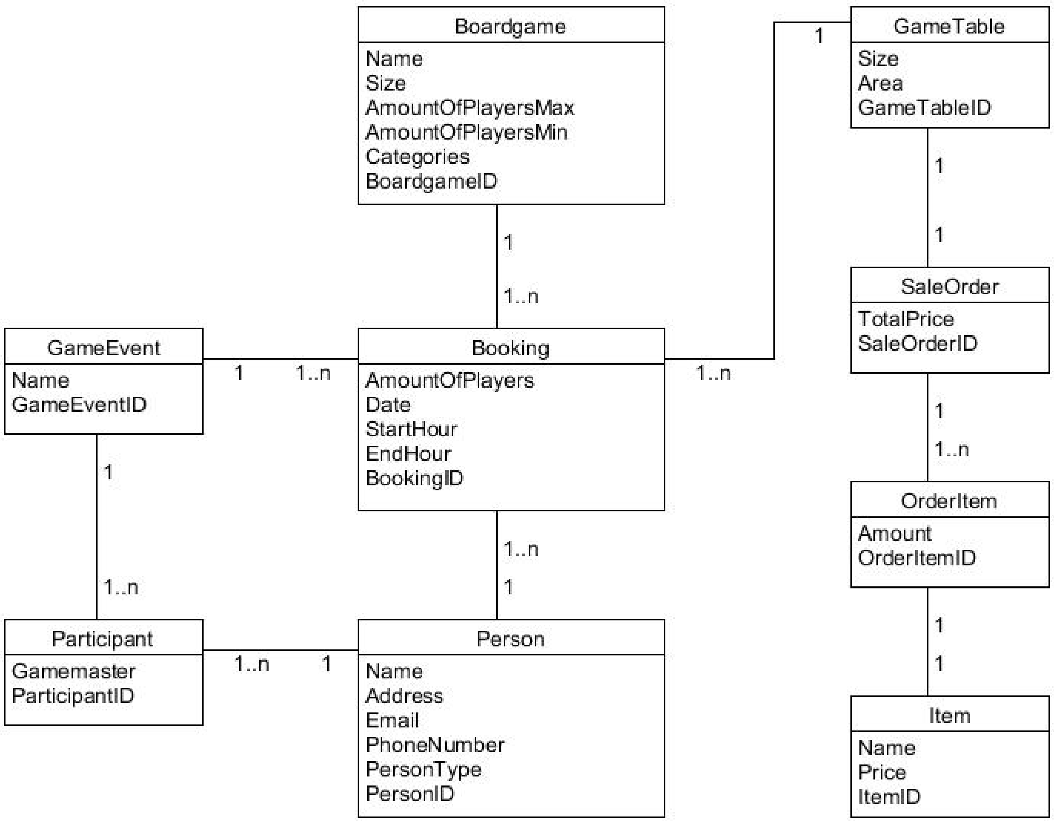


Fig. 14 Domain Model

Apart from the board game cafe there is a small kiosk which sells snacks and beverages. This shop is controlled by the Item, OrderItem and Order classes, and is then added to a certain table and can be settled at the end of the gaming session.

2.5 System sequence diagrams

Two system sequence diagrams for the book table and create event uses cases were created, in order to show how a particular scenario is executed, in what kind of order and what inter-system events it consists. Both SSDs illustrate the interactions between the volunteer and the system. The first diagram (Fig. 15) is for book table, where the volunteer searches boardgames, the inputs are Name, PlayerAmount, Size and Category. The system returns list of board games. Then the volunteer picks a boardgame and sends the boardgame object to the system, which returns the boardgame name. Picking a board game is done first so it can categorize which game table can be picked since it holds PlayerAmount. The next step is searching a game table by inputting Area, Date, Amount of hours and Amount of Players. The system should return list of game tables. Finding a customer comes last and to do that the volunteer has to input the customer's phone number, which returns a customer object. Finally, the volunteer has to add the new booking to the system which returns the booking object.

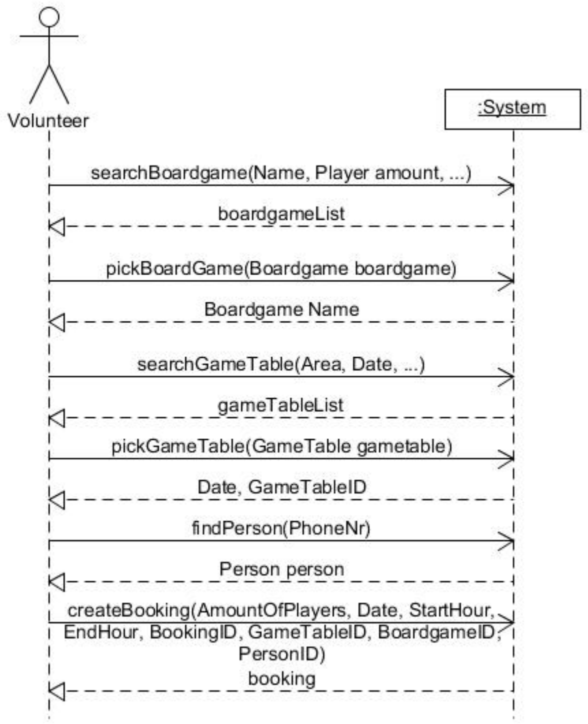

Volunteer
:System
searchBoardgame(Name, Player amount, ...)
boardgameList
pickBoardGame(Boardgame boardgame)
Boardgame Name
searchGameTable(Area, Date, ...)
gameTableList
pickGameTable(GameTable gametable)
Date, GameTableID
findPerson(PhoneNr)
Person person
createBooking(AmountOfPlayers, Date, StartHour, EndHour, BookingID, GameTableID, BoardgameID, PersonID)
booking

Fig. 15 SSD for BookTable

The second system sequence diagram (Fig. 16) shows the interaction in the create event use case. Firstly, the user creates an event object and the only input is event name, returning GameEventID. Then the user finds a person based on phone number which returns a person object. This is how the first participant is created by using the person object, which returns participant. Then the volunteer would search a board game by one or more of the four parameters Name, PlayerAmount, Size, Category which returns a list of board games. Following up is searching a game table by inputting area, amount of players and board game which returns list of game tables. Lastly, the user adds the booking to the event which returns the booking object.

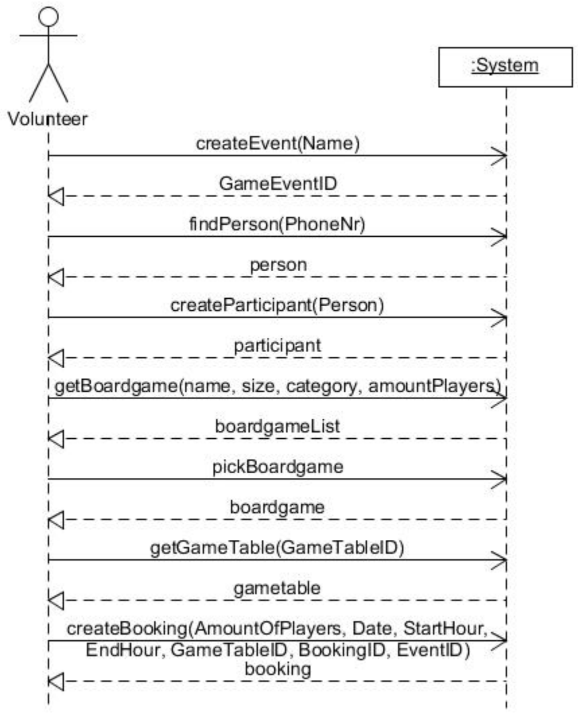

Volunteer
:System
createEvent(Name)
GameEventID
findPerson(PhoneNr)
person
createParticipant(Person)
participant
getBoardgame(name, size, category, amountPlayers)
boardgameList
pickBoardgame
boardgame
getGameTable(GameTableID)
gametable
createBooking(AmountOfPlayers, Date, StartHour, EndHour, GameTableID, BookingID, EventID)
booking

Fig. 16 SSD for CreateEvent

2.6 Operation contracts

This section is about operation contracts, two operation contracts will be shown in this section, one for each of the two main use cases. Operation contracts were created in order to represent the step-by-step functionality of a specific use case. These contracts shows the important parts of each case that occurs, such as preconditions, postconditions.

Use Case Book Table

In this example, Book Table case is used. The important preconditions of this case are that the customer, the board game and the table are in the database and that the table is available during the chosen period of time. In postconditions, at first, the booking object is created. Customer, table, boardgame are assigned to the booking. Date, startTime, endTime, amountOfPlayers are updated and applies to the booking object.

Operation: bookTable(Boardgame, amountOfPlayers, date, startTime, endTime, table, customer)
Use Case: Book Table
Preconditions: Customer exists. The table is available during the period between startTime and endTime on date. Board game is available during the period between startTime and endTime on date.
Postconditions: Booking object created, called booking Booking associates with customer Booking associates with table Booking associates with boardgame Attribute updated, date becomes booking.date Attribute updated, startTime becomes booking.startTime Attribute updated, endTime becomes booking.endTime Attribute updated, amountOfPlayers becomes booking.playerAmount

Fig. 17 Operation conract for Book Table

Use Case Create Event

In the second example, createEvent case is used. The important preconditions in this case are that the customer exists, tables and board games are available for the dates for the reservation. The postconditions for this case are event object is created.

Operation: createEvent(Boardgame, amountOfPlayers, Dates, table, customer)

Use Case: Create Event

Preconditions:
Customer exists.
The tables are available for the dates of reservation.
Board games are available for the dates of reservation.

Postconditions:
Event object created, called event
Event associates with participant
Event associates with person
Event associates with table
Event associates with boardgame
Attribute updated, date becomes event.date
Attribute updated, amountOfPlayers becomes event.playerAmount

Fig. 18 Operation conract for Create Event

2.7 GRASP (General Responsibility Assignment Software Patterns or Principles)

After meeting requirements and creating a domain model, reasonable methods to appropriate classes must be added. GRASP was the tool used to help to understand fundamental object design and be able to argue for the design in a comprehensible way. GRASP consist of nine patterns, which assign responsibilities to classes and objects. Responsibilities can either ‘do’ or ‘know’ something, for example, creating an object or knowing about a related object. Each pattern has a name, a problem and a solution, which when applied to a design can prevent errors in later stages of implementation. This tool was useful to find the best solutions for further system design and some of the applied patterns and examples will be presented in the following chapter.

Starting with Creator pattern, which is applied to Order and OrderItem connection. Since Order contains many OrderItems, according to the Creator pattern, Order gets a responsibility of creating OrderItems. Choosing Order as the creator also supports Low Coupling, which indicates lower maintenance dependencies and higher possibilities for reuse. [11]

Another pattern which was applied is the Information Expert. This can be seen in Order again, where the responsibility assigned to it was to know TotalPrice, adequately OrderItem has to know subtotal of an item, and lastly Item has to know the price of an item. So to fulfil responsibilities of TotalPrice, three responsibilities were assigned to three different design classes of objects. This responsibility leads to High Cohesion, where objects are focused so they do not perform any unrelated things.

One more example for Low Coupling in system architecture is where BookTableUI passes the data to BookTableCtr, which creates BookTable, and this way BookTableUI does not get the responsibility to create BookTable, which again leads to High Cohesion.

Further applied pattern was Indirection, where BookTable was responsible for knowing information about BoardGame, Table, Person. Those do not have any direct coupling between each other and because of this, reusability stays higher.

Applying patterns were the most reasonable tool while designing logical and efficient system. Even though some connections between classes and objects might seem obvious, it is always better to have a certain argument for the made decision.

2.8 Communication diagrams

To represent the interactions between objects and layers of system architecture, communication diagrams were used. The same diagrams indicated the flow of control and data. The mentioned interaction diagrams were made hand in hand with GRASP patterns in order to have an objective and logical system design, where it is clear how different layers communicate to each other. As the most important use case for this project was Book Table, its realisation through communication diagram will be explained further. Because of the size of these diagrams, only Book Table (Fig. 18) was chosen to be presented in the report, while Create Event can be seen in the appendix (see Appendix 8)

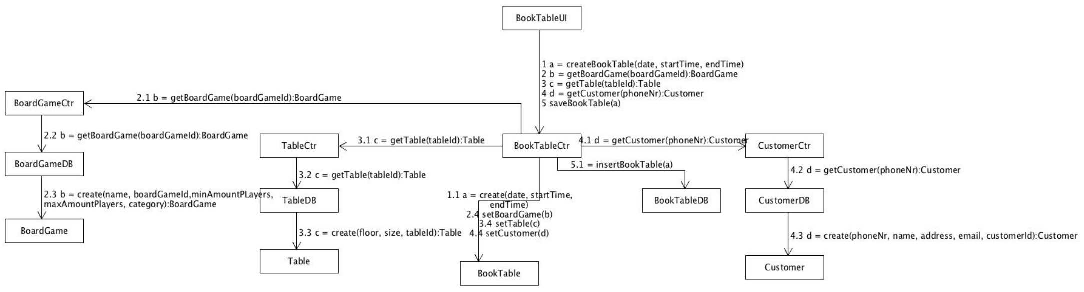

BookTableUI
1 a = createBookTable(date, startTime, endTime)
2 b = getBoardGame(boardGameId):BoardGame
3 c = getTable(tableId):Table
4 d = getCustomer(phoneNr):Customer
5 saveBookTable(a)
BoardGameCtr
2.1 b = getBoardGame(boardGameId):BoardGame
2.2 b = getBoardGame(boardGameId):BoardGame
BoardGameDB
2.3 b = create(name, boardGameId,minAmountPLayers, maxAmountPlayers, category):BoardGame
BoardGame
TableCtr
3.1 c = getTable(tableId):Table
3.2 c = getTable(tableId):Table
TableDB
3.3 c = create(floor, size, tableId):Table
Table
BookTableCtr
1.1 a = create(date, startTime, endTime)
2.4 setBoardGame(b)
3.4 setTable(c)
4.4 setCustomer(d)
BookTable
5.1 = insertBookTable(a)
BookTableDB
4.1 d = getCustomer(phoneNr):Customer
CustomerCtr
4.2 d = getCustomer(phoneNr):Customer
CustomerDB
4.3 d = create(phoneNr, name, address, email, customerId):Customer
Customer

Fig. 19 Communication Diagram for Book Table

Is it can be seen in Fig. 19, four different layers were designed. There is a model, database, control and UI layers. The model layer is connected to the database layer, which is connected to the control layer, which through BookTableCtr is connected to the UI layer. The controllers are using the database classes to retrieve data from the database, which in turn is send to the model layer for constructing objects. These objects will then be returned to the controllers for manipulating the required data. As the observant reader will notice the BookTable part of the communication diagram is set up quite differently than the other parts. This is because all the other controllers access the database to get all the data for the objects and then manipulates the objects, but the BookTableCtr does not retrieve any data to create the object, instead it receives all the data from the UI and the other controllers, before inserting the object into the database. The only data the database has to give to create a new booking is the bookingID, as it will just be the newest ID in the database plus one. Retrieving and calculating the new ID can be done at the time of insertion, rendering the database layer of the BookTable part completely useless until inserting the data. While inserting data, it will be checked up against current bookings and then, if it does not conflict with any of them, added.

2.9 Class diagram

The class diagram on Fig. 20 describes the system in detail. The project group created the class diagram in order to show the system's classes with their attributes, as well as showing different methods that occur in the classes. There are plenty of possible methods that haven't been added to the diagram, because they are not going to be implemented into this case yet. The diagram consists of numerous kinds of methods, for example: 'createBooking' which books a table, but in order to get all sorts of information, it needs to callout other methods in other classes, like 'searchGameTable' to find a suitable table for the booking. 'ConnectionDB' class is the core of the program, because it creates instances between the program and the database, in order to send information to it. The class is used almost everywhere in all times. Classes such as 'createGameEvent' have not been added due to lack of time to implement them, but will be considered being added in the near future.

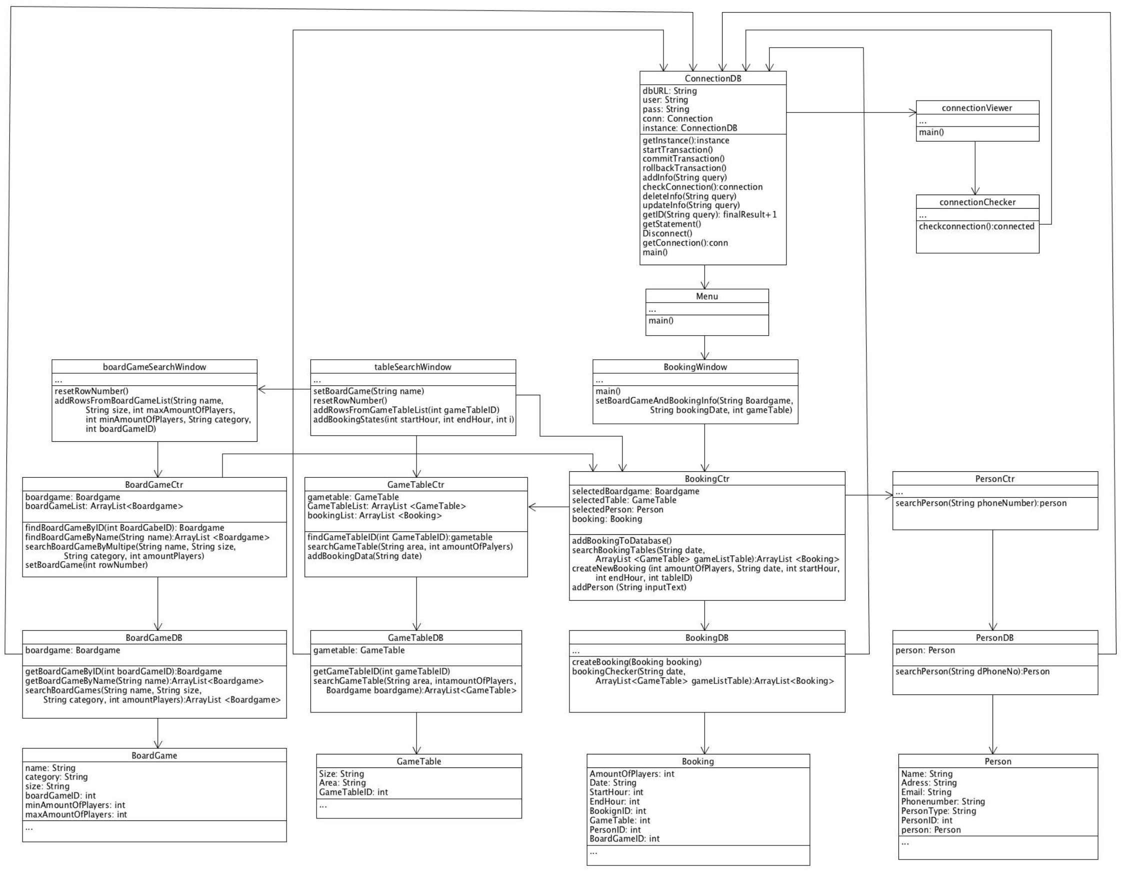

ConnectionDB
dbURL: String
user: String
pass: String
conn: Connection
instance: ConnectionDB
getInstance():instance
startTransaction()
commitTransaction()
rollbackTransaction()
addInfo(String query)
checkConnection():connection
deleteInfo(String query)
updateInfo(String query)
getID(String query): finalResult+1
getStatement()
Disconnect()
getConnection():conn
main()
connectionViewer
...
main()
connectionChecker
...
checkconnection():connected
Menu
...
main()
boardGameSearchWindow
...
resetRowNumber()
addRowsFromBoardGameList(String name, String size, int maxAmountOfPlayers, int minAmountOfPlayers, String category, int boardGameID)
tableSearchWindow
...
setBoardGame(String name)
resetRowNumber()
addRowsFromGameTableList(int gameTableID)
addBookingStates(int startHour, int endHour, int i)
BookingWindow
...
main()
setBoardGameAndBookingInfo(String Boardgame, String bookingDate, int gameTable)
BoardGameCtr
boardgame: Boardgame
boardGameList: ArrayList<Boardgame>
findBoardGameByID(int BoardGabeID): Boardgame
findBoardGameByName(String name):ArrayList <Boardgame>
searchBoardGameByMultipe(String name, String size, String category, int amountPlayers)
setBoardGame(int rowNumber)
GameTableCtr
gametable: GameTable
GameTableList: ArrayList <GameTable>
bookingList: ArrayList <Booking>
findGameTableID(int GameTableID):gametable
searchGameTable(String area, int amountOfPalyers)
addBookingData(String date)
BookingCtr
selectedBoardgame: Boardgame
selectedTable: GameTable
selectedPerson: Person
booking: Booking
addBookingToDatabase()
searchBookingTables(String date, ArrayList <GameTable> gameListTable):ArrayList <Booking>
createNewBooking (int amountOfPlayers, String date, int startHour, int endHour, int tableID)
addPerson (String inputText)
PersonCtr
...
searchPerson(String phoneNumber):person
BoardGameDB
boardgame: Boardgame
getBoardGameByID(int boardGameID):Boardgame
getBoardGameByName(String name):ArrayList<Boardgame>
searchBoardGames(String name, String size, String category, int amountPlayers):ArrayList <Boardgame>
GameTableDB
gametable: GameTable
getGameTableID(int gameTableID)
searchGameTable(String area, intamountOfPlayers, Boardgame boardgame):ArrayList<GameTable>
BookingDB
...
createBooking(Booking booking)
bookingChecker(String date, ArrayList<GameTable> gameListTable):ArrayList<Booking>
PersonDB
person: Person
searchPerson(String dPhoneNo):Person
BoardGame
name: String
category: String
size: String
boardGameID: int
minAmountOfPlayers: int
maxAmountOfPlayers: int
...
GameTable
Size: String
Area: String
GameTableID: int
...
Booking
AmountOfPlayers: int
Date: String
StartHour: int
EndHour: int
BookignID: int
GameTable: int
PersonID: int
BoardGameID: int
...
Person
Name: String
Adress: String
Email: String
Phonenumber: String
PersonType: String
PersonID: int
person: Person
...

Fig. 20 Class Diagram

There is quite a difference between the communication diagram and the class diagram. This is because of changes made during the programming phase of the project, giving quite a different structure to the whole system. Because of this it was decided to create a class diagram based on the finished program to give the reader a better overview of the code before reading through so many lines of code. Which would be made harder for even an experienced programmer if that person was trying to hold it up to the old class diagram.

One of the differences in the new class diagram, which can also be seen in the communication diagram is that the coupling is not as great as the design. There are too many classes connected to the ConnectionDB class, these method calls should at least have been done between classes in the domain model and not from so many different layers and classes. The ConnectionDB is also holding the main method and is the one that starts multiple threads to run the program concurrently. With one thread running the connectionViewer and one thread running the booking methods, starting with the Menu frame and then working on the rest of the program from there. All that said, a lot of the original design is still in place, the Boardgame part of the program is a good example of a more proper connection between the layers. The final program design will be explained further in the implementation chapter, before that the relational model will be shown and explained.

2.10 Relational model

The relational database presented in Fig.21 shows the references made through the primary and foreign keys between the relations. All tuples have a primary key as an attribute, which makes them unique, such as the GameEventID attribute in the GameEvent relation. Here the GameEventID is referenced as a foreign key in the Participant relation and also in the Booking relation for the purpose of having multiple participants and bookings in a single game event.

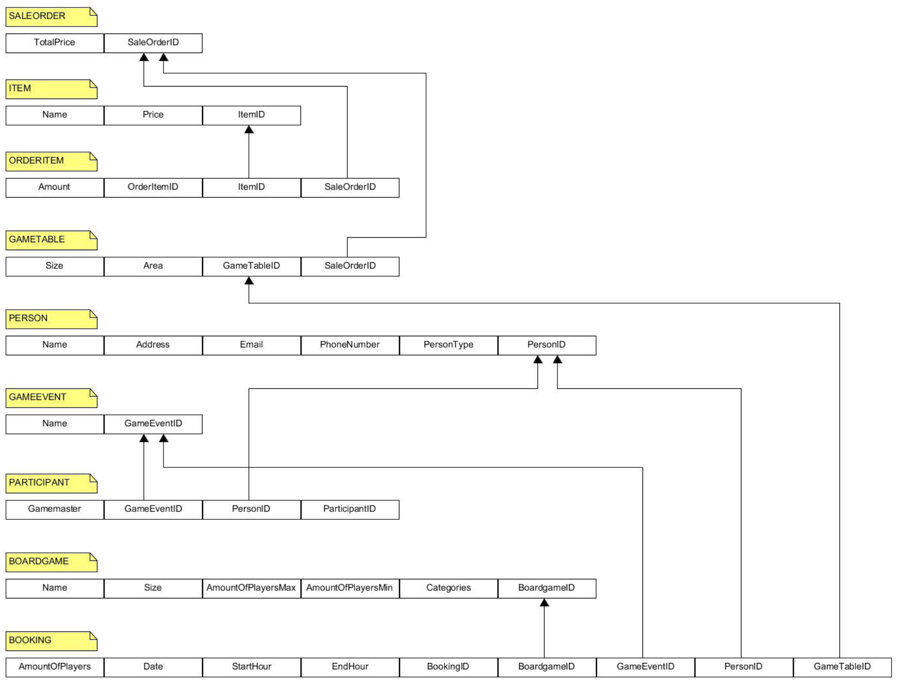


Fig. 21 Relational Model

The same applies for the rest of the relations, Booking being the relation with the most foreign keys, referencing to the Boardgame, GameEvent, Person and GameTable relations to create the overall booking process of the system.

A relational model is important to have before starting up the database operations, especially when the different tables are going to be as intertwined as this design suggests. After adding all the tables and adding multiple data to the system, changing a single table can have a lavine kind of effect on the rest of the database tables. Single table having a foreign key from a table that has to be deleted and readded will cause problems for that table, having to delete that table as well in order to be allowed to change the first table. Making the right choices about all the different attributes, which may be null and which may not, which attributes the different tables should have, are important aspects to go through thoroughly. Changing just one attribute to null from not null or the other way around or even just changing its data type will create the before mentioned lavine, given the developers a lot of extra work to set up the database again and of course backup all the data.

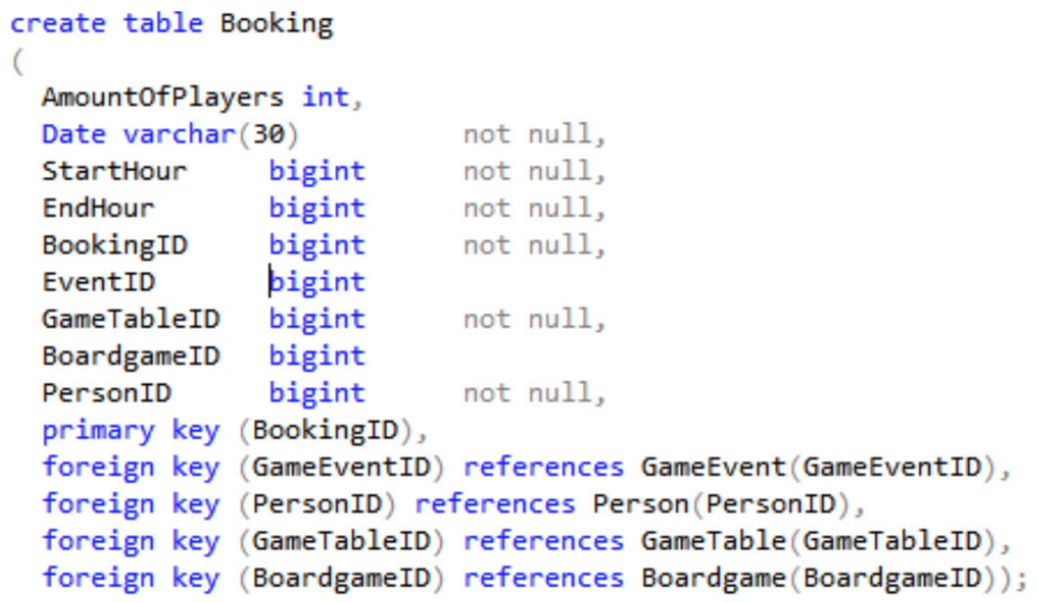
3.IMPLEMENTATION

Through the report a system has taken form. This chapter will explain the finished system, or at least the part chosen to be implemented, which is the booking system, with possibility of adding boardgames, userinfo and the rest of the desired booking info into the database. The chapter will not go through the entire code line by line, but take out a few pieces of it that has been found interesting during the development phase. The full code can be seen in Appendix 3.

Following the system development phase comes the implementation phase, based on the diagrams and starting with the SQL relations which help the system store, read, update and delete information through the Java methods.

The creation of the relations is easily built on the domain model. A relation is created for every class and the same attributes are used, the only thing added would be the data types and the referencing keys.

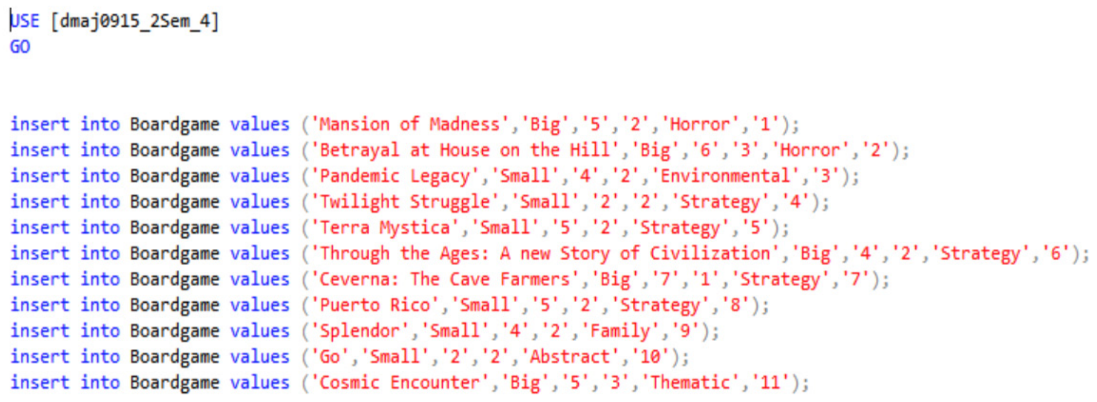
For example, the SQL script made for the Booking class in Fig. 22, has multiple attributes where most of them are foreign keys referring to other relations. The data types are bigint for the IDs and varchar for the Date attribute. EventID and BoardgameID have the possibility of being null as a booking can be made without necessarily participating at an event and also without selecting a board game, the customer might be undecided so the board game can be selected on the spot. The data types will be optimized in the future development phase, as using int and char instead of bigint and varchar would significantly reduce disk space and therefore improve the system's performance.



```
create table Booking
(
  AmountOfPlayers int,
  Date varchar(30)             not null,
  StartHour      bigint        not null,
  EndHour        bigint        not null,
  BookingID      bigint        not null,
  EventID        bigint
  GameTableID    bigint        not null,
  BoardgameID    bigint
  PersonID       bigint        not null,
  primary key (BookingID),
  foreign key (GameEventID) references GameEvent(GameEventID),
  foreign key (PersonID) references Person(PersonID),
  foreign key (GameTableID) references GameTable(GameTableID),
  foreign key (BoardgameID) references Boardgame(BoardgameID));
```

Fig. 22 SQL Script for Booking class

The insert SQL scripts simply insert information into the relations ,as tuples, in accordance with every tables' attributes and data types. Here, in Fig. 23, data is inserted into the Boardgame table and to be more specific, a number of 11 tuples have been inserted, which means that 11 different board games are available in the database. Each one of them having a distinctive name, size, maximum and minimum number of players, category and also an ID. This data will be accessed and used through the Java methods of the program.



```
USE [dmaj0915_2Sem_4]
GO


insert into Boardgame values ('Mansion of Madness','Big','5','2','Horror','1');
insert into Boardgame values ('Betrayal at House on the Hill','Big','6','3','Horror','2');
insert into Boardgame values ('Pandemic Legacy','Small','4','2','Environmental','3');
insert into Boardgame values ('Twilight Struggle','Small','2','2','Strategy','4');
insert into Boardgame values ('Terra Mystica','Small','5','2','Strategy','5');
insert into Boardgame values ('Through the Ages: A new Story of Civilization','Big','4','2','Strategy','6');
insert into Boardgame values ('Ceverna: The Cave Farmers','Big','7','1','Strategy','7');
insert into Boardgame values ('Puerto Rico','Small','5','2','Strategy','8');
insert into Boardgame values ('Splendor','Small','4','2','Family','9');
insert into Boardgame values ('Go','Small','2','2','Abstract','10');
insert into Boardgame values ('Cosmic Encounter','Big','5','3','Thematic','11');
```

Fig. 23 SQL Script for Booking class

The layers in the system

The system developed in this project has four layers, Model layer, Database layer, Control layer and a GUI layer. Focusing on the booking system ensures that a lot of the classes mentioned in the different design sections are removed from the full implementation. Therefore the model layer holds only four classes, Boardgame, Booking, GameTable and Person. Database layer holds six classes, BoardGameDB, BookingDB, connectionChecker, ConnectionDB, GameTableDB, PersonDB. Control layer holds the same amount as the model layer, four classes, BoardGameCtr, BookingCtr, GameTableCtr and PersonCtr. Lastly the GUI layer holds five classes, BoardGameSearchWindow, BookingWindow, connectionWiever, Menu, TableSearchWindow.

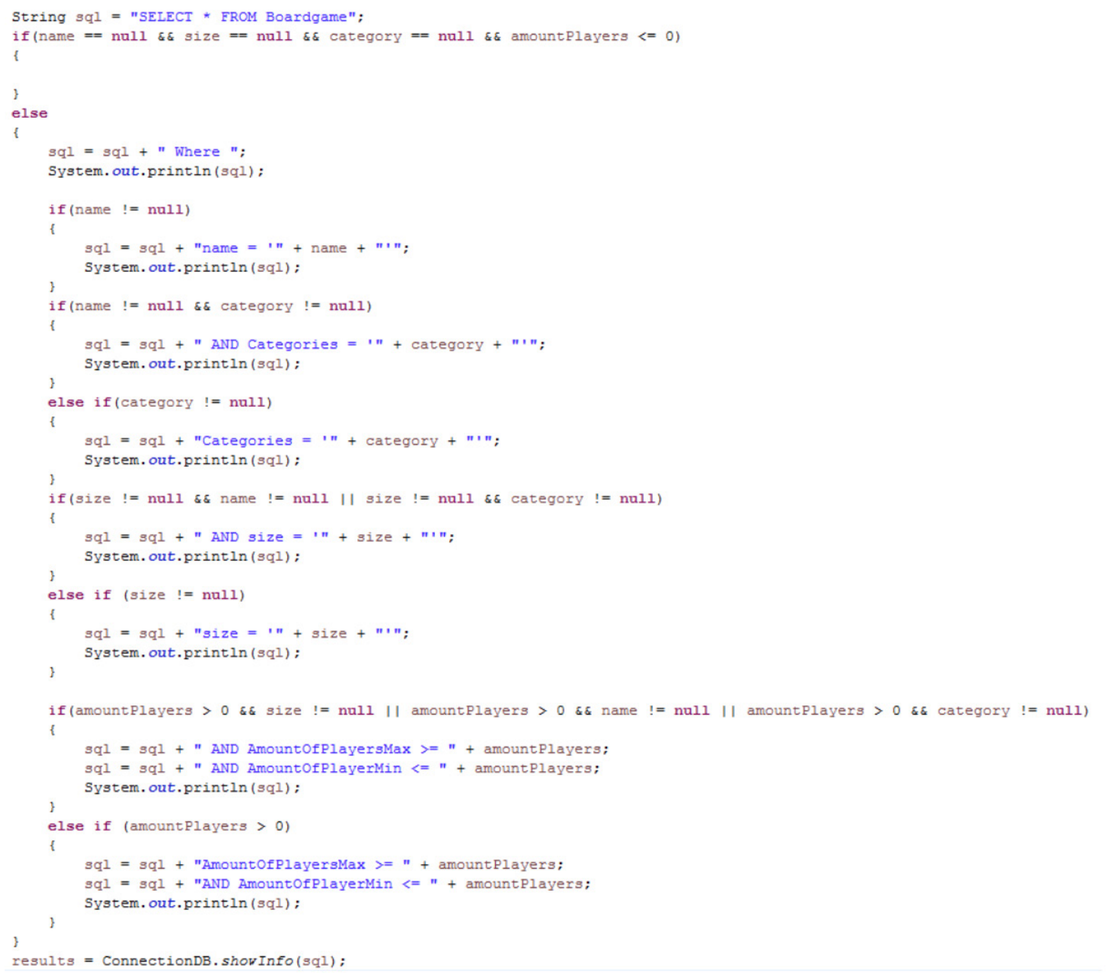
The classes of the model layer

The classes of the model layer can all be explained simply, they hold the attributes that are necessary to work with the different objects and are instantiated most often to hold data from the database or data that will be added to the database at a later time. These four classes hold constructors, some using multiple constructors for different parameters, and get and set methods for all attributes.

The use of multiple constructors is to use one constructor to get information from the database and another to construct the object to add info to the database. This is because of the primary key which is an int holding an ID for the object, so when getting info from the database the ID already exists and it can simply be retrieved from the database. When a new instance of the object is added to the database the ID is unknown, even to the user of the system and it needs to be calculated by the database layer, thereby getting the parameter from the insert method.

The classes of the database layer

The classes in the database layer are about sending and receiving data from and to the database. Some of them include search methods, both by a single attribute or by multiple. One of these search methods can be found in the BoardGameDB class, there is both a search method to find by name and ID. As well as one to find by multiple parameters, name, size, category and amount of players. One of the search methods can be seen in the following code snippet.



```java
String sql = "SELECT * FROM Boardgame";
if(name == null && size == null && category == null && amountPlayers <= 0)
{

}
else
{
    sql = sql + " Where ";
    System.out.println(sql);

    if(name != null)
    {
        sql = sql + "name = '" + name + "'";
        System.out.println(sql);
    }
    if(name != null && category != null)
    {
        sql = sql + " AND Categories = '" + category + "'";
        System.out.println(sql);
    }
    else if(category != null)
    {
        sql = sql + "Categories = '" + category + "'";
        System.out.println(sql);
    }
    if(size != null && name != null || size != null && category != null)
    {
        sql = sql + " AND size = '" + size + "'";
        System.out.println(sql);
    }
    else if (size != null)
    {
        sql = sql + "size = '" + size + "'";
        System.out.println(sql);
    }

    if(amountPlayers > 0 && size != null || amountPlayers > 0 && name != null || amountPlayers > 0 && category != null)
    {
        sql = sql + " AND AmountOfPlayersMax >= " + amountPlayers;
        sql = sql + " AND AmountOfPlayerMin <= " + amountPlayers;
        System.out.println(sql);
    }
    else if (amountPlayers > 0)
    {
        sql = sql + "AmountOfPlayersMax >= " + amountPlayers;
        sql = sql + "AND AmountOfPlayerMin <= " + amountPlayers;
        System.out.println(sql);
    }
}
results = ConnectionDB.showInfo(sql);
```

Fig. 24 Part of board game search method

As it can be seen, the snippet starts with an empty if statement, to make sure that there is data in the search fields. If no data has been added by the user, the search is ran by the ConnectionDB with the string "SELECT * FROM Boardgame" which returns all data in the boardgame table in the database. If some of the search fields are not empty the string is built, finding everything in the board game table with different values depending on the search data added by the user.

Another interesting class in the DB layer is the connectionDB, it holds the methods used for adding transactions and the method used for starting up the program in two different threads. One thread runs the connectionChecker to continuously check the database connection and another thread runs the booking system. This has been added to show the possibility of concurrency in programming.

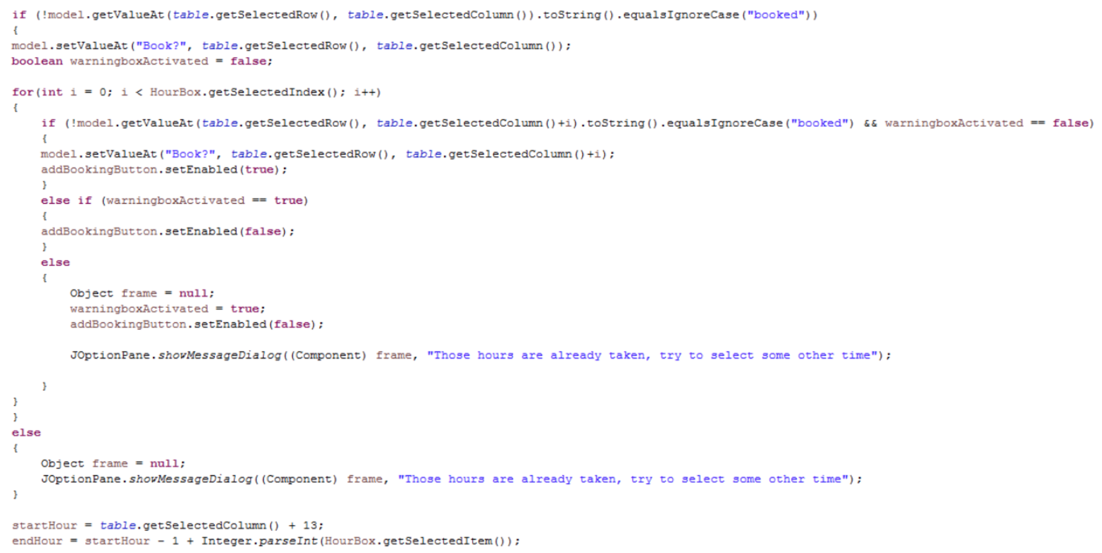
The transactions method which were mentioned earlier has been added to make sure that before adding a booking, everything has been checked and done, so the database doesn’t hold half the desired information or adds a conflicting booking to the table.

The classes of the control layer

The control layer in the system is used to handle the objects created, calling the model layer, and most of the classes in the control layer are filled with methods which is basically just a method call to a method call for an associated database class. The BookingCtr class holds four different objects, selectedBoardgame, selectedTable, selectedPerson and booking. These are used to construct a full booking that can then be added to the database when the user is done selecting all the attributes the user needs to save. For example, boardgameID is able to be null in a booking, so the user does not need to add a boardgame to the booking at all. But it does need a selected person and a selected table.

The classes of the GUI layer

Of all the GUI classes, the most interesting is the TableSearchWindow. This frame holds a table devised to show the different game tables availability, with a column for each opening hour and a row for each table. So when the game tables were searched, the Jtable in the frame is getting populated, with not only the tableID but also the hours the different game tables are booked for. On top of that, it was necessary to have a design, which allowed the user to pick the hours he wants a certain game table. The method should indicate the hours picked and show the user the chosen possibility, before the user continues with the booking. But a problem was found here because when the user changed the picked start hour, the indicator of the chosen hours stayed. This was handled by first emptying the table and then repopulating it every time the user changes the chosen start hour. After this was fixed another interesting method had to be added, when the user picked the start hour and the Jtable was populated to indicate the chosen hours, the system had to check if the table was already booked at those hours. A part of this method can be seen in the following snippet.



```
if (!model.getValueAt(table.getSelectedRow(), table.getSelectedColumn()).toString().equalsIgnoreCase("booked"))
{
model.setValueAt("Book?", table.getSelectedRow(), table.getSelectedColumn());
boolean warningboxActivated = false;

for(int i = 0; i < HourBox.getSelectedIndex(); i++)
{
    if (!model.getValueAt(table.getSelectedRow(), table.getSelectedColumn()+i).toString().equalsIgnoreCase("booked") && warningboxActivated == false)
    {
    model.setValueAt("Book?", table.getSelectedRow(), table.getSelectedColumn()+i);
    addBookingButton.setEnabled(true);
    }
    else if (warningboxActivated == true)
    {
    addBookingButton.setEnabled(false);
    }
    else
    {
        Object frame = null;
        warningboxActivated = true;
        addBookingButton.setEnabled(false);

        JOptionPane.showMessageDialog((Component) frame, "Those hours are already taken, try to select some other time");

    }
}
}
else
{
    Object frame = null;
    JOptionPane.showMessageDialog((Component) frame, "Those hours are already taken, try to select some other time");
}

startHour = table.getSelectedColumn() + 13;
endHour = startHour - 1 + Integer.parseInt(HourBox.getSelectedItem());
```

Fig. 25 Part of table population method

First, the system checks if the chosen hours have the text "booked" in the fields, if this is not the case, the chosen fields are populated with "Book?" as an indicator to which hours the user will book if he continues the booking with the current selection. This is done a number of times equal to the amount of hours the user has picked. If any of the hours picked are booked already, a warning message will pop up and warn that they are already being used.

Before the result is returned, 13 is added to the start hour because the first hour is in column one and that is hour 14:00. After the implementation of the code was finished, the next step would be testing the code. The next chapter will cover the testing process of the project, including a technical - and a usability test.

4. TESTING

4.1 Technical testing

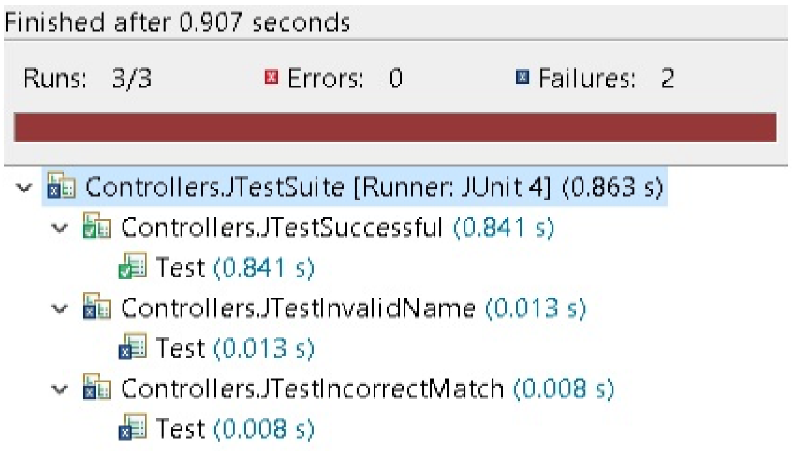
As mentioned earlier, this chapter will cover the two tests. The technical test comes first and it is a test of the code which looks into pre defined scenarios with certain parameter inputs and estimated outputs. If the output is equal to the expectations the test is deemed a success.

Test case ID	Scenario	Input				Expected result
		Name	Player amount	Game size	Cate-gory	
TC_1:	Scenario 1: Successful picking of a board game	V	V	V	V	JTable is populated consisting of data based on the input
TC_2:	Scenario 2: Incorrect match in board game name and amount of players	V	I	N/A	N/A	The JTable is not populated. The user gets a message: Insufficient amount of players is selected
TC_3:	Scenario 3: Board game is spelled wrong	I	N/A	N/A	N/A	The JTable is not populated. User gets a message: The board game not found

V: Valid; I: Incorrect, N/A: Input not necessary

Fig. 26 Table of scenarios for technical testing

Technical test was made in order to know exactly what will happen to the program if either scenario succeeds or fails, so then a specific result is expected. With this table (Fig. 26) it is faster to find errors when using the program, because then the tester can check the table if either one of the inputs were wrong. Fig. 27 showcases the technical part of testing the program. There are some different cases that were made to show different circumstances that happen if either one input is missing or not. For example, in Scenario 1, all of the inputs are valid and the follow up of that is that the JTable in the UI is populated by data that is gathered from the database. But in Scenario 2, where an input is missing or is incorrect, the JTable won't be populated and the user will get a warning, that there is an input missing. In scenario 3, if the board game's name is writ Table won't be populate

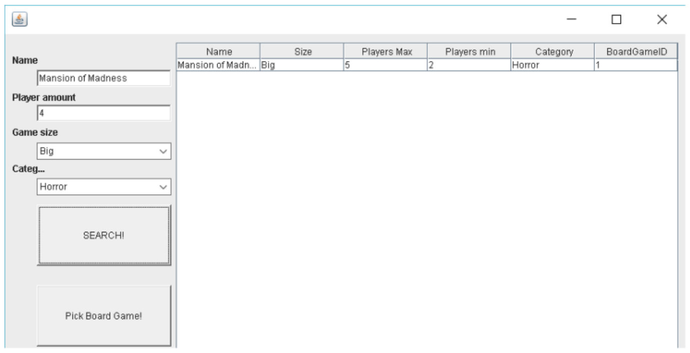


```
Finished after 0.907 seconds

Runs:  3/3        Errors:  0        Failures:  2

Controllers.JTestSuite [Runner: JUnit 4] (0.863 s)
    Controllers.JTestSuccessful (0.841 s)
        Test (0.841 s)
    Controllers.JTestInvalidName (0.013 s)
        Test (0.013 s)
    Controllers.JTestIncorrectMatch (0.008 s)
        Test (0.008 s)
```

Fig. 27 Excecution of JTestSuite Class



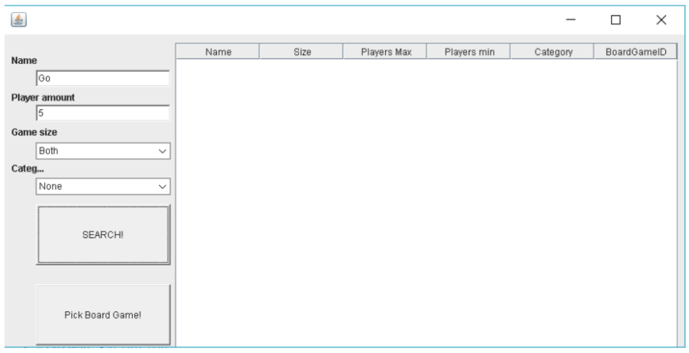
Name: Mansion of Madness
Player amount: 4
Game size: Big
Categ...: Horror

Name	Size	Players Max	Players min	Category	BoardGameID
Mansion of Madn...	Big	5	2	Horror	1

Fig. 28 Test case Scenario 1

Fig. 28 displays the scenario where all inputs are correct and then the JTable is populated with the results.

Fig. 29 displays Scenario 2, where one input is corrct and the other one is incorect. In this case the name and amount of players of the game are mismatched.

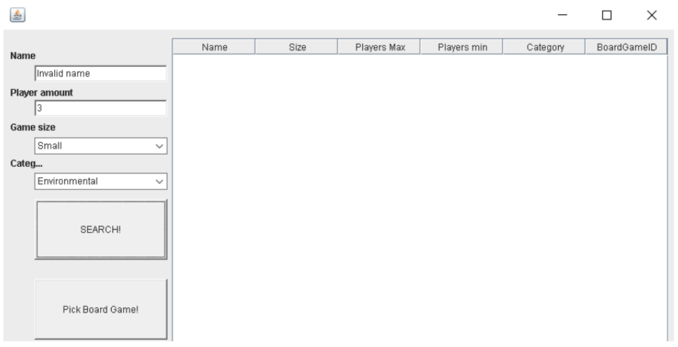


Name: Go
Player amount: 5
Game size: Both
Categ...: None

Name	Size	Players Max	Players min	Category	BoardGameID

Fig. 29 Test case Scenario 2

In Fig. 30, it showcases when either one of the inputs, for example, name is incorrect, then the search system does not find a boardgame and the JTable is not populated. This also works the same way if you have any other input wrong, then there are no results.


Name: Invalid name
Player amount: 3
Game size: Small
Categ...: Environmental

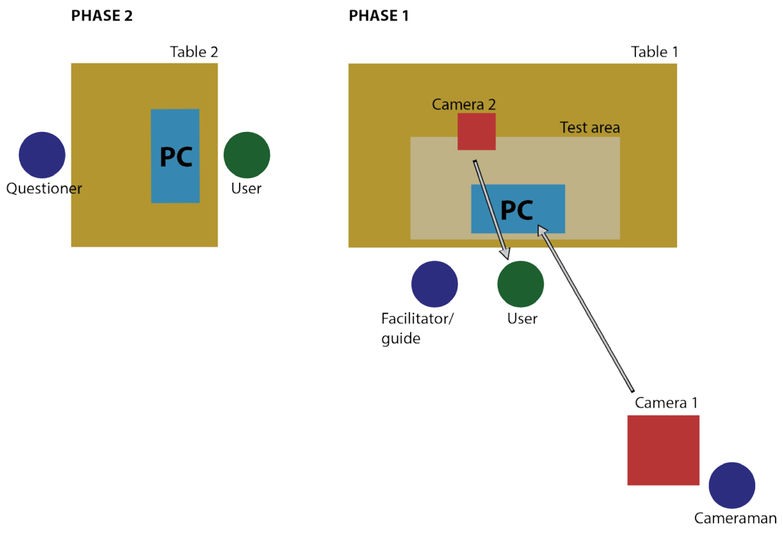
Name	Size	Players Max	Players min	Category	BoardGameID

4.2 Usability test design

In order to evaluate the product, a usability test was performed. The aim of this test was to observe how easy to navigate and understand the system was for the users, as well as to find out their thoughts about the concept and the GUI. With the given potential feedback, it was expected to acquire knowledge about the user's perspective, thus find out some positives and negatives of the program, and what is it that can be improved in the future.

Setup

The test was performed on eight UCN students, majority of them coming from Multimedia Design major, in the span of one day. It was set in a university classroom and conducted by three group members (Fig. 31). As it can be seen on the figure, the whole setup was predefined before the actual test began, and each member of the group was given a specific role. For the purpose of the test being not over crowded,



it was decided that only three people will conduct the test, while the other two group members were looking for participants. The experiment was divided into two phases. The first phase was where the actual test was performed. There is a user and a camera filming the participant's facial expressions while interacting with the program. Next to the user there was a facilitator. The role of the facilitator entails to welcome the users, introduce them to the experiment and make them comfortable and confident about the whole experiment procedure. Besides that, his role was to help the users in need, but that should be kept at a minimum, in order to get more fair results and observations. There was another camera behind the user and the facilitator, this camera was controlled by a cameraman. Its purpose served filming the whole setup by providing a bigger picture over the user-system interaction, as well as focusing on the PC screen (Fig. 32).

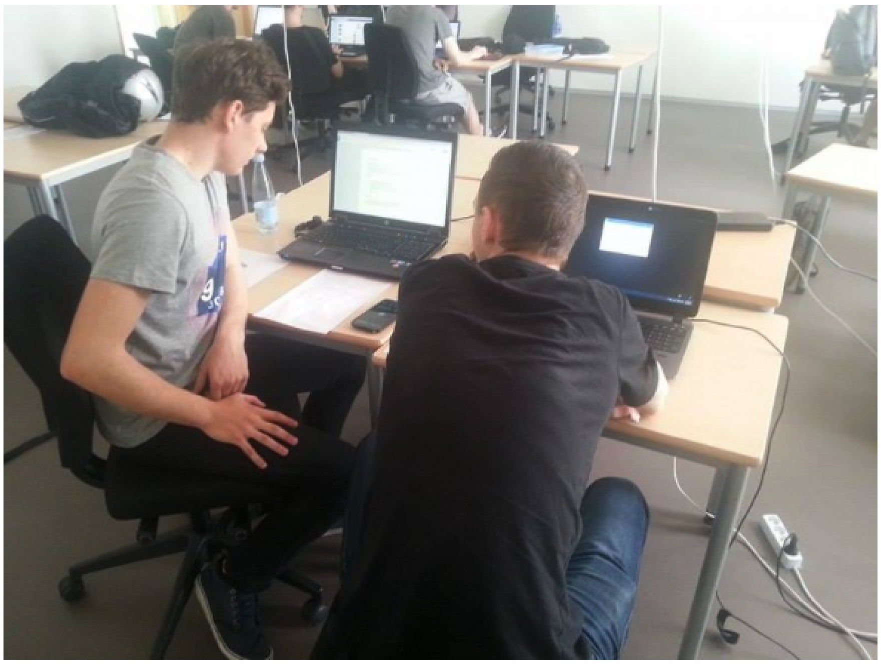


Fig. 32 Photo of usability test

The second phase consisted of questionnaire which aimed to collect the results of the test. It was handled by a questioner who was welcoming participants after they were done with the test.

Method

Firstly, the facilitator welcomes the participant and introduces him to the group and to the experiment. Then the participant is asked a few warm up questions to make him feel relaxed and then the test begins. It was also very important to receive the user's permission for filming the experiment as well as informing them about the anonymity of the experiment. Then the facilitator reads out predefined steps which the user has to complete while being timed. Those step are the following:

1. Find customer with phone number 20218819

2. Find and pick the board game Pandemic Legacy.

3. You change your mind and want to get another game instead. Hint: You want a Horror game for 5 players and for a big table.

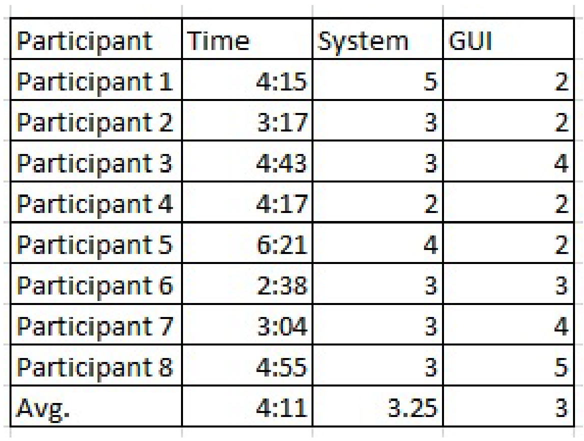
4. Pick a gametable for the same amount of players you selected previously. Select upstairs , date being today's date and book it for 5 hours. Then add everything to the booking.

5. Create the booking.

After those steps were completed each user was given complete freedom to interact with the system. The only information they received was a customer phone list. The idea behind this was to let them interact with the system however they want, thus revealing potential flaws in the system and in the GUI. Finally, the users were asked to give their feedback to the questioner.

Feedback and Reflections

A questionnaire was prepared in order to get each participant's feedback and thoughts on the system. All the testers were aged between 18 to 25, with a lot of background in using computer systems, and most of them being Multimedia Design majors and one being in Computer Science. When asked how efficient the system was most of the participants said it was fairly easy to use, although some were not sure enough what will pressing on something lead to. Different responses were given when asked about the design of the GUI, as its average rating was established at well designed. Fig. 33 shows the time of each participant, and their rating of the system and the GUI, one being very bad/hard and five being very good/easy.



Participant	Time	System	GUI
Participant 1	4:15	5	2
Participant 2	3:17	3	2
Participant 3	4:43	3	4
Participant 4	4:17	2	2
Participant 5	6:21	4	2
Participant 6	2:38	3	3
Participant 7	3:04	3	4
Participant 8	4:55	3	5
Avg.	4:11	3.25	3

Fig. 33 Test sumarry

From the experiment it is possible to draw a few conclusions about the whole concept of the system and what changes need to be made based on the participants feedback.

- At first sight, the system was not very intuitive, but as soon as the testers figured it out, it was handled a lot easier.

- The system needs to give more information to its users, as well as highlight important sections/buttons.

- The functionality of the system was found, in general, fine by most of the testers, as the expected functionality features were present.

- Some of the elements in the GUI, such as the hours in the Game Table table, were hard to understand or distinguish.

- The system lacked back buttons and error messages, which was noticed by most of the testers.

- The design of the system looked old fashioned and the testers found a bug, selecting a different start hour at the same table was not possible without selecting another table and then going back to the desired one, but apart from that the system was functioning well.

- Graphically, the system did not look aesthetic enough for most of the test participants

The results of the test gave a better overview on what needs to be added, changed and removed further. Even though there was a lot of constructive criticism, the concept of the system was appreciated by the testers. It can easily be evaluated that the GUI of the system needed most changes, as this was most disliked and criticized. However, this will be discussed later in another section (See sec. Future Development).

CONCLUSION

Future development

This section will cover what elements is the group planning to add in further development of the program.

Firstly, based on the feedback from the experiment, there is a lot of room for GUI improvements. Those include, improving it aesthetically and adding to it a more modern approach, which the user can relate to easily. Such changes are changing the colours of the interface, repositioning and resizing buttons, as well as adding highlights to most important parts. Adding back buttons and error messages is very essential and it was pointed out a couple of times during the test study.

During the design phase another GUI was designed, this was a bit more graphical interface, where the tables are shown with different colors, based on their status, and size could essentially help the user, especially if he is new to the system. (see Mock-ups.) The tables would remain green as long as they are free and turn red for the period of being booked. Showing the location and shape of the tables from top-down view could help the users pick a spot based on their wishes, for instance a table close to a window. In further development it would be nice to test out two or more different GUIs and finding out which would work best on same participants, is an option.

Another very important feature to be done is implementing the event system to the program. It is important not only because it is one of the main use cases designed in this project, but also it is a feature that the cafe needs to have.

Currently the system is not testing the availability of the chosen board game when the user is creating a booking. This is a fairly important test before program launch, because missing a board game the user was certain was booked by him would lead to angry customers.

A lot of the input fields does not give pop up warnings as was initially intended, for example a small pop up saying “No person with that phonenumber is in the database” or “Please add a phone number before you hit search”. The list of these missing messages are long, and would give a nice finishing touch.

Some clean up of the program on a design level would be nice, not calling connectionDB from so many different classes for example. Maybe having the main method inside the Menu frame instead of the connectionDB.

A lot of different features was explained but skipped in the implementation, these features are for example, the order part and the event part of the designed program, in future development adding all the designed features to give the customer and the cafe a full program would indeed be a priority.

Conclusion:

During the project, part of the designed system was implemented successfully. There is still a lot of designed functions that was not added, these has been discussed in the further development. The program is using Swing objects, it is a java program, and it uses both transactions and multiple threads in the final program. The database design is followed pretty accurately although the design of the java program and the final result does not follow each other as well, we will still conclude that the program manages to deliver what was promised, and with more time it would be easy enough to do a bit of clean up so it could follow the design better. The business parts hold all relevant information about the board game cafe and gives a fully understandable view of the company with some well formed artefacts. As does the system design follow the unified process as wished for by the problem statement. The group feels that the problem statement has been fulfilled fully.

Evaluation of the team effort:

The project team has used multiple team working exercises mentioned in the classes at school. Some of the practices from SCRUM has been picked to give the group a streamlined designing and development process, holding a friday meeting every friday, even when classes was still ongoing and following a certain schedule for all these meetings, changing SCRUM leader every friday so all group members were able to try the leadership role. The tasks for the SCRUM leader and the artefacts for the friday meeting can be seen in the Appendix 6, 7. Another artefact used by the group was trello which gives the group the opportunity to use an electronic task board, having digital post-it notes to show ongoing, done and to be done tasks so group members can add their name to a task and start working. Screenshot of Trello can be seen in Appendix 4.

In the beginning of the project a group contract was made, this contract consisted of several articles that should be followed by the entire group, the contract can be seen in Appendix 5. One of the interesting parts of the contract was a tardiness point, for every 10 min. someone was late for a meeting one point was added to the tardiness post-it in Trello, and then at the Friday meeting the one with the most points was assigned a small punishment, this could be anything from cake to a ticket to the board game cafe, to homemade pizza, the idea was that it should be something that helped the group having social events, to permit some relaxation together.

Halfway through the project the group started having some smaller issues, mostly small annoyances that went unsaid to each other, so the friday meeting was added another feature, "the hot chair" this was used as a venting mechanism for the entire group. A person would be in the hot chair and receive a compliment and a personal shortcoming from the other team members in turn. Without trying to defend himself, but taking the criticism and the compliment and then work with that. This exercise helped a lot with some of the problems the group was having and it actually helped people work on their shortcomings, for example a small anger issue one of the team members was having was almost completely eradicated.

The team had a schedule, showing the overall tasks of the group, and the more detailed tasks could then be found in Trello.

The different tasks was handled in smaller groups, in the beginning group members would change between the business group and the program design group, by the roll of a die online. Rolling a die for each group member to assign the member to either group. This made sure that everyone had a chance to work with all problems in the project and also working with all the different group members, avoiding two people to always work together without the rest.

REFERENCES

[1]http://www.redbooks.ibm.com/redbooks/pdfs/gg244338.pdf

[2]http://www.cio.com/article/2377681/time-management-productivity/14-tips-for-creating-a-paperless-office.html

[3]http://smallbusiness.chron.com/traditional-hierarchical-organizational-structure-26174.html

[4]http://smallbusiness.chron.com/advantages-disadvantages-using-volunteers-work-settings-36680.html

[5]http://dicendrinks.com/

[6] http://www.aalborgbraetspilscafe.dk/default.aspx

[7]http://www.free-management-ebooks.com/dldebk-pdf/fme-five-forces-framework.pdf

[8]http://www.entrepreneur.dk/DTU%20Patent%20Course%202015/Porter%205%20forces.pdf

[9]http://www.boardgamebliss.com/collections/all?sort_by=price-descending

[10]http://link.springer.com/chapter/10.1007%2F978-3-642-22098-2_54

[11]C.Larman. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)