



COPENHAGEN SCHOOL OF
DESIGN AND TECHNOLOGY

Mario's pizza bar

Link til GitHub Repository:

<https://github.com/seba167m/MariosPizzaBar>

Git hub – navne

Mohammad Adel Murtada

GitHub brugernavn: Mojens

<https://github.com/mojens>

Simon Igild

GitHub brugernavn: Chinesethunder

<https://github.com/chinesethunder>

Sebastian Bak Lundahl

GitHub brugernavn: Seba167m

<https://github.com/seba167m/>

Malthe Holm Karlsson

GitHub brugernavn: IfMalthereturnawesome

<https://github.com/ifmalthereturnawesome/>

Projekt Rapport

Marios Pizza

DAT21b

Project Participants:

Mohammad Adel Murtada

Simon Igild

Sebastian Bak Lundahl

Malthe Holm Karlsson

Contents

Vision	4
Business Rules / Domain Rules	4
Requirements	5
Functional and non-functional Requirements	6
Functional	6
Non-Functional requirements.....	6
Use Cases	7
Use Case diagram	7
UC02	7
UC04	8
UC01	8
UC03	9
System sequence diagrams	11
Håndtering af bestillinger - Fuldfør bestilling – Use case nummer: UC01	11
Håndtering af bestillinger – Rediger bestilling – Use case nummer: UC01.....	11
Håndtering af bestillinger – Slet bestilling – Use case nummer: UC01	12
Registrer bestilling – Use case nummer: UC02	12
Se ordre – Use case nummer: UC04	12
Class diagram	13
1. version af Class diagram	13
2. Final Class diagram	13
Sequence Diagrams	13
Sekvens diagram - Registrer bestilling - Main Scenario	14
Sekvens diagram - Fuldfør Ordre – Main Scenario	14
Sekvens diagram – Se Ordre - Main Scenario	15
Operation Contracts	16
Matrix Diagram	18
Glossary	19

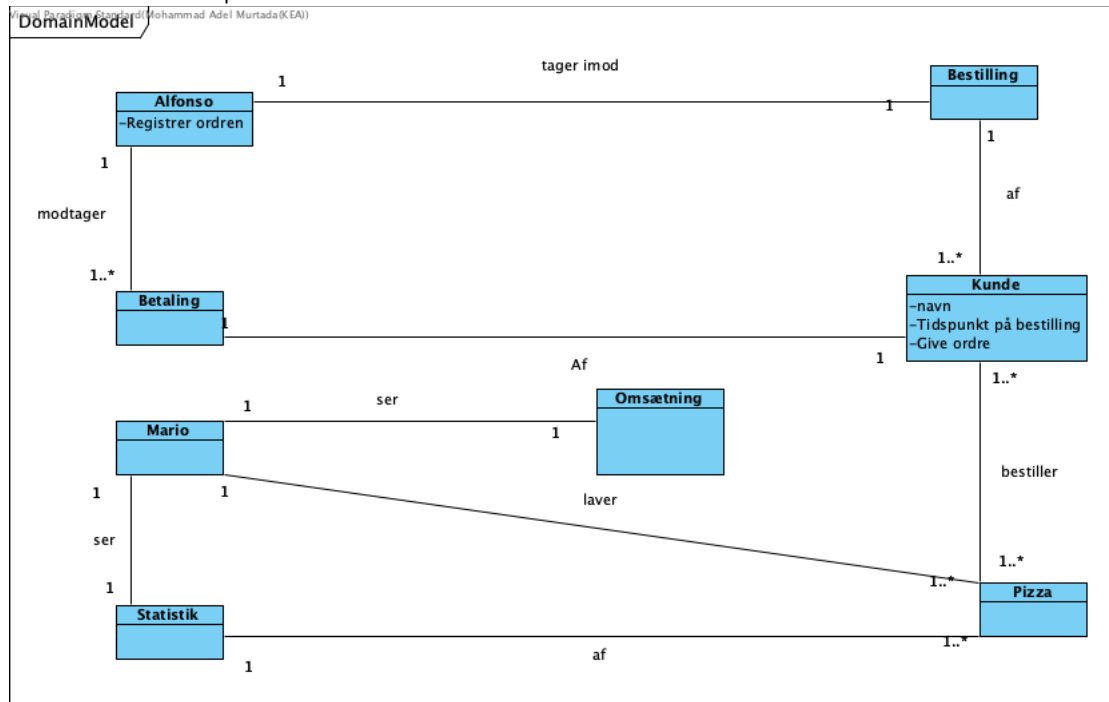
Vision

Et system der kan håndtere bestillingerne fra kunderne, hvor brugeren af systemet let kan se hvilke ordrer er blevet lavet og derudover, tilføje, slette og redigere bestillinger efter behov. Det skal ydermere, være simpelt at kunne se popularitet og omsætning for forskellige pizzaer.

Medmere, skal programmet kunne køre på en langsom computer og virke uden internet.

Business Rules / Domain Rules

Domainmodel over Mario's pizzeria:



Figur 1

Requirements

- Det skal være let så Mario kan benytte sig af det
- At kunne se hvilken pizza der nu skal laves
- At kunne se hvad tid pizzaen skal afhentes
- Skal kunne køre på Marios Gamle Dell Computer
- Skal kunne se en oversigt over omsætningen
- Man skal kunne indtaste ordrene
- Man skal kunne fjerne fra listen af ordre når der er betalt og afhentet
- Man skal kunne se en liste med bestillingerne
- At kunne se hele menukortet
- Den skal kunne sortere automatisk efter tidspunkt
- Man skal kunne se en statistik på hvilke pizzaer der er mest populære

I visual paradigm, har vi via tekstanalyse markeret vores requirements ud fra teksten som blev tildelt. Det har vi gjort således:

- Programmet skal bare kunne køre på hans gamle Dell-laptop, som står i pizzeriaet og ikke er på nettet.
- "Jeg kunne godt tænke mig hele tiden at kunne se en liste med bestillingerne, og hvornår de skal afhentes. Jeg har over 30 forskellige pizzaer, og de fleste kunder ringer og bestiller en times tid i forvejen. Nogle kommer også direkte ind i butikken.
- Det ville være rart at have mulighed for at se hele mit menukort på skærmen, så man kan se på det, når der er en kunde i telefonen eller ved disken.
- Min nevø, Alfonso, tager imod bestillinger, så han kan indtaste ordrerne. Han tager sig af at betjene computeren. Jeg skal bare kunne se listen over bestillingerne og på en eller anden måde få at vide, hvilken pizza, der nu skal laves. Måske kunne man sortere dem efter tidspunkt. Det kan jeg ikke helt gennemskue. Bare det bliver let for mig.
- Når jeg har lavet en pizza, skal jeg kunne råbe til min nevø, når pizzaen er klar. Så kan han fjerne den fra listen, når den er afhentet og betalt.
- Jeg vil gerne kunne gemme alle ordrerne, når de ekspederet. På den måde vil jeg kunne se omsætningen, og senere lave statistik på hvilke pizzaer, der er mest populære."

Figur 2

Functional and non-functional Requirements

Functional

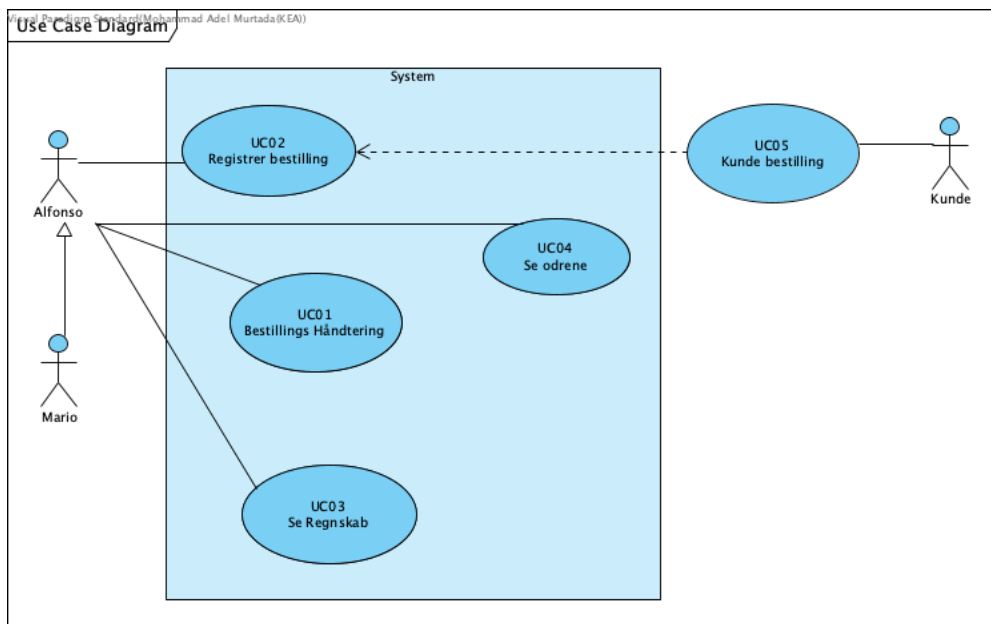
Id	Requirement	Comment
FR001	Skal kunne køre på Marios Gamle Dell Computer	Det er et krav fordi hans computer ikke kan køre noget stort.
FR002	Man skal kunne se en liste med bestillingerne	Da Mario skal lave pizzaer, så skal han kunne have muligheden for at kunne se oversigten over bestillinger.
FR003	Det skal være let så Mario kan benytte sig af det	Da programmet er lavet til Mario, skal det også kunne være noget Mario kan benytte sig af.
FR004	At kunne se hvilken pizza der nu skal laves	Det er nødvendigt for Mario at kunne se hvilken pizza der skal laves
FR005	At kunne se hvad tid pizzaen skal afhentes	Det er nødvendigt at vide hvornår kunden afhenter pizzaen så Mario kan planlægge at lave pizzaen.
FR006	Man skal kunne indtaste ordrene	Det skal selvfølgelig have mulighed for at kunne taste noget ind.
FR007	Man skal kunne fjerne fra listen af ordre når der er betalt og afhentet	Så man ikke roder rundt i det, så det nødvendigt at man kan slette afsluttede ordre.

Non-Functional requirements

Id	Requirement	Comment
NFR001	At kunne se hele menukortet	Det kunne være fedt at kunne se hele menukortet men dog er det ikke en nødvendighed
NFR002	Skal kunne se en oversigt over omsætningen	Det kunne være en fordel at man kan kunne se en oversigt over sin omsætning i stedet for at det er på papir
NFR003	Den skal kunne sortere automatisk efter tidspunkt	Det kunne være smart at den sortere ordrene ud fra tidspunktet den er blevet indtastet.
NFR004	Man skal kunne se en statistik på hvilke pizzaer der er mest populære	Dette er ikke en nødvendighed men det kunne være en fordel for Mario og pizzeriaet at han ved hvilken pizza der sælges mest af.

Use Cases

Use Case diagram



Figur 3

UC02

Use Case Name	Registrer bestilling
Scope	At kunne registrere en eller flere indkommende bestillinger fra kunder i butik og over telefon
Level	User-goal
Primary Actor	Alfonso
Stakeholders and Interests	Alfonso, som gerne vil have et system at registrere bestillinger i
Preconditions	Kunde bestillinger
Success Guarantee	Der skal modtages en bestilling før den kan registreres
Main Success Scenario	<div>En pizza:</div> <div><div>1. Kunde ringer eller går ind i butikken</div><div>2. Kunde bestiller hans ordre</div><div>3. Alfonso tager imod ordre</div><div>4. Alfonso taster ordrene in i systemet, ved at skrive tidspunkt for afhentning, nummer på pizza, kommentar, antal pizzaer</div><div>5. SYSTEM registrer ordren inde i systemet</div><div>6. SYSTEM sender nu ordre videre til Håndtering af bestillinger og Se Regnskab</div></div>
Extensions	<div>Flere pizzaer:</div> <div><div>1. Kunde ringer eller går ind i butikken</div><div>2. Kunde bestiller hans ordre</div><div>3. Alfonso tager imod ordre</div><div>4. Alfonso taster ordren in i systemet, ved at skrive tidspunkt for afhentning, nummer på pizza, kommentar, antal pizzaer</div><div>5. SYSTEM registrer ordren inde i systemet</div></div>

	6. SYSTEM sender nu ordre videre til Håndtering af bestillinger og Se Regnskab
Special Requirements	NFR001: At kunne se hele menukortet

UC04

Use Case Name	Se ordre
Scope	At kunne se nuværende ordre i systemet
Level	User-goal
Primary Actor	Mario
Stakeholders and Interests	Mario som gerne vil kunne se hvilke pizzaer han skal lave.
Preconditions	Registrer bestilling
Success Guarantee	Der skal være en liste med registret bestillinger før de kan håndteres
Main Success Scenario	<div>Der skal være modtages bestillinger før de kan ses i systemet:</div> <div><div>1. Kunde ringer eller går ind i butikken</div><div>2. Kunde bestiller hans ordre</div><div>3. Alfonso tager imod ordre</div><div>4. Alfonso taster ordren in i systemet, ved at skrive tidspunkt for afhentning, nummer på pizza, kommentar, antal pizzaer</div><div>5. SYSTEM registrer ordren inde i systemet</div><div>6. SYSTEM sender nu ordre videre til Se nuværende ordre</div><div>7. SYSTEM viser oversigt over alle registreret ordre og sorter efter afhentningstid.</div></div>
Extensions	null
Special Requirements	REQ004: Registrerer bestilling REQ001 Se en liste med bestillinger

UC01

Use Case Name	Bestillings Håndtering		
Scope	Redigere bestillinger Slette bestillinger Fuldfører bestillinger		
Level	User-goal		
Primary Actor	Alfonso		
Stakeholders and Interests	Alfonso, som har med kundekontakt af gøre. Han skal kunne håndtere kundernes ordre.		
Preconditions	Se liste med bestillinger Kunne fjerne ordre fra liste Registrer bestillinger		
Success Guarantee	Der skal være en liste med bestillinger før de kan håndteres		
Main Success Scenario	Fuldføre bestillinger: <table><tr><td>1. Kunde ringer til restauranten</td></tr><tr><td>2. Alfonso opretter bestilling på pizzaer</td></tr></table>	1. Kunde ringer til restauranten	2. Alfonso opretter bestilling på pizzaer
1. Kunde ringer til restauranten			
2. Alfonso opretter bestilling på pizzaer			

	<div><div>3. SYSTEM Opretter bestillingen</div><div>4. Mario Ser ordren i systemer og laver den</div><div>5. Mario Råber til Alfonso at ordren er færdig</div><div>6. Alfonso Kalder på kunden</div><div>7. Kunde betaler for sin ordre</div><div>8. Alfonso Vælger menuen til at fuldføre ordre</div><div>9. SYSTEM Spørg hvilken ordre han vil fuldføre</div><div>10. Alfonso Vælger kundens ordre og fuldfører den</div><div>11. SYSTEM Opdaterer ordre listen</div></div>
Extensions	<div>Slet bestillinger:<div><div>1. Kunde ringer til restauranten</div><div>2. Alfonso opretter bestilling på pizzaer</div><div>3. SYSTEM Opretter bestillingen</div><div>4. Kunde Ringer og vil annullere sin ordre</div><div>5. Alfonso Vælger menuen til at slette ordre</div><div>6. SYSTEM Spørg hvilken ordre han vil slette</div><div>7. Alfonso Vælger kundens ordre</div><div>8. SYSTEM Sletter ordren</div></div></div> <div>Rediger bestillinger:<div><div>1. Kunde ringer til restauranten</div><div>2. Alfonso opretter bestilling på pizzaer</div><div>3. SYSTEM Opretter bestillingen</div><div>4. Kunde Ringer og vil ændre sin ordre</div><div>5. Alfonso Vælger menuen til at redigere ordre</div><div>6. SYSTEM Spørg hvilken ordre han vil ændre</div><div>7. Alfonso Vælger kundens ordre og ændre den</div><div>8. SYSTEM Opdaterer ordren</div></div></div>
Special Requirements	<div>REQ001: Se en liste med bestillinger REQ004: Registrerer bestilling REQ006: Sortere dem efter tidspunkt REQ008: Så kan han fjerne den fra listen, når den er afhentet og betalt</div>

UC03

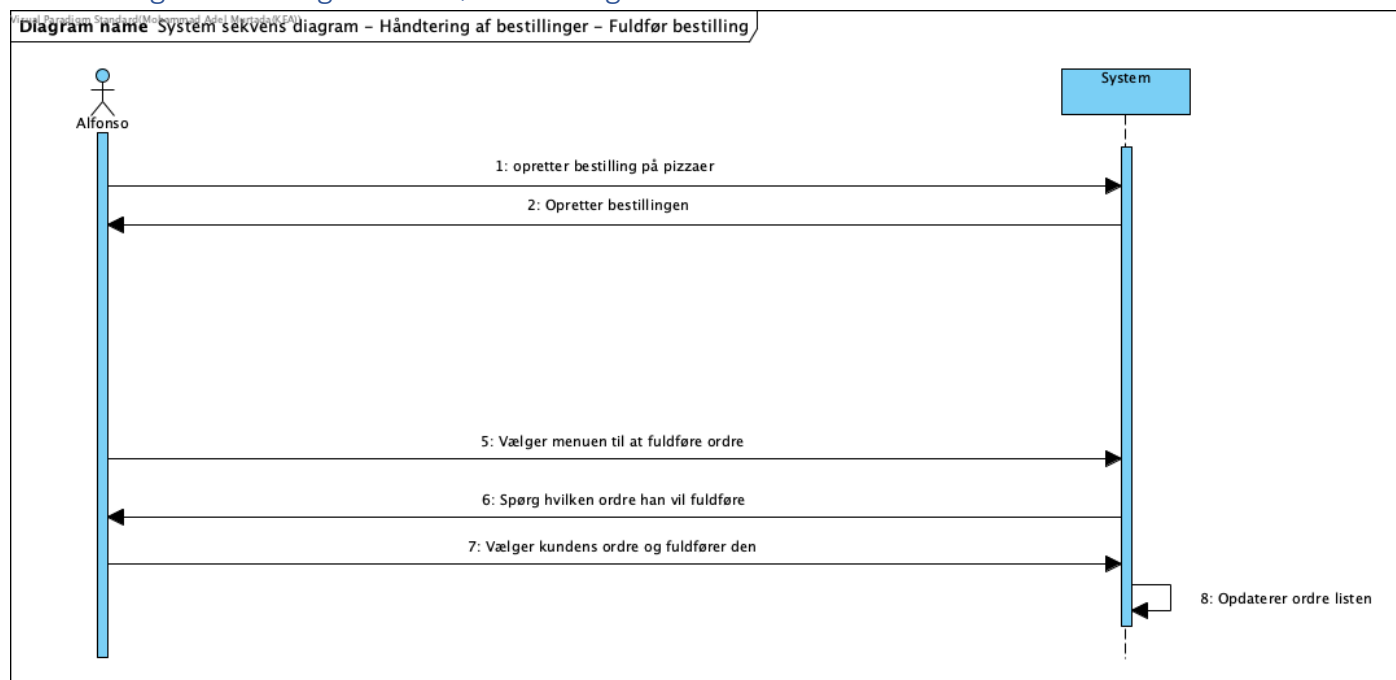
Use Case Name	Se regnskab
Scope	Statistik Omsætning Ordreoversigt
Level	User-goal
Primary Actor	Alfonso og Mario. De skal kunne se regnskab I slutningen af dagen
Stakeholders and Interests	Alfonso skal kunne se fuldførte ordre Mario skal kunne se statik over mest solgte pizzaer og dagens omsætning
Preconditions	Registret bestillinger
Success Guarantee	At kunne have en oversigt over dagen
Main Success Scenario	Omsætning:

	1. Alfonso trykker på se omsætning
	2. SYSTEM Spørger om hvilken tidsperiode brugeren vil se
	3. SYSTEM Viser omsætning fra den valgte tidsperiode.
Extensions	Ordreoversigt:
	1. Alfonso trykker på se fuldførte ordre
	2. SYSTEM Spørger om hvilken tidsperiode brugeren vil se
	3. SYSTEM Viser fuldførte ordre fra den valgte tidsperiode.
	Statistik:
	1. Alfonso trykker på se populære pizzaer
	2. SYSTEM Spørger om hvilken tidsperiode brugeren vil se
	3. SYSTEM Viser populære pizza fra den valgte tidsperiode.
Special Requirements	REQ009:
	Kunne se omsætning
	REQ010: Statistik på hvilke pizzaer, der er mest populære

System sequence diagrams

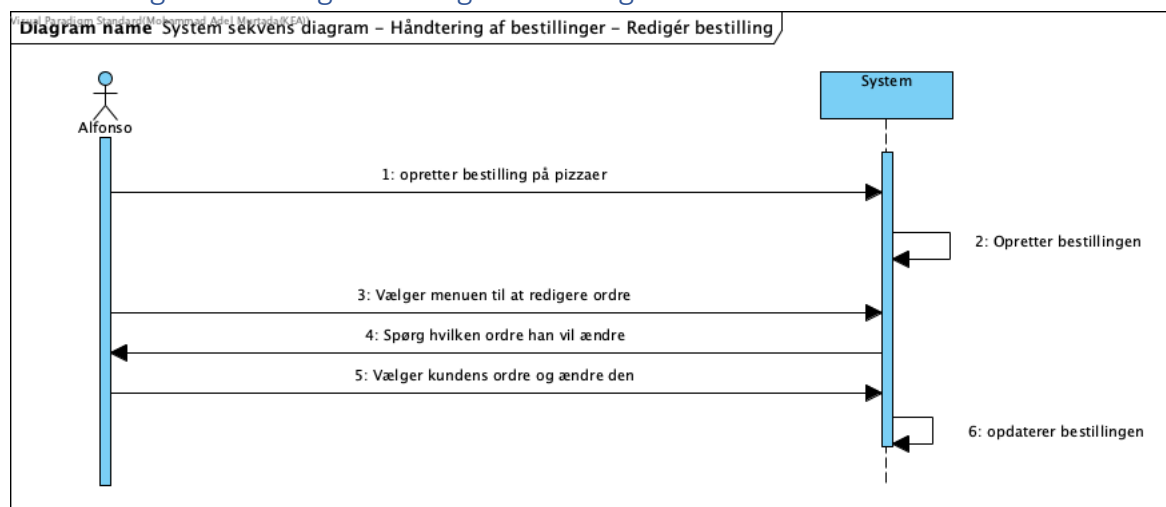
Her er 5 forskellige system sekvens diagrammer, udefra vores use cases.

Håndtering af bestillinger - Fuldfør bestilling – Use case nummer: UC01



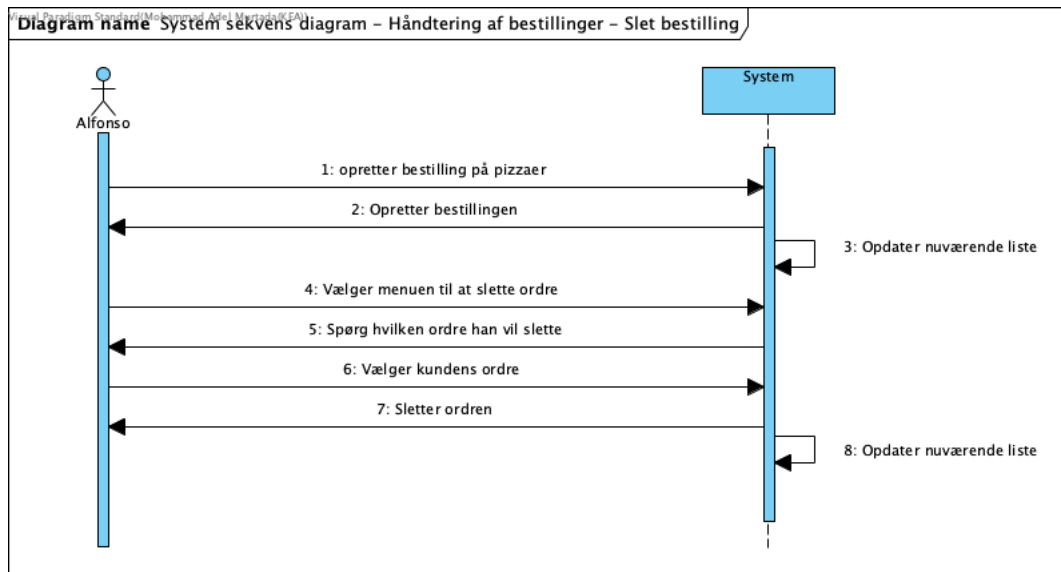
Figur 4

Håndtering af bestillinger – Rediger bestilling – Use case nummer: UC01



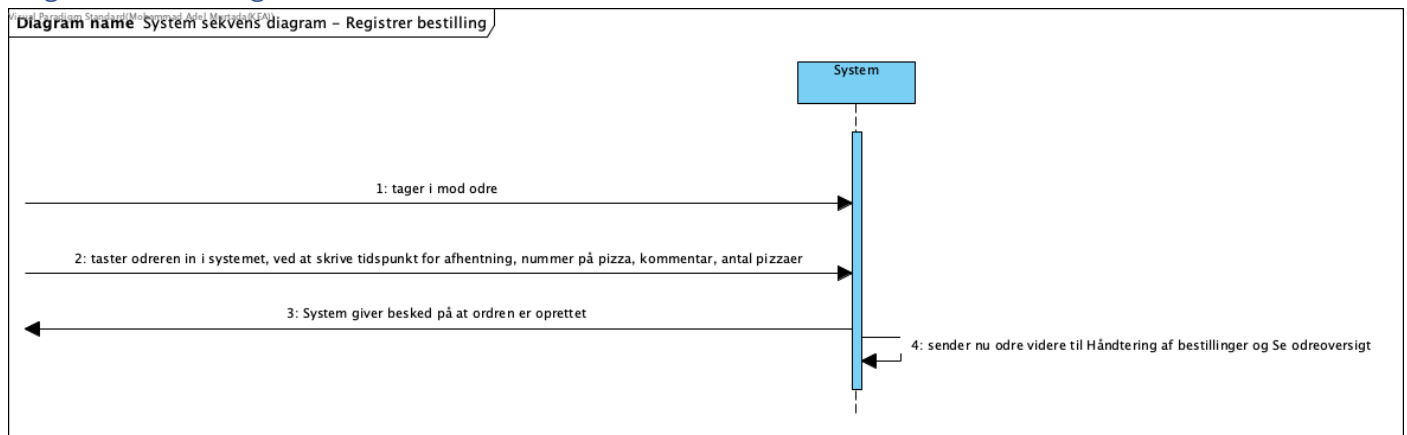
Figur 5

Håndtering af bestillinger – Slet bestilling – Use case nummer: UC01



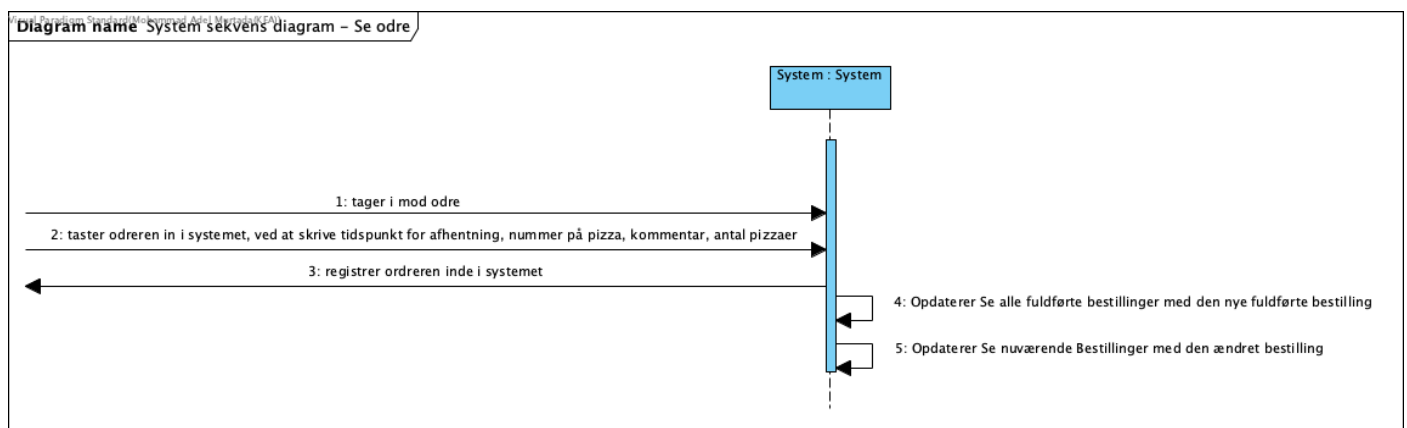
Figur 6

Registrer bestilling – Use case nummer: UC02



Figur 7

Se ordre – Use case nummer: UC04



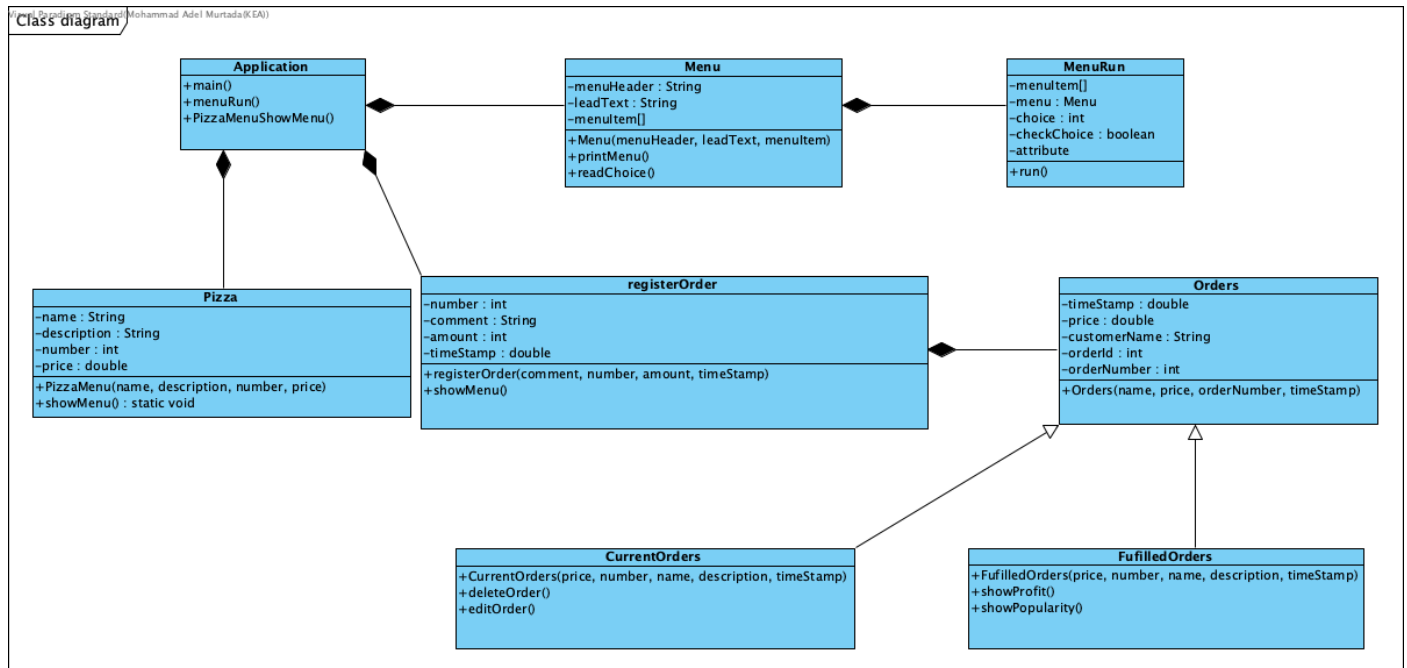
Figur 8

Class diagram

Vi lavede 2 Class diagrams.

Det ene var vores vision, hvilket var den vi lavede før vi gik i gang med at kode.

1. vision af Class diagram

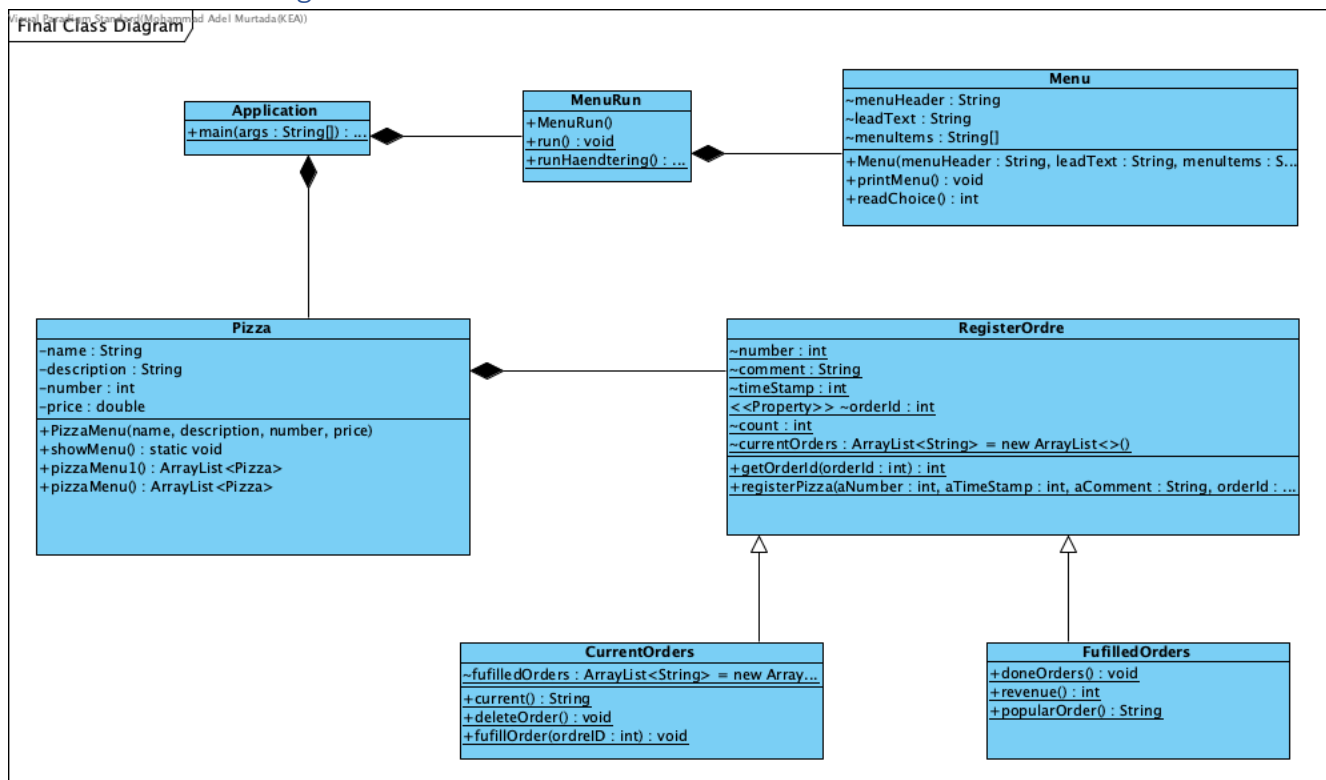


Figur 9

Men vores Class diagram, endte med at se anderledes ud efter vi havde kodet den.

Den endte med at se således ud som vist på Figur 10.

2. Final Class diagram

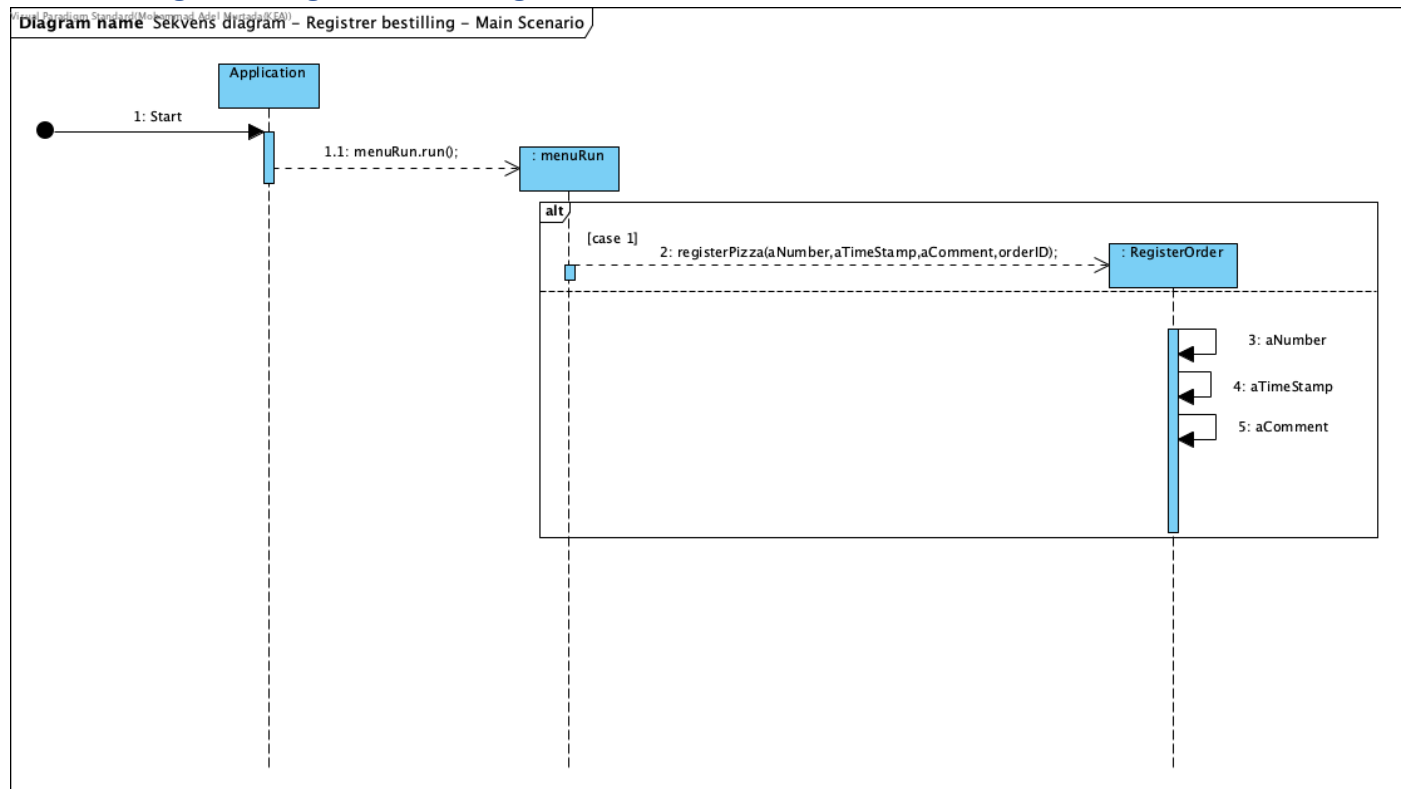


Figur 10

Sequence Diagrams

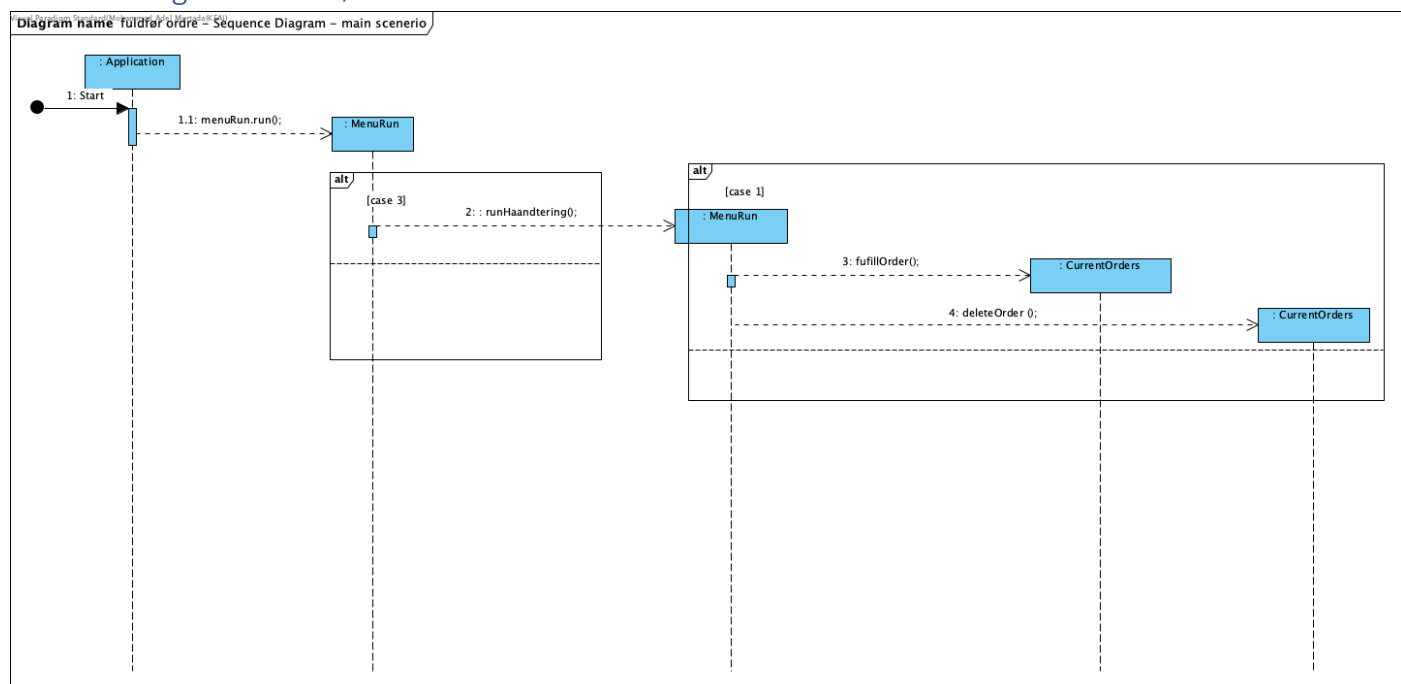
Vi har lavet 3 sekvens diagrammer ud fra 3 main succes scenerier.

Sekvens diagram - Registrer bestilling - Main Scenario



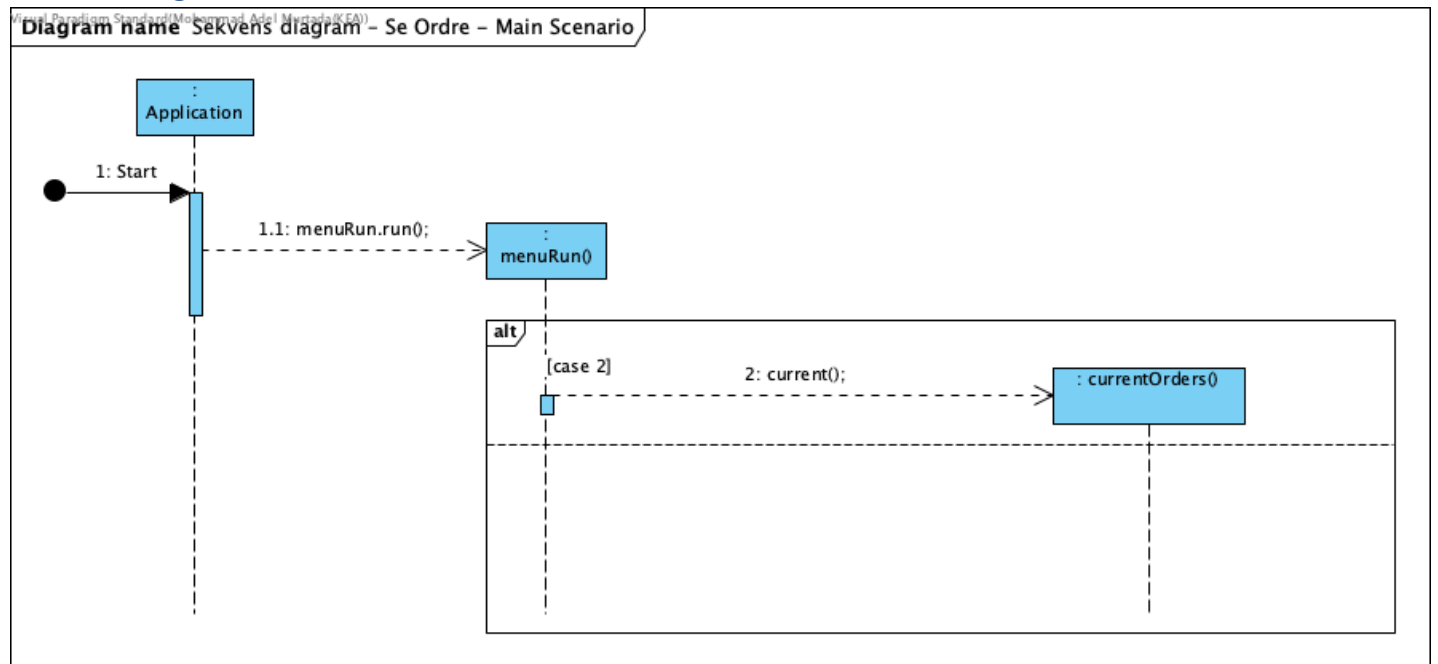
Figur 11

Sekvens diagram - Fuldør Ordre – Main Scenario



Figur 12

Sekvens diagram – Se Ordre - Main Scenario



Figur 13

Operation Contracts

Operation	Pizza (aNumber : Integer, aName : String, aDescription : String, aPrice : int)
Cross References	UC02 Registrer bestilling
Preconditions	Alfonso skal registrere en bestilling og det kan han ikke gøre uden en pizza objekt
Postconditions	Der er nu en pizza objekt der er klar til at blive registreret

Operation	pizzaMenu ()
Cross References	UC02 Registrer bestilling
Preconditions	Alfonso skal kunne have muligheden for at se menukortet, hvis han skulle glemme noget ved registrering.
Postconditions	Der er nu et overblik over alle pizza objekter, med deres informationer

Operation	ArrayList<Pizza>pizzaMenu1 ()
Cross References	UC02 Registrer bestilling
Preconditions	Når man registrer en bestilling skal man kunne referer til en pizza objekt, som kommer fra metoden.
Postconditions	Det er nu muligt at bruge arraylisten i metoden som referat når der skal registreres en ordre.

Operation	registerPizza (aNumber : int, aTimeStamp : int, aComment : String)
Cross References	UC02 Registrer bestilling
Preconditions	Når man registrer en ordre skal man kunne placere de ordrer i en liste med forskellige parametre som giver muligheden for at man kan sortere listen.
Postconditions	Det er nu sådan at efter man har registreret, en ordre så er den smidt over i en ny liste.

Operation	Current ()
Cross References	UC04 Se nuværende ordre
Preconditions	Man skulle kunne se alle de registrerede ordre.
Postconditions	Man skal nu kunne bruge denne liste til at slette og fuldføre en ordre.

Operation	deleteOrder (ordreID : int)
Cross References	UC01 Håndtering af bestillinger
Preconditions	Man skulle kunne være en ordre og slette den fra listen over nuværende ordre
Postconditions	Efter man har slettet ordren, skal den ikke kunne ses over nuværende ordre.

Operation	fulfillOrder (ordreID : int)
Cross References	UC01 Håndtering af bestillinger
Preconditions	Når man er færdig med en ordre, skal man kunne sende en ordre videre til en anden liste over fuldførte ordre.
Postconditions	Ordren er nu over i en anden liste som er fuldførte ordre, hvor der så kan blive udregnet statistik over omsætning og popularitet.

Operation	doneOrders ()
Cross References	UC03 Se Regnskab
Preconditions	Man skal kunne se en liste over fuldførte ordre.
Postconditions	Ordren er i listen over fuldførte ordre og de kan nu blive udregnet til statistik over omsætning og popularitet.

Operation	revenue ()
Cross References	UC03 Se Regnskab
Preconditions	Man skal kunne se et beløb over omsætningen på alle fuldførte ordrer.
Postconditions	Nu kan omsætningen blive vist.









Operation	popularOrder ()
Cross References	UC03 Se Regnskab
Preconditions	Man skal kunne se et overblik over alle ordre, og sortere dem efter pizza nummer og ordre id.
Postconditions	Nu kan man se en liste over fuldførte ordre som er sorteret efter pizza nummer og ordre id, på den måde kan man tydeligt se hvilken pizza der er mest berømt.

Operation	run ()
Cross References	UC02 Registrer bestilling & UC03 Se Regnskab
Preconditions	En menu så man nemt kan navigere rundt i de forskellige ting man kan lave i programmet.
Postconditions	Man får nu et overblik over de forskellige funktioner man kan foretage sig i programmet

Operation	runHaandtering
Cross References	UC01 Håndtering af bestillinger
Preconditions	En menu som specifikt viser de forskellige muligheder inde for use casen UC01 Håndtering af bestillinger
Postconditions	Man får nu et overblik over de forskellige funktioner man kan foretage sig i programmet, og det skal kunne køre ud fra de forskellige metoder.,

Matrix Diagram

Her har vi lavet en matrix diagram, som sætter vores requirements op i mod i vores use cases, og tjekker om vi har udfyldt de forskellige requirements i use casene.

	Bestillings Håndtering	Kunde bestilling	Registrer bestilling	Se odrene	Se Regnskab
 kunne se omsætningen					✓
 Registrerer bestilling	✓	✓	✓	✓	
 se en liste med bestillingerne	✓			✓	
 se hele mit menukort på skærmen			✓		
 sortere dem efter tidspunkt	✓				
 statistik på hvilke pizzaer, der er ...					✓
 System spørger om ordre		✓			
 Så kan han fjerne den fra listen, nå...	✓				

Figur 14

Glossary

Her er en glossary list i alfabetisk rækkefølge:

Name	Description
Application	<p>En applikation er et andet ord for et stykke software eller et program.</p> <p>Ordet applikation kommer fra det engelske ord applikation. I daglig tale bruges forkortelsen app i langt højere grad end ordet applikation. Især siger man app, når der er tale om applikationer til smartphones og tablets.</p>
ArrayList	<p>The ArrayList class is a resizable <u>array</u>, which can be found in the java.util package. The difference between a built-in array and an ArrayList in Java, is that the size of an array cannot be modified (if you want to add or remove elements to/from an array, you have to create a new one). While elements can be added and removed from an ArrayList whenever you want. The syntax is also slightly different:</p>
double	<p>The Java double keyword is a primitive data type. It is a double-precision 64-bit IEEE 754 floating point. It is used to declare the variables and methods. It generally represents the decimal numbers.</p>
Exstensions	<p>Extensions are important and normally comprise the majority of the text. They indicate all the other scenarios or branches, both success and failure. Observe in the fully dressed example that the Extensions section was considerably longer and more complex than the Main Success Scenario section; this is common.</p>
int	<p>The Java int keyword is a primitive data type. It is used to declare variables. It can also be used with methods to return integer type values. It can hold a 32-bit signed two's complement integer.</p>
Level	<p>In Cockburn's system, use cases are classified as at the user-goal level or the subfunction level, among others. A user-goal level use case is the common kind that describe the scenarios to fulfill the goals of a primary actor to get work done; it roughly corresponds to an elementary business process (EBP) in business process engineering. A subfunction-level use case describes substeps required to support a user goal, and is usually created to factor out duplicate substeps shared by several regular use cases (to avoid duplicating common text); an example is the subfunction use case Pay by Credit, which could be shared by many regular use cases.</p>
main	<p>In Java programs, the point from where the program starts its execution or simply the entry point of Java programs is the main () method. Hence, it is one of the most important methods of java and having proper understanding of it is very important.</p>
Main Succes Scenario	<p>This has also been called the "happy path" scenario, or the more prosaic "Basic Flow" or "Typical Flow." It describes a typical success path that satisfies the interests of the stakeholders. Note that it often does not include any conditions or branching. Although not wrong or illegal, it is arguably more comprehensible and extendible to be very consistent and defer all conditional handling to the Extensions section.</p>
menu	<p>A menu is a way to arrange <i>buttons</i>. There are several types.</p> <ul style="list-style-type: none">• Traditional dropdown menus are positioned across the top of a window in a <i>menu bar</i>, and display below the menu name.• Popup menus appear when the user clicks, <i>eg</i> with the right mouse button, on a component that can handle a popup request.
Preconditions	<p>state what must always be true before a scenario is begun in the use case. Preconditions are not tested within the use case; rather, they are conditions that are assumed to be true. Typically, a precondition implies a scenario of another use case, such as logging in, that has successfully completed. Note that there are conditions that must be true, but are not worth writing, such as "the system has power." Preconditions communicate noteworthy assumptions that the writer thinks readers should be alerted to.</p>

Primary Actor	The principal actor that calls upon system services to fulfill a goal.
REQ	During early requirements analysis, to get a quick sense of subject and scope. May take only a few minutes to create.
Run	When the run() method calls, the code specified in the run() method is executed. You can call the run() method multiple times.
Run2	When the run() method calls, the code specified in the run() method is executed. You can call the run() method multiple times.
Scope	The scope bounds the system (or systems) under design. Typically, a use case describes use of one software (or hardware plus software) system; in this case it is known as a system use case. At a broader scope, use cases can also describe how a business is used by its customers and partners. Such an enterprise-level process description is called a business use case and is a good example of the wide applicability of use cases, but they aren't covered in this introductory book.
Se Alle fuldførte bestillinger	
Se nuværende Bestillinger	
Special Requirements	If a non-functional requirement, quality attribute, or constraint relates specifically to a use case, record it with the use case. These include qualities such as performance, reliability, and usability, and design constraints (often in I/O devices) that have been mandated or considered likely.
Stakeholders and interests	This list is more important and practical than may appear at first glance. It suggests and bounds what the system must do.
Statistik på hvilke pizzaer, der er mest populære	
String	String is a sequence of characters. In java, objects of String are immutable which means a constant and cannot be changed once created
Success Guarantee	Success guarantees (or postconditions) state what must be true on successful completion of the use case—either the main success scenario or some alternate path. The guarantee should meet the needs of all stakeholders.
timeStamp	a digital record of the time of occurrence of a particular event.
Use Case	Use cases are text stories, widely used to discover and record requirements. They influence many aspects of a project—including OOA/D—and will be input to many subsequent artifacts in the case studies. This chapter explores basic concepts, including how to write use cases and draw a UML use case diagram. This chapter also shows the value of analysis skill over knowing UML notation; the UML use case diagram is trivial to learn, but the many guidelines to identify and write good use cases take weeks—or longer—to fully digest.
User-goal	Another value of use cases is that they emphasize the user goals and perspective; we ask the question “Who is using the system, what are their typical scenarios of use, and what are their goals?” This is a more user-centric emphasis compared to simply asking for a list of system features.
void	void means this method doesn't return a value. Methods can return a single value in Java and it has to be defined in the method declaration. However, you can use return by itself to exit the method.