NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for fysikk, informatikk og matematikk

Institutt for datateknikk og informasjonsvitenskap

BOKMÅL

Sensurfrist: 29. juni, 2007

Eksamen i fag
TDT4140 Systemutvikling

8. juni, 2007
kl 0900 - 1300

Hjelpemidler A1:
Kalkulator tillatt
Alle trykte og håndskrevne hjelpemidler tillatt:

Faglig kontakt under eksamen:
Professor Tor Stålhane, tlf. 94484

Poengene viser hvor mange poeng det er mulig å få på hver oppgave. Innen en oppgave teller deloppgaver likt, med mindre annet er angitt.

Lykke til!

.

Innledning

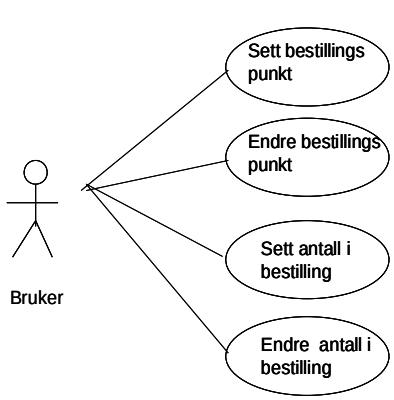
Overalt i oppgaven der vi bruker ordet ”system” mener vi systemet som er beskrevet i vedlegg A.

Dersom du trenger informasjon som ikke står i oppgaveteksten må du:

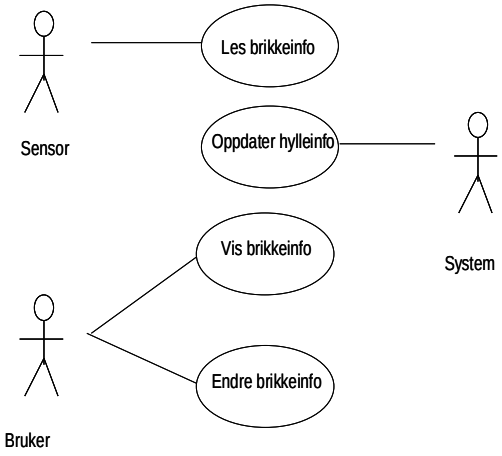
- Forklare kort hvorfor du trenger denne informasjonen
- Gjøre de nødvendige antagelsene. Disse antagelsene må beskrives i besvarelsen.

Oppgave 1 – Use case diagram, 25 poeng

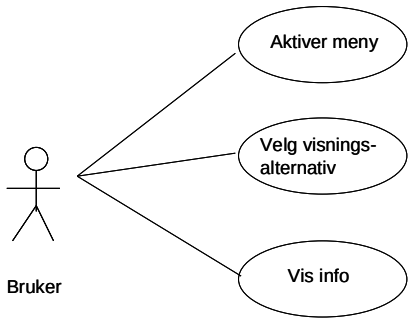
1. Lag use case diagram for alle funksjonene til det intelligente kjølerommet ”Cool Food”.



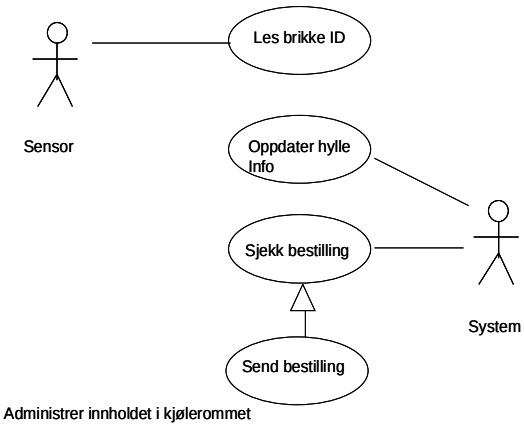
Automatisk varebestilling



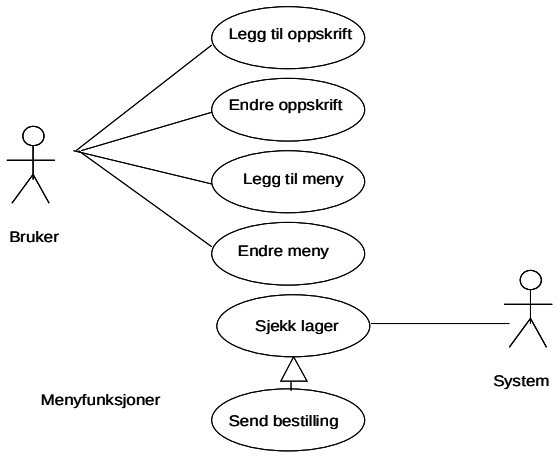
Registrering av boks i kjølerommet



Sjekke innhold i kjølerommet



Administrer innholdet i kjølerommet



I flere av use case diagrammene kan man velge å ha et use case som for eksempel inneholder både "legg til" og "endre". Dette gir færre use case, men dette må gjenspeiles i estimatene ved at hvert use case blir mer komplisert.

2. Lag tekstlige use case for følgende scenarier:
   - En boks med kjøttdeig blir tatt ut av kjølelageret.

Navn. En boks med kjøttdeig blir tatt ut av kjølelageret
Prebetingelser: det finnes en boks med kjøttdeig på kjølelageret
1. ID-brikka på boksen blir registrert av sensoren 2. Sjekk ID-brikka mot hyllelista 3. Brikka finnes i hyllelista => boksen blir tatt ut. 4. Identifiser innhold fra brikke ID (kjøttdeig) 5. Antall bokser med kjøttdeig dekrementeres med 1 6. Sjekk antallet mot bestillingspunkt 7. Nok igjen => avslutt
5.1 Ikke nok igjen => send bestilling 5.2 Gå til 6.

   - En oppskrift som er en del av ukemenyen blir endret.

Navn: Endring i en oppskrift som tilhører en eksisterende ukemeny
Prebetingelser: Ukemenyen finnes. Alt som trengtes til den opprinnelige versjonen av oppskriften er tilgjengelig på lageret.
1. Hent meny 2. Hent oppskrift 3. Utfør oppskriftsendringene 4. For alle varene i oppskriften som er endret: 5. Er det er nok for den nye oppskriften 6. Nok igjen av alt => avslutt
5.1 Ikke nok igjen av noe => send bestilling av det som mangler 5.2 Gå til 6

   - En person vil finne ut hva som er gått ut på dato i kjølelageret.

Navn: Hva har gått ut på dato
Prebetingelser: Logget inn på systemet
1. Velg funksjonen ”Vis det som går ut på dato innen to dager” 2. Systemet printer liste over bokser med relevant innhold samt hvilken hylle hver enkelt boks står i 3. Avslutt

Pass på at det er samsvar mellom use case diagrammene og de tekstlige use casene – for eksempel når det gjelder avvikshandtering.

Oppgave 2 – Planlegging, 30 poeng

Det er satt av tre personer til å gjennomføre prosjektet – du og to til. Du oppdager raskt at du er den eneste som har solid kunnskap om UML og Java. De andre to har et 14 dagers kurs fra EDB-høyskolen og har tidligere utviklet websider for lokale idrettslag.

1. Bruk use case estimering til å finne ut hva det vil koste å implementere systemet. Se vedlegg B.

Jeg har kopiert inn tabellene for å lett kunne fylle inn den informasjonen jeg har. Det er ikke nødvendig at studentene har med tabellene i besvarelsen, slev om det er et pluss. Det er ikke nødvendig å trekke mye viss de mangler en forklaring på hvorfor de har valgt de faktorverdiene de har valgt. Verdier på TCF, ECF, UUCW og UAW som har store avvik fra de verdiene jeg har brukt må imidlertid ha en rimelig begrunnelse.

Technical Factor	Description	Weight	Score
T1	Distributed system	2	1
T2	Performance	1	0
T3	End User Efficiency	1	0
T4	Complex internal Processing	1	0
T5	Reusability	1	0
T6	Easy to install	0.5	5
T7	Easy to use	0.5	5
T8	Portable	2	0
T9	Easy to change	1	0
T10	Concurrent	1	0
T11	Special security features	1	0

TCF = 0.67

Environmental Factor	Description	Weight	Score
E1	Familiarity with UML	1.5	2.3
E2	Application Experience	0.5	0

E3	Object Oriented Experience	1	2.3
E4	Lead analyst capability	0.5	5
E5	Motivation	1	5
E6	Stable Requirements	2	5
E7	Part-time workers	-1	0
E8	Difficult Programming language	-1	2.3

Har valgt å bruke en skår på (5 +1 +1) / 3 = 2.3 siden vi har en svært erfaren og dyktig person samt to personer med lite erfaring.

ECF = 0.77

Use case diagram	*Enkelt*	*Middels*	*Kompleks*
Automatisk varebestilling	***4***	***-***	***-***
Registrering av boks	***4***	***-***	***-***
Sjekk innhold	***3***	***-***	***-***
Administrer innhold	***3***	***1***	***-***
Menyfunksjoner	***2***	***4***	***-***
Totalt	***16***	***5***	***-***

Use Case Type	Description	Weight	Use cases
Simple	A simple user interface and touches only a single database entity; its success scenario has 3 steps or less; its implementation involves less than 5 classes.	5	16
Average	More interface design and touches 2 or more database entities; between 4 to 7 steps; its implementation involves between 5 to 10 classes.	10	5
Complex	Involves a complex user interface or processing and touches 3 or more database entities; over seven steps; its implementation involves more than 10 classes.	15	0

UUCW = 130

Actor Type	Description	Weight	Actors
Simple	The Actor represents another system with a defined API.	1	System, sensor
Average	The Actor represents another system interacting through a protocol, like TCP/IP.	2	
Complex	The Actor is a person interacting via an interface.	3	User

UAW = 5

UUCP = 135 og UCP = 135*0.67*0.77 = 69.6 use case poeng => 70*20 = 1400 timeverk eller 175 dagsverk, altså et "lite årsverk". Tre personer som jobber full tid vil altså trenge omtrent 59 dager eller 3 måneder på jobben. Dette er resultatene ved å bruke mest sannsynelige verdi for antall timeverk per justert use case poeng. Det er også mulig å bruke intervallet (15 – 30 tv) og angi et intervall for kostnadene i timeverk. Dette gir at antall timeverk blir mellom 1050 og 2100 timeverk eller 132 til 263 dagsverk.

2. Lag Ganttdiagram for utviklingen av dette systemet, gitt at dere er tre personer på prosjektet.

Det er mange måter å lage Ganttdiagrammet på, særlig siden jeg ikke har satt noen tidsfrist for når prosjektet skal levere. Det er imidlertid neon punkter som må være på plass i besvarelsen.

- ***Tabellen øverst på side 5 viser at det er stor forskjell på systemdelene. Vi ser for eksempel at automatisk varebestilling og registrering av boks er 20 ikke-justerte use case poeng, mens menyadministrasjonen er på 50 ikke-justerte use case poeng. Disse forskjellene må gjenspeiles i Ganttdiagrammet.***
- ***Det er viktig at det er en logisk sekvens på implementasjonen av funksjonaliteten. Dette er særlig viktig for mulighetene til å utføre tester. Det er derfor en dårlig løsning å starte med for eksempel menyfunksjonaliteten.***
- ***Det må være noe design og arkitekturaktivitet i starten av prosjektet – før man begynner med å implementere de enkelte funksjonene. Det må også være med en avsluttende systemtest.***
- ***Hver enkelt funksjon må ha aktivitetene design (kan være liten), implementasjon, testing og integrering med de andre funksjonene.***
- ***Planen må ta hensyn til at to av utviklerne er uerfarne. Det vil for eksempel være uheldig å kjøre tre parallelle aktiviteter fra starten.***

3. Like etter at planene er ferdig får dere beskjed fra "Cool Food" om at dere må ha "noe" ferdig til den neste utstillingen av kjøkkenutstyr som er i Hamburg om en uke. Lag en ny plan som gjenspeiler dette tilleggskravet. Du må selv velge ut de delene av systemet du mener det er viktigst å ha på plass til utstillinga.

Det finnes mange svar på dette spørsmålet. Viss svaret skal godkjennes må det tilfredsstille to krav: det må

- ***Være gjennomførbart innenfor tidsfristen – 5 – 7 dager***
- ***Ha funksjonalitet som demonstrerer de viktigste nye egenskapene – for eksempel koblingen mellom menyer og innhold i kjølelageret eller oversikt over innhold samt automatisk bestilling.***

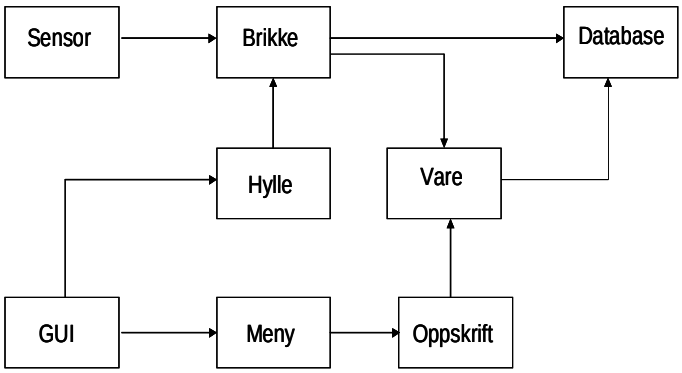
Jeg har valgt å lage en enkel prototype for utstillingsversjonen. Dette vil kost en ukes kalendertid, noe som ikke er kritisk siden det ikke er oppgitt noen leveringsdato for hele systemet. På plussiden vil prototypen kunne gi oss viktige erfaringer når vi skal bygge det virkelige systemet. Etter min mening bør prototypen ha følgende funksjonalitet:

- ***Sette bokser inn i en hylle***
- ***Lagre info om innhold. All info lagres i en enkel database – for eksempel Access***
- ***Ta fram info om innholdet i boksen***
- ***Ta bokser ut av hylle og se at det blir sendt ut bestilling viss det er for lite igjen***

Vi bør sette op et kjølerom med bare en hylle med tilhørende sensor og bare ha et enkelt Windows-basert GUI.

Oppgave 3 – Klassediagram, 30 poeng

1. Lag samtlige klassediagram for systemet på et nivå som passer til bruk tidlig i utviklingen av systemet – overordna design.

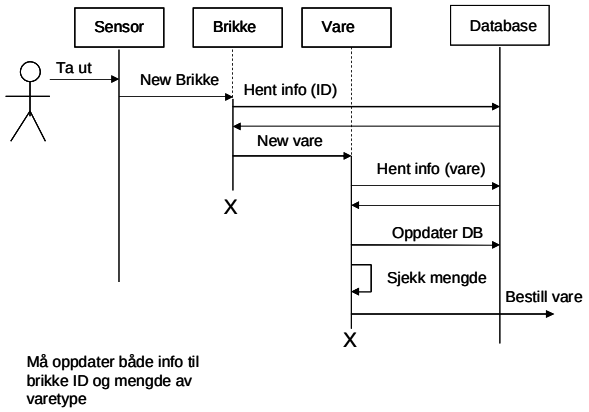


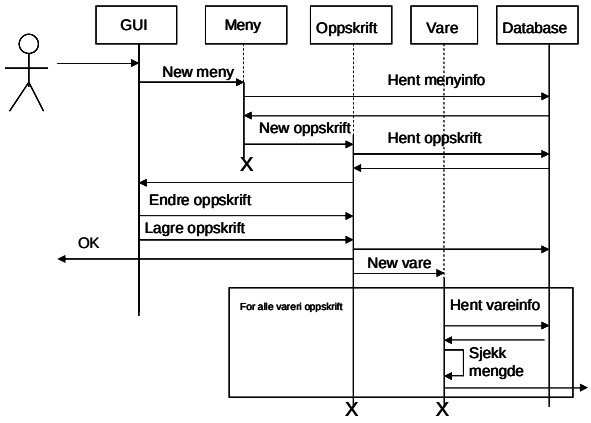
Det er mulig å lage et sett av klassediagram uten å innføre "Hylle" og "Vare". Dette er imidlertid en dårligere løsning som fører til mye ekstra info i klassen "Brikke" og kunstige koblinger med mellom klassene "Oppskrift og "Brikke". Denne løsningen bør derfor ikke bør få full skår.

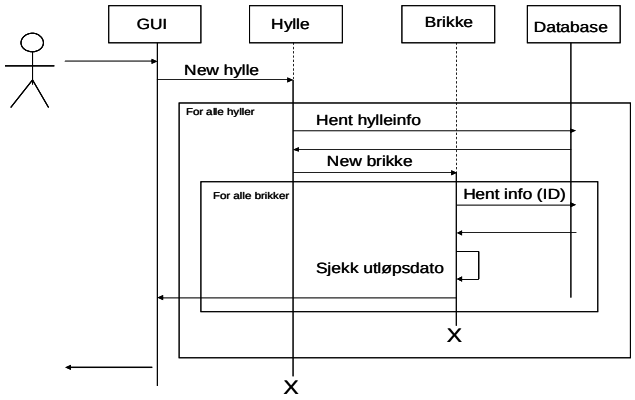
I diagrammet har jeg bare vist klassenavnene. Besvarelsen bør inneholde de viktigste attributtene for hver klasse, for eksempel:

- ***Brikke – utløpsdato***
- ***Meny – start og sluttdato***
- ***Oppskrift – nødvendig mengden av hver enkelt vare som inngår***

2. Lag sekvensdiagrammer for de scenariene som er beskrevet i oppgave 1, deloppgave 2.







I sekvensdiagrammene er det brukt UML notasjon for å vise looper. Det er OK å gjøre dette på andre måter også, for eksempel med kommentarer.

3. Lag sekvensdiagram for følgende scenario: En person tar ut en boks med gulrøtter fra kjølelageret. Systemet finner ut at antall bokser med gulrøtter er mindre enn bestillingspunktet og sender ut en ny bestilling.

Sekvensdiagrammet her bør likne på sekvensdiagrammet øverst til venstre, bare med gulrøtter i stedet for en generell vare.

Det er viktig at det er sammenheng mellom use case diagrammene, tekstlige use case, klassediagrammer og sekvensdiagrammer. Dette betyr for eksempel at:

- ***Det må finnes klasser for de objektene som er brukt i sekvensdiagrammene.***
- ***Der det er objekter som utveksler meldinger må det være assosiasjoner mellom de tilsvarende klassene i klassediagrammet.***
- ***Det må være samme mengde funksjonalitet i use case diagrammene og de tekstlige use casene. Der det finnes unntakshandtering i use case diagram må det også finnes i det tekstlige use caset og omvendt.***

Oppgave 4 – Testing, 15 poeng

Både vi og ”Cool Food” ser det som viktig at den delen av systemet som vises på messa ikke feiler i løpet av den tiden messa varer – sju dager. For å øke sannsyneligheten for en vellykka demonstrasjon vil vi gjennomføre en risikoanalyse og deretter gjennomføre et sett av tester for å prøve å forsikre oss om at de mest alvorlige risikofaktorene ikke vil lage problemer. Du blir satt på jobben.

1. Lag en risikoanalyse av en ukes demonstrasjon av det nye kjølerommet på kjøkkenutstyrsmessa i Hamburg. Identifiser de viktigste risikoene og definer preventive og / eller korrektive tiltak.

Foreslår følgende risikotabell. Det finnes flere andre mulige sett av risiki, avhengig av løsning, ambisjonsnivå for løsningen og analysenivå. Det som er viktig for besvarelsen er at alle de kolonnene som er brukt i understående tabell er med.

Hendelse	*Kons*	*Sans*	*Risk*	*Tiltak*	*Ansvarlig*
Sensoren registrer ikke ID-brikka inn eller ut	***H***	***L***	***M***	***Sett inn to sensorer som fungerer uavhengig av***	***Jon***

				hverandre	
GUI svikter	***H***	***M***	***H***	***Bruk så enkel GUI som mulig, uten at dette må gi inntrykk av at systemet er primitivt***	***Per***
Klarer ikke å ta fram info knyttet til innholdet i boksen	***H***	***L***	***M***	***Lagre info parallelt i database og i memory***	***Kalle***
Sender ikke bestilling ved for lavt lagerinnhold	***H***	***M***	***H***		***Torgrim***

2. Lag en enkel testplan for å sjekke at tiltakene virker som planlagt.

Svarene på dette spørsmålet avhenger av hva vi har gjort i risikoanalysen i spørsmål 1 – særlig hva vi har satt inn av tiltak. For mitt eksempel må vi utføre følgende tester:

- ***Legg inn boks i kjølerommet med begge sensorene OK. Sjekk at brikke-ID blir registrert. Registrer innholdet i boksen (pølser) via GUI.***
- ***Legg inn boks i kjølerommet med bare den ene sensoren OK. Sjekk at brikke-ID blir registrert. Registrer innholdet i boksen (pølser) via GUI.***
- ***Legg inn boks i kjølerommet med bare den andre sensoren OK. Sjekk at brikke-ID blir registrert. Registrer innholdet i boksen (pølser) via GUI. Det er nå tre bokser med pølser i kjølelageret.***
- ***Be om å få se hva som finnes av pølser på kjølelageret.***
- ***Stopp databasen og be om å få se hva som finnes av pølser på kjølelageret.***
- ***Start databasen, skriv over det reserverte memory-området og be om å få se hva som finnes av pølser på kjølelageret.***
- ***Sett bestillingspunktet for pølser til 3. Ta ut en boks.***
- ***Ta ut en boks til. Hva skjer nå?***

Eksperiment - frivillig

Dette spørsmålet gir ingen poeng og det teller ikke verken positivt eller negativt på din karakter. Vi har et forskningsprogram der vi – blant mange andre ting - ser på menneskers evne til å vurdere volum og kvalitet av eget arbeid. Dersom du vil hjelpe oss kan du skrive inn det antallet poeng du ***tror du får*** totalt på denne eksamen (alle fire oppgavene) nederst på siste side av besvarelsen din.

Det er viktig at sensor under rettinga registrerer dette tallet i en egen kolonne på regnearket

Vedlegg A - System for administrasjon av kjølerom

Firmaet ”Cool Food” leverer kjølerom til restauranter og storkjøkken. De har laget et nytt produkt som de kaller det intelligent kjølerom. Det nye produktet består av et kjølerom med:

- PC-tilknytning
- En brikkeleser (sensor) for hver hylle i kjølerommet. Brikken inneholder informasjon som kan leses av en sensor. Når brikken kommer nær sensoren vil sensoren lese brikkens ID.
- Bokser i alle nødvendige størrelser med brikker i lokket. Hver brikke her et unikt ID nummer.

Når en sensor leser ID-nummeret til en brikke blir dette nummeret sjekket mot listen av ID-nummer som er registrert på tilhørende hylla. Dersom brikkas ID-nummer allerede finnes i hylla skal systemet anta at den tilhørende boksen taes ut av hylla. Brikkens ID-nummer skal fjernes fra hyllas liste av ID-nummer. Dersom brikkens ID-nummer ikke finnes i hyllas liste skal systemet anta at den tilhørende boksen settes inn i hylla. Brikkas ID-nummer skal legges til hyllas liste av ID-nummer.

Følgende funksjonalitet skal implementeres:

- Automatisk varebestilling. Det må være mulig å
  - Sette eller endre bestillingspunkt for varene som kan oppbevares i kjølerommet. Når antallet av en vare når bestillingspunktet for varen må det bestilles mer av denne varen fra bedriftens leverandør av denne typen vare. Bestillingen sendes pr. elektronisk post.
  - Sette eller endre antall av en vare som skal bestilles når antallet når bestillingspunktet samt email-adressa til den som skal ha bestilling.

- Registrering av bokser som legges inn på kjølerommet:
  Det som skal oppbevares i kjølerommet legges i en av de boksene som følger med kjølerommet ved installasjon. Når boksen legges inn i en hylle i kjølerommet vil innholdet i brikken leses av hyllas sensor. Deretter vil ett av to skje:
  1. Informasjonen knytta til brikken er tom og det vil komme opp et skjema på PC-skjermen. Dette skjemaet kan fylles ut med informasjon og deretter lagres i systemets database. Denne informasjonen inkluderer blant annet mengde i boksen og holdbarhetsdato.
  2. Brikken inneholder informasjon som vises på PC skjermen. Det er mulig å redigere på denne informasjonen for deretter å lagre den i systemets database.

- Sjekking av innhold i kjølerommet:
  Ved hjelp av en kommando kan man få opp en meny der man kan be om å få vist informasjon om:
  - Hele innholdet i kjølerommet.
  - Alt som er i kjølerommet av en definert kategori, for eksempel grønnsaker.
  - Alt som er av en bestemt vare, for eksempel øl av merket VB.
  - Alle bokser med innhold som vil overskride holdbarhetsdato i løpet av de neste to dagene

- Funksjoner for å administrere innholdet i kjølerommet:
    - Sett inn ID-brikkas nummer i lista til hylla der boksen blir lagra når den blir satt inn i hylla.
    - Fjern ID-brikkas nummer fra hyllas liste når en boks blir tatt ut av hylla.
    - Sjekk gjenværende antall av varen mot bestillingspunktet for varen når en boks blir tatt ut av hylla. Viss antallet er likt eller mindre enn bestillingspunktet skal det sendes en bestilling.

- Meny funksjoner. Vær oppmerksom på at dette er menyer av mat, ikke menyer på PCen. Det skal være mulig å legge til, fjerne eller endre en eller flere
    - Oppskrifter på matretter. Oppskriftene må inneholde mengde og type av alle matvarene som inngår.
    - Ukemenyer med en eller flere retter hver dag. Ukemenyen har informasjon om startdato og sluttdato – for eksempel 1. april til 7. april – og antall planlagte porsjoner av hver matrett..
    - Når en ny meny blir lagt inn i systemet vil systemet sjekke om den har nok av alle matvarene og eventuelt bestille det som trengs for den kommende uken.

Vedlegg B - Use Case Points

Use Case Points (UCP) is an estimation method that provides the ability to estimate an application's size and effort from its use cases. Based on work by Gustav Karner in 1993, UCP analyzes the use case actors, scenarios and various technical and environmental factors and abstracts them into an equation.

The equation is composed of four variables:

1. Technical Complexity Factor (TCF).
2. Environment Complexity Factor (ECF).
3. Unadjusted Use Case Points (UUCP).
4. Productivity Factor (PF).

Each variable is defined and computed separately, using perceived values and various constants. The complete equation is:

```
UCP = TCF * ECF * UUCP * PF
```

The necessary steps to generate the estimate based on the UCP method are:

1. Determine and compute the Technical Factors.
2. Determine and compute the Environmental Factors.
3. Compute the Unadjusted Use Case Points.
4. Determine the Productivity Factor.
5. Compute the product of the variables.

Technical Complexity Factors

Thirteen standard technical factors exist to estimate the impact on productivity that various technical issues have on an application. Each factor is weighted according to its relative impact. A weight of 0 indicates the factor is irrelevant and the value 5 means that the factor has a strong impact.

Technical Factor	**Description**	**Weight**
T1	Distributed system	2
T2	Performance	1
T3	End User Efficiency	1
T4	Complex internal Processing	1
T5	Reusability	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent	1
T11	Special security	1

	features	
T12	Provides direct access for third parties	1
T13	Special user training facilities are required	1

Figure 1: Technical Factors.

For each project, the technical factors are evaluated by the development team and assigned a value from 0 to 5 according to their perceived complexity – multithreaded apps. require more skill and time than single threaded applications, for example, as do reusable apps. A perceived complexity of 0 means the technical factor is irrelevant for this project; 3 is average; 5 means that it has a strong influence.

Each factor's weight is multiplied by its perceived complexity to produce its calculated factor. The calculated factors are summed to produce the Total Technical Factor. To produce the final TCF, two constants are computed with the Total Technical Factor. The complete formula to compute the TCF is as follows:

```
TCF = 0.6 + (0.01*Total Technical Factor)
```

Environmental Complexity Factors

Environmental Complexity estimates the impact on productivity that various environmental factors have on an application. Each environmental factor is evaluated and weighted according to its perceived impact and assigned a value between 0 and 5. A rating of 0 means the environmental factor is irrelevant for this project; 3 is average; 5 means it has strong influence. Thus, if the participants are very familiar with UML, this will give value of 5 * weight = 7.5. In the same way, if the programming language is very well known, we get a value of 0 * weight = 0.

Environmental Factor	**Description**	**Weight**
E1	Familiarity with UML	1.5
E2	Application Experience	0.5
E3	Object Oriented Experience	1
E4	Lead analyst capability	0.5
E5	Motivation	1
E6	Stable Requirements	2
E7	Part-time workers	-1
E8	Difficult Programming language	-1

Figure 2: Example Environmental Factors.

Each factor's weight is multiplied by its perceived complexity to produce its calculated factor. The calculated factors are summed to produce the Total Environmental Factor. To produce the final ECF, two constants are computed with the Total Environmental Factor. The complete formula to compute the ECF is as follows:

```
ECF = 1.4 + (-0.03*Total Environmental Factor)
```

Unadjusted Use Case Points (UUCP)

Unadjusted Use Case Points are computed based on two computations:

1. The *Unadjusted Use Case Weight* (UUCW) based on the total number of activities (or steps) contained in all the use case Scenarios.
2. The *Unadjusted Actor Weight* (UAW) based on the combined complexity of all the use cases Actors.

UUCW

Individual use cases are categorized as Simple, Average or Complex, and weighted depending on the number of steps they contain - including alternative flows.

Use Case Type	Description	Weight
Simple	A simple user interface and touches only a single database entity; its success scenario has 3 steps or less; its implementation involves less than 5 classes.	5
Average	More interface design and touches 2 or more database entities; between 4 to 7 steps; its implementation involves between 5 to 10 classes.	10
Complex	Involves a complex user interface or processing and touches 3 or more database entities; over seven steps; its implementation involves more than 10 classes.	15

Figure 3: Use Case Categories.

The UUCW is computed by counting the number of use cases in each category, multiplying each category of use case with its weight and adding the products.

UAW

In a similar manner, the Actors are classified as Simple, Average or Complex based on their interactions.

Actor Type	Description	Weight
Simple	The Actor represents another system with a defined API.	1
Average	The Actor represents another system interacting through a protocol, like TCP/IP.	2
Complex	The Actor is a person interacting via an interface.	3

Figure 4: Actor Classifications.

The UAW is calculated by counting the number of actors in each category, multiplying each total by its specified weighting factor, and then adding the products.

Finally, the UUCP is computed by adding the UUCW and the UAW.

Productivity Factor

The Productivity Factor (PF) is a ratio of the number of man hours per use case point based on past projects. If no historical data has been collected, a figure between 15 and 30 is suggested by industry experts. A typical value is 20.

Final Calculation

The Use Case Points is determined by multiplying all the variables:

```
UCP = TCP * ECF * UUCP * PF
```