Mink Farm Rapport



Faith, Høgni, Kaj, Søren & Jakob - DM79 – Projekt Gruppe 3

Denne rapport dokumenterer processen med at analysere, designe og implementere et it-system for en mink farmer. Vi anvender viden fra ITIO(IT i organisation), samt metoder og principper for OOA / D(Object Oriented Analysis og Design) til at implementere et it-system der kan hjælpe en mink farmer til at administrere oplysninger omkring syge minke i hans farm. Systemet er implementeret i java sprog og bruger MSSQL server som database.

**University College Nordjylland**

**Sofiendalsvej 60**

**9000 - Aalborg**

**05-09-2012 – 07-01-2013**

**Antal Sidder: 90**

Indholdsfortegnelse

[Kravspecifikation 3](#_Toc345188944)

[Kodekontrakt 5](#_Toc345188945)

[Java 5](#_Toc345188946)

[Navngivning 5](#_Toc345188947)

[Navngivning af variabler 5](#_Toc345188948)

[Navngivning af metoder 5](#_Toc345188949)

[Klassestruktur 5](#_Toc345188950)

[Database/SQL 5](#_Toc345188951)

[SVN 6](#_Toc345188952)

[Mappestruktur 6](#_Toc345188953)

[Opsætningsbeskrivelse 6](#_Toc345188954)

[IT-Forundersøgelse 7](#_Toc345188955)

[Virksomhedskarakteristik 7](#_Toc345188956)

[Struktur 7](#_Toc345188957)

[Kontrolspændet 7](#_Toc345188958)

[Organisationskultur 8](#_Toc345188959)

[Virksomhedens interessegrupper 10](#_Toc345188960)

[Problemer, hypoteser og løsningsmuligheder 11](#_Toc345188961)

[SWOT Analyse 12](#_Toc345188962)

[Strategi Analyse 13](#_Toc345188963)

[IT strategi 14](#_Toc345188964)

[Applikationer og information 14](#_Toc345188965)

[Teknologi 14](#_Toc345188966)

[Organiseringen af IT-funktionerne 14](#_Toc345188967)

[IT handlingsplan 15](#_Toc345188968)

[Cost/Benefit 15](#_Toc345188969)

[Regnskabsanalyse 16](#_Toc345188970)

[Implementering 18](#_Toc345188971)

[Prioritering 19](#_Toc345188972)

[Brugerdeltagelse 19](#_Toc345188973)

[Vision og målsætning 20](#_Toc345188974)

[Tilpasning af organisationen/ledelsen 20](#_Toc345188975)

[Brugere af systemet 20](#_Toc345188976)

[Idé og mission 21](#_Toc345188977)

[System Design 22](#_Toc345188978)

[Inception Fase 24](#_Toc345188979)

[System Vision 24](#_Toc345188980)

[Situationsanalyse 25](#_Toc345188981)

[System Bruger Beskrivelse 27](#_Toc345188982)

[Produkt Oversigt 28](#_Toc345188983)

[Vigtige Features 28](#_Toc345188984)

[Use-cases 30](#_Toc345188985)

[Domæne Model 36](#_Toc345188986)

[Elaboration Fase 38](#_Toc345188987)

[Arkitektur 38](#_Toc345188988)

[Database Design 39](#_Toc345188989)

[Design Model 44](#_Toc345188990)

[Den mest kritiske use-case 46](#_Toc345188991)

[System Konstruktion 50](#_Toc345188992)

[Singleton Mønstre 50](#_Toc345188993)

[Implementering 51](#_Toc345188994)

[Perspektivering 58](#_Toc345188995)

[Konklusion 59](#_Toc345188996)

[Underskrift 59](#_Toc345188997)

[Bilag 60](#_Toc345188998)

[Samarbejdsaftale 60](#_Toc345188999)

[Tidsplan 61](#_Toc345189000)

[Regnskab 62](#_Toc345189001)

[Data Dictionary 85](#_Toc345189002)

# Kravspecifikation

**Formål**

Formålet med denne sektion er at specificere de krav der skulle være til det software-system som projekt gruppe 3 har fået til opgave at udvikle for Kjeld V Larsens Minkfarm i efteråret 05-10-2012

Kunde: Kjeld V Larsens Minkfarm.

Produkt: Et It-system der kan holde styr på diverse ting for en minkfarmer fx hvilke raser sidder hvor, hvilken kvalitet minkende er til avl, og diverse statistik. Ud over dette vil vi også bestræbe os om at kunne holde styr på syge mink, med hensyn til hvilken sygdom der er tale om og hvilket bur de skal overflyttes til.

Version: Dette er første version af systemet.

**Tiltænkte modtagere**

De tiltænkte modtagere af denne kravspecifikation er udviklerne af produktet i projektgruppe 3 samt Kjeld V Larsen.

**Produktets virkefelt**

Kjeld V Larsens minkfarm skal blive bedre organiseret og selve arbejdsprocessen skal blive forsimplet og nemmere at holde styr på hvor hvilke raser, sidder hvor men også hvilken kvalitet de hver især har. Det vil være lettere fra udefra kommende at overskue hvilke mink der er syge og hvilken sygdom de er ramt af.

**Produktperspektiv**

Et system der ligger på fast på en computer hvor der er internet adgang så man kan komme i kontakt med databaserne. I databaserne ligger informationer om de mink der er på farmen. Systemet er nyt og skal derfor ikke overtage funktioner fra et lignende program da Kjeld V Larsens minkfarm ikke har et i forvejen.

**Krav**

Logge på da der er mere end 1 mand i dette firma har vi valgt at der skal bruges et login så der kan ses hvem der har registreret hvad.

Man logger på med et brugernavn og kodeord.

Respons: Det opgivne brugernavn og kodeord kontrolleres, og hvis de passer sammen, kommer man ind på forsiden af programmet.

Alternativt forløb: Hvis brugernavn og kodeord ikke passer sammen, får kunden en tilbagemelding om, at data ikke var registreret, eller at han har indtastet forkerte data og kan forsøge igen.

**Kvalitet**

Programmet skal kunne holde styr på hvilken kvalitet hver mink er. Dette skal registreres af Kjeld V Larsen men når han har gjort det en gang skal systemet kunne holde styr på det for ham plus at hvis der kommer skader på en mink skal det registreres så når tiden kommer til pelsning kan man se hvilke mink der har skader og derfor ikke kommer til at koste så meget på auktion.

# Kodekontrakt

Al kode skal skrives på engelsk.

## Java

#### Navngivning af klasser

Klassenavne starter med et stort bogstav og UpperCamelCase i sammensatte ord.

### Navngivning af variabler

Alle variabelnavne skal skrives med små bogstaver og lowerCamelCase i sammensatte ord.

### Navngivning af metoder

På metoder åbnes og lukkes tuborgparanteserne under metodeheaderen. Loops, switch og så videre, åbnes tuborgparantes efter og lukkes under.

Alle metodenavne er skrevet i lowerCamelCase.

### Klassestruktur

* Klassebeskrivelse, indeholder: beskrivelse af klassen, navne på gruppen som initialer samt versionsnummer angivet som dato.
* Konstruktør: inputvariabler navngives som instancevariablerne, og sættes ved this-operatoren i konstruktøren.
* Getters og settere: getter og setter for hver instansvariabel skrives samlet. Getter først og setter efter. Således at de står i par.

### Database/SQL

Relation navne starter med et stort bogstav, og de sammensatte ord staves i UpperCamelCase.  
Attributnavne med små bogstaver.  
Alle SQL-queries skal skrives med små bogstaver.

## SVN

Alle commits indeholder kommentarer.

### Mappestruktur

En mappe til programmering hvor kun .java filer lægges op [Implementation].

En mappe til diagrammer, klar til revision af gruppen [UML Filer].

En mappe der indeholder den samlede rapport, færdige dokumenter [WordFiler].

### Opsætningsbeskrivelse

SVNMappe:

Implementation

UML Filer

Word Filer

# IT-Forundersøgelse

## Virksomhedskarakteristik

### Struktur

#### Organisationsprincip

Ejer

Medarbejder

#### Vurdering af organisationsstrukturen

Minkfarmen er en hierarkis organisation der er Kjeld, som er ejer og så er der hans medhjælper. Dog ville Kjeld altid være i toppen af pyramiden, hvor hans medhjælper godt kan risikere at blive rykket lidt til side, hvis der skulle blive brug for flere medhjælpere.

En hierarkisk organisationsstruktur er opbygget på den måde at alle udtagen den øverste led i pyramiden er bunden af en anden enhed. Jo større virksomheden er, jo flere led af mellemledere og ansætte vil der være i strukturen. Jo tættere man ligger på toppen af pyramiden des mere magt og indflydelse har man og længere ned man kigger desto mindre indflydelse har den ansatte.

#### Arbejdsdelingsprincippet

Minkfarmen er en lille forretning med kun 2 medarbejdere

### Kontrolspændet

Eftersom minkfarmen kun har 2 medarbejdere, så har virksomheden ikke et bredt kontrolspænd

### Organisationskultur

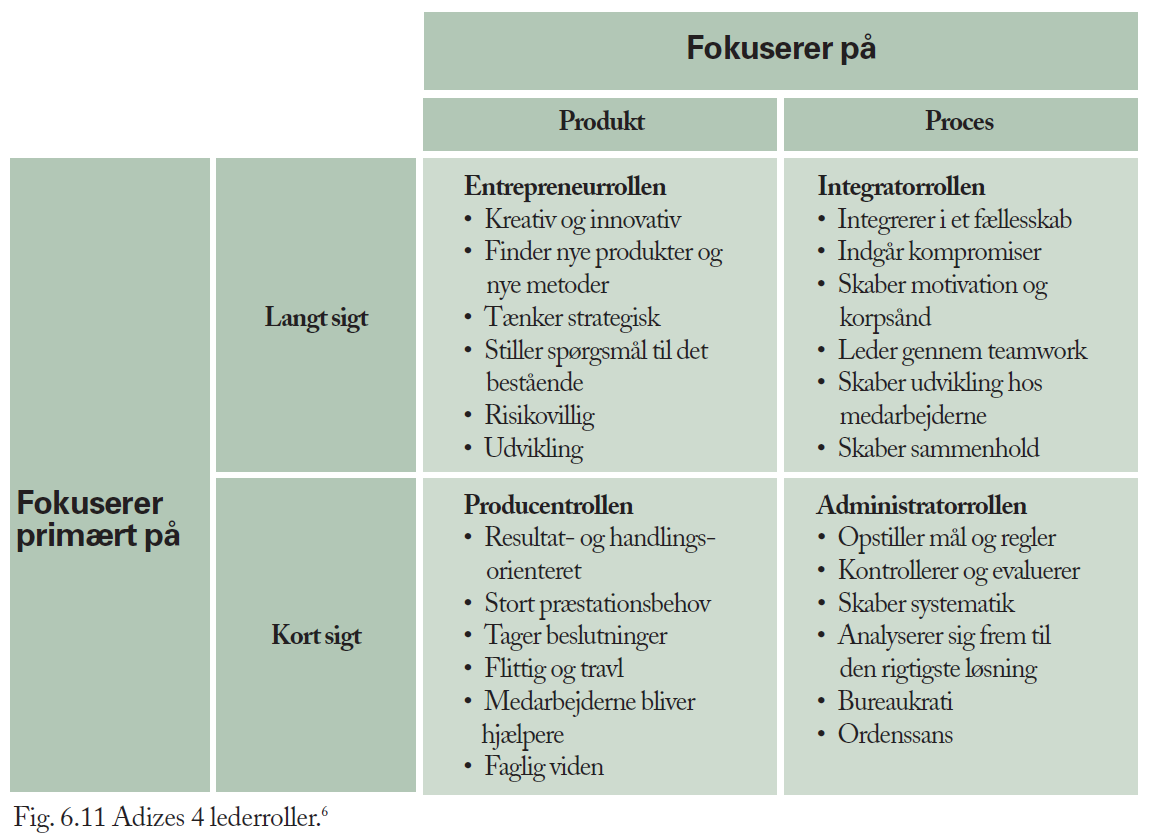
#### Lederkarakteristik

Kjeld er en klassisk jysk minkfarmer med fødderne solidt plantet på jorden, og selv om det er ham der er chefen er han ikke bange for at tage del i det beskidte arbejde. Han er dog ikke bange for at lade noget af ansvaret gå til hans kompetente medarbejder.

Hvis vi skal sætte Kjeld ind i Blake & Moutons ledergitter vil han være i Holdlederen da han og medarbejderen arbejder efter det samme mål nemlig at få de bedste mink skind der er muligt.

Kjeld er ”den autokratiske ” leder ifølge Jack Bobo’s ledelsesstile og det er han da det er ham alene der bestemmer når det kommer til stykker men han er ikke bange for at give ansvar til sin medarbejder, men hvis det ikke går som det skal bliver det på Kjeld metode.

I McGregors X-og Y-syn vil Kjeld have et Y-syn nemlig ved at han mener at arbejde er lige så naturligt for et menneske som at kunne hvile sig, men også at han giver sin medarbejder lov til at tage ansvar og gøre nogle ting som han mener det burde gøres.

Ud fra de 4 lederroller, som er beskrevet i Adizes’ model, passer Kjeld ind på producentrollen. Det gør han fordi han har masser af faglig viden og er resultat orienteret, da det kun handler om hvor godt et skind der kommer ud i sidste ende.

*1 Fig. 6.11, Organisation, Henrik Kjær, Hans Jørgen Skriver, Erik Staunstrup, side 177*

Kjeld’s kode vil så være som følgende: paEI fordi det er hans firma og derfor direktør.

Ud fra Lewins ledelsesformer kan man ikke sætte Kjeld ind i en specifik boks da han er en blanding mellem Autoritær og Demokratisk for godt nok er det ham der bestemmer det der skal gøres men hans medarbejder er ikke oprørske eller og Kjeld virker ikke upopulær hos sin medarbejder og der er ikke konflikter mellem Kjeld og hans medarbejder.

#### Virksomhedskultur

Virksomheden Kjeld V. Larsens Minkfarm, er et typisk familie firma, da forholdet mellem Kjeld og hans medarbejder er hierarkisk. Der er ingen tvivl om hvem, der har kontrollen, da Kjeld også har et lille landbrug ved siden af, kan det sagtens forekomme at Kjelds medarbejder ordner de praktiske ting i landbruget, hvor Kjeld ordner de ting, der er hos minkene. Dog sker dette slet ikke i de perioder hvor der er parring og pelsning af minkene.   
Derfor er firmaet et familiekulturs firma.

*2 Fig. 10.6 Organisation, Henrik Kjær, Hans Jørgen Skriver, Erik Staunstrup, side 301*

#### Konsekvenser for motivation og arbejdstilfredshed

*3 Fig. 3.9, Organisation, Henrik Kjær, Hans Jørgen Skriver, Erik Staunstrup, side 78*

I og med at minkfarmen kun har 2 ansatte Kjeld og hans medhjælper, er det svært at lave en konkret analyse over deres tilfredshed, vi prøver dog alligevel at kigge lidt på Herzberg’s teorier omkring Motivationsfaktorer og Vedligeholdelsesfaktorer. Selve arbejdet giver dem begge et ansvar, da de ikke rigtig kan sende opgaverne videre til de andre, da der ikke er andre end de 2. Der er ikke mulighed for nogen form for forfremmelse, med mindre at farmen bliver udvidet kraftigt, hvilket der ikke tanker om at gøre. For at være minkavler kræver det at man elsker sit arbejde, da der går rigtig mange timer med det, hvilket Kjeld også gør, han påpegede dog at de havde oplevede nogle ikke så gode år i starten af hans tid som ejer af farmen, men de seneste år har været helt forrygende for minkindustrien, hvor priserne bare et steget stødt år efter år. Dette har givet minkavlerne en god løn, hvilket ifølge Herzberg er med til at give en større tilfredshed blandt de 2 minkfarmere.

## Virksomhedens interessegrupper

De vigtigste interessegrupper, som minkfarmen har, er dens ejer og medarbejder. Dog kan den ikke fungere uden sine kunder, dog er det Kopenhagen Fur, som er en andelsforening blandt minkavlerne[[1]](#footnote-1). Kopenhagen Fur står for at sælge alle pelsene videre som hovedsagligt bliver solgt til Kina og Hong Kong – og udgøre 1/3 af den samlede danske eksport til Kina og Hong Kong.   
Minkfarmen kan ikke køre rundt uden sine leverandøre, da det er dem, der kommer med forskellige materialer så som foder, bure osv.   
En stor del af den succes de danske minkfarmere har, skyldes de standarder som Kopenhagen Fur har sat, og de regler som Det Dyreetiske Råd har skrevet i deres rapport ”Bekendtgørelse om beskyttelse af pelsdyr”.[[2]](#footnote-2)

## Problemer, hypoteser og løsningsmuligheder

|  |  |  |
| --- | --- | --- |
| **Problem** | **Hypoteser** | **Løsning** |
| Problemer hvis der er mandefald | Problemer i virksomheden når virksomheden bliver ramt af sygdom | Ansætte en fast vikar |
| Intet it-system | Information bliver skrevet manuelt ned på papir | Implementering af IT-System |
| Ingen IT erfaring | Har ingen viden eller erfaring med IT | Eventuelt et IT kursus |
| Har intet foder system | Minkene bliver fodret manuelt | Implementering af foder system |

*Tabel 1 Problemer, Hypoteser og Løsningsmuligheder*

*Denne tabel viser, hvordan et problem bliver gennemtænkt og en mulig løsning bliver valgt.*

## SWOT Analyse

|  |  |
| --- | --- |
| **Internal Conditions** | |
| **Strengths**  Medlem af Kopenhagen Fur  Gode bure  Stor viden  God kvalitet  Simpel organisation  Nemt ved at lave ændringer  Automatisk udmugnings system  Har landbrug ved siden af minkfarm  Alarmsystem | **Weaknesses**  Problemer hvis der er mandefald  Intet it-system  Har ingen it erfaring  Har ikke et foder system |
| **External conditions** | |
| **Opportunities**  Opgradere til et A/S eller ApS firma | **Threats**  Dyrt foder  Sygdom blandt dyr  Dyreaktivister  Varierende marked  Stor kontrol (hvis det går lidt galt går det hurtigt meget galt) |

*Tabel 2 SWOT Analyse*

*Ved brug af en SWOT analyse er det muligt at få et generelt overblik over virksomhedens styrker og svagheder, også kigger man på hvilke muligheder og hvilke trusler virksomheden har.*

*Da vi så danner et overblik over virksomhedens styrker (Strength), så har ejeren samlet sig en stor viden indenfor mink industrien da han har arbejdet som minkfarmer siden engang i 1980’erne. Med den viden minkfarmeren har skrabet sig sammen, så har han ingen problemer med fodring, hygiejne, sygdom, avling og også har han et godt indblik over hvilke kvalitets krav Copenhagen Fur har til minkpelsen som farmen producerer.*

*Minkfarmens svaghed (Weakness) består generelt af et manglende IT-system og virksomhedens manglende interesse for et.*

*Minkfarmens muligheder(Opportunities) er store. Hvis virksomheden implementerer et it-system så kan ejeren få et større oversigt med fx fodring, sygdom, arv og kvalitet.*

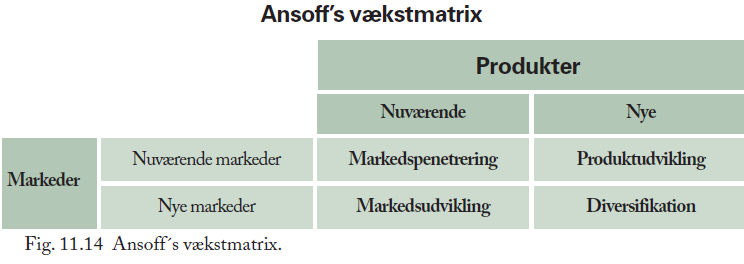
*De hoveds trusler (Threats) som Minkfarmen har, er de skiftende priser ude i marked. Med de skiftende priser så er der en vis usikkerhed over virksomheden da fodret også bliver dyrere og dyrere. Også er der meget stramt kontrol fra regeringen af, da de skal opfylde de krav som er blevet sat, så dyrevelfærd, fodring, og sygdom bliver holdt under kontrol.*

*Virksomheden burde satse på et IT-system, da det ville kunne gøre nogle forskellige opgaver lettere.*

## Strategi Analyse

Kjeld V. Larsens minkfarm vil investere i et IT-System da han ikke har et lige nu. Da han mener at det ville lette hans hverdag, hvis han ikke skulle gå og huske på alt hvad der skal gøres og ikke behøver at gå rundt med alle informationer omkring hver enkel mink i hovedet. Dette vil gøre at hvis enten han eller hans medarbejdere flytter en eller flere mink, kan de andre nemt gå ind og se hvor de nu er og hvilken race mink det er der er flyttet og hvor god minkens kvalitet er.

Når man kigger på Ansoff’s vækstmatrice vil man kunne aflede hvilken strategi Kjeld bruger.

Her mener vi at Kjeld V. Larsens minkfarm fokuserer på markedspenetrering og produktudvikling, i forhold til maxi-mini strategien[[3]](#footnote-3). Og dette begrunder vi med at kvaliteten hele tiden bliver lagt i højsædet og der bliver branded rigtig meget med at Danmark har de bedste skind i verden (dette er fra Kopenhagen fur’s side, Kopenhagen fur er en sammenslutning af danske minkfarms ejere). Kjeld prøver at avle flere arter og farver af mink og derved udvikle hans produkter så han ikke er afhængelig af at prisen på en farve er god eller dårlig.

*Fig. 11.14 Organisation, Henrik Kjær, Hans Jørgen Skriver, Erik Staunstrup, side 346*

## IT strategi

IT strategien skal understøtte forretningsstrategien. Den beskriver hvordan forretningsvisionen opnås gennem beskrivelse af behovet for applikationer, information og teknologi. Her beskriver vi hvordan strategien skal udmøntes i Kjeld V. Larsens minkfarm, ved bruge af teknologi, hvem der er ansvarlig og hvem der berøres.

### Applikationer og information

**På kortere sigt**

Der skal implementeres et IT system, der kan håndtere og erstatte nogle af de arbejdsopgaver på minkfarmen, som de løser manuelt nu, så som hvilke mink racer sidder hvor, samt hvilken kvalitet de hver især har. Der skal findes roller til medarbejderne i firmaet, som skal være tovholdere på IT‐systemet og vurderes om der skal suppleres med konsulenter.

**På længere sigt**

Systembruger i Kjeld V. Larsens farm skal opbygge kompetencer og erfaringer, så de kan udnytte systemet optimalt. Derved kan de reducere papirarbejde, reducere tid til information-flow gennem systemet, og være bedre til at beslutte, hvilken mink race skal parres, med sigte på at få de bedste kvaliteter.

### Teknologi

Dette nye system vi laver for Kjeld V Larsen kommer til at kræve at der er en computer på farmen samt computeren skal han også have en server til databaserne. De skal bruge disse servere til at holde styr på burene på farmen. På den måde vil der ikke ligge noget vigtigt på computeren, så hvis der sker et nedbrud på computeren vil Kjeld ikke miste noget information da det hele ligger på serveren.

### Organiseringen af IT-funktionerne

Organisation af IT-Funktionerne

Da der ikke var noget gammelt system er der ikke en system administrator og derfor påtager vi support og administrator rollen dette koster et mindre beløb om måneden.

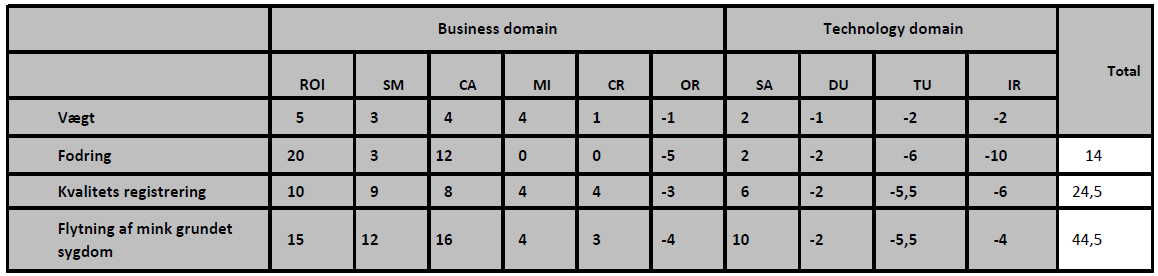
## IT handlingsplan

### Cost/Benefit

Kjeld V Larsens minkfarms formål med at implementere et IT-system vil være at effiktivitere arbejdsgangen ved at få lagt noget af den viden som Kjeld har ned på en database så det bliver nemmere og hurtigere at få fat på end at skulle spørge Kjeld. Lige nu har Kjeld V Larsens minkfarm ikke noget IT-system og dette gør at deres workflow tit bliver forstyret og dermed formår de ikke at få gjort så meget på den pågældende arbejdsdag som de skulle have. Med dette nye system skulle medarbejderen gerne kunne koncentrere sig om det landbrug der også indgår på farmen og derved vil de kun have en mand til at køre minkfarmen og derved vil de få et større overskud. Det bedste system de vil få ud af dette forløb vil være flytning af mink med henholdvis sygdomme eller hvis de er blevet raske skal de kunne flyttes tilbage til det bur de var i starten inden de fik bidsår eller blev testet positiv for plasmacytose.

i forbindelse med at systemet implementeres vil der være en lille reduktion i de ansattes effektivitet da der lige skal være en tilvendelses periode hvor systemet skal læres, der vil selvfølgelig være en introduktion til systemet. Det skulle dog gerne ende ud med at alle de ansatte på farmen vil være meget mere effektive.

#### Parker and Benson Matrice

Som det vises ude i kolonnen ”Total”, er der en klar oversigt hvilket system der er klart til at blive iværksat hurtigst muligt og hvilket system, der simpelthen ikke kan betale sig at implementere. Da Kjeld selv ikke mener at et foder system ville kunne hjælpe ham endnu mere end hans nuværende metode gør, har vi valgt ud fra hans argumentation og vores udregning, ikke at implementere et fodersystem.   
Igen har vi valgt at tage udgangspunkt i Kjeld’s argumentation og vores egen udregning til at sikre os hvilket system, som vil være det bedste at implementere hurtigst muligt da det netop er det, som kan give den hurtigste tilbagebetalingsværdi – dette system er Kvalitets statistik da det, vil gøre det lettere for Kjeld og gøre det mere overskueligt for besøgende. Dog ligger Sygdoms kontrol meget tæt på Kvalitets statistik, dette gør den fordi den på mange måder minder om kvalitets statistik, dog som man kan se i vores Parker and Benson Matrice har den aldrig fået flere point end kvalitets statistik.

### Regnskabsanalyse

For at danne os et overblik over hvordan økonomien hænger sammen i minkindustrien, er vi nødt til at kigge nærmere på diverse informationer, som vi har læst os frem til i Kopenhagen Fur’s årsberetning. Her under ses diverse nøgletal:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nøgletal** | Enhed | 2009 | 2010 | 2011 |
| Ændring i egenkapital | Mio. kr. | 592 | 756 | 829 |
| Soliditet | % | 58,6 | 70,8 | 58,2 |
| Afkastningsgrad | % | 4,25 | 19,57 | 16,1 |
| Kapitalens omsætningshast. |  | 4,37 | 7,22 | 6,13 |
| Overskudsgrad | % | 0,97 | 2,71 | 2,60 |
| Dækningsgrad | % | 91,95 | 94,98 | 95,30 |

I og med at omsætningen er steget stødt over de seneste 3 år, kan vi se at overskuddet samtidigt er steget – hvilket viser en forretning i rivende udvikling.

Hvis vi kigger på tallene er det tydligt at se at 2010 var et fantastisk år, hvor afkastningsgraden var 19,57 %, hvilket betyder at den samlede investering forrentes rigtig godt. Dog har der været udsvingninger i afkastningsgraden hvilket skyldes forandring i posteringen renter før skat, der lige som afkastningsgraden stiger stødt imellem ’09 og 10 og efterfølgende falder den minimalt fra ’10 til ’11. Samtidig med at afkastningsgraden falder fra ’10 til ’11 så stiger aktiverne med 356mil. Hvilket også har en effekt på afkastningsgraden.   
For at gå dybere ind i rentabiliteten vil vi nu se på overskudsgraden. Det er det nøgletal der viser os hvor god evnen i virksomheden er til at frembringe overskud. Tallet har sammenhæng imellem renter før skat og omsætningen. Ved at udregne overskudsgraden kan vi se at den på samme måde som afkastningsgraden stiger og falder. Dette skyldes her igen renterne før skat, hvor vi kan se udsving.

Virksomhedens dækningsbidrag er også stigende. Hvilket betyder at virksomheden tjener mere pr. solgte stk. Dette er også med til at øge overskuddet i virksomheden.

Ud fra gennemgangen af regnskabet vil vi vurdere at der grundet den øget egenkapital er grundlag for en investering i et IT-system. Dog skal det siges at dette ikke er Kjeld V. Larsens’ eget regnskab – da hans regnskab også indeholder hans personlige ting og derfor ikke ville være relevant og nok også svære at fremskaffe. Vi valgte derfor at søge trykhed i Kopenhagen Fur’s årsberetninger, hvor deres regnskab var at finde.

## Implementering

Det nye IT-system skal være så brugervenligt som det overhovedet er muligt, det skal det være, på grund af at jo mere brugervenligt det er, jo mindre tid og penge skal der bruges på at oplære med-arbejdere i det. Hvis det er meget brugervenligt vil det heller ikke tage ligeså lang tid at lære eventuelt nye medarbejdere i at bruge systemet, at et nyt IT-system er nemt og brugervenligt går også at medarbejdere der er vant til at bruge et andet system er villigere og motiverede til at lære at bruge det nye IT-system.



Når man ser på Leavitt’s systemmodel vil det hovedsageligt være delen ”Teknologi” der bliver ændret. Da der kommer til at være et IT-system og alt ikke bare skal huskes i hovedet. Medarbejderne og Kjeld skal selvfølgelig også lære programmet så der bliver også ændret i delen ”Personer” men det er også det der bliver ændret opgaverne på farmen vil forblive de samme og strukturen vil også være den samme.  
Vi vil så have support på systemet samt vi vil være administrator på systemet da der ikke er noget i virksomheden der har interesse i at være administrator.

*Fig. 12.15 Organisation, Henrik Kjær, Hans Jørgen Skriver, Erik Staunstrup, side 387*

### Prioritering

Som det ses ud fra vores Parker and Benson Matrice, har vi valgt at prioritere således, da den viser hvilke systemer, der er vigtigst for virksomheden.

1. Flytning af mink grundet sygdom
2. Kvalitets registrering
3. Fodring

## Brugerdeltagelse

Når man kigger på vores lederkarakteristik kan man se at Kjeld er placeret i McGregors Y-syn, og derved involverer han sin medarbejder så denne virksomhed er en virksomhed i harmoni. Og grunden til at den er i harmoni er bl.a. at der er enighed om at den her virksomhed skal levere et kvalitets produkt og ikke bare kunne levere mange skind. Og der er en stabil skruktur da alle ved at hvis der er et problem er det Kjeld der bestemmer men ellers ved medarbejderen hvad der skal gøres på en normal arbejdsdag.

*Fig. 12.17 Organisation, Henrik Kjær, Hans Jørgen Skriver, Erik Staunstrup, side 396*

*Fig. 12.18 Organisation, Henrik Kjær, Hans Jørgen Skriver, Erik Staunstrup, side 396*

## Vision og målsætning

**Kjelds vision er at:**

* Skabe et kvalitets produkt.
* Skabe en mere moderne og effektiv farm.
* Skabe en arbejdsplads der er mere personlig og med mere medansvar.
* Skabe en større konkurrenceevne i en mere global verden.

**Virksomhedens målsætning er:**

At få mere af den personlige viden ned i data så hvis der sker noget vil farmen stadig kunne køre videre. Og derved også at gøre det nemmere at komme ind som ny medarbejder.

### Tilpasning af organisationen/ledelsen

Da der ikke har været noget IT-system før dette har der ikke været en administrator før, og da Kjeld ikke mener han er specielt god til IT vil vi som konsulentvirksomhed oprette en support linje som Kjeld eller en af hans medarbejdere kan kontakte hvis der skulle komme et problem.

På grund af dette nye system vil der blive delt en masse viden som ellers ville blive hos den individuelle person og farmen vil blive mere produktiv da medarbejderne ikke hele tiden skal spørge og derved forstyrre Kjeld.

### Brugere af systemet

Interessenterne er Kjeld og hans medarbejder. De vil kunne være bedre til at få et hurtig overblik af hvilken mink racer sidder hvor og hvilken kvalitet de hver især har. Dette vil give Kjeld og hans medarbejder hurtig adgang til information omkring mink i farmen, en nemmere arbejdsproces eller en nemmere hverdag, samt at det vil øge information-flow i farmen.

### Idé og mission

Kjeld V Larsen vil producere et kvalitets produkt, han vil gøre dette ved at have en produktiv hverdag og medarbejdere han kan stole på og give ansvar. Måden han for den slags medarbejdere er at dele ud af hans viden omkring dette fag, men også at få medarbejdere der gider lære. Når han gør dette vil disse medarbejdere automatisk få mere ansvar og derved for Kjeld en mere produktiv hverdag. Kvaliteten i produktet for han ved over en længere periode at vælge de avlsdyr med de bedste gener men også at få nye avlsdyr ind hvert år.

Alt dette vil gøre at Kjeld fortsat vil være konkurrence dygtig i den her branche.

# System Design

Dette projekt bruger UP model, til udvikling af systemet, som er et krav for dette semesters projekt.

Unified Process (UP) er en iterativ og gradvis objektorienteret softwareudvikling proces ramme, der bruges til at modellere *hvad*, *hvem* og *hvornår* softwaren udvikles. Det bruges til at definere rollerne for hver person eller hvert team i projektet og de opgaver der skal udføres af disse personer eller teams. Det hjælper til at sikre kvalitet, minimere risici og reducere omkostningerne. UP er arkitekturcentreret og use-case driven.   
Hver fase består af en række iterationer. Antallet af iterationer afhænger af udviklingsprojektets kompleksitet.

UP definerer fire faser*:[[4]](#footnote-4)*

* Inception (Forberedelsesfasen)
  + Forberedelsesfasen er ikke kun en kravspecifikation, som det er i vandfaldsmodellen. I stedet for analyseres de mest kritiske krav og fastslå de grundlæggende idéer om systemet. Der bliver udarbejdet en risikoanalyse, hvilket er med til at vurdere om projektet kan gøres muligt. Normalt består denne fase af 1-2 iterationer.
* Elaboration (Etableringsfasen)
  + Fordybning af første fase. Der bliver her udspecificeret de valgte krav for første fase – ved at beskrive dem og målbare gøre dem. Ud fra risikoanalysen bliver de største risici fjernet.   
    Det er også i denne fase at den grundlæggende arkitektur for programmeringen bliver virkeliggjort og at tests bliver påbegyndt.

Denne fase består normalt af en del iterationer.

* Construction (Konstruktionsfasen)
  + Systemet bliver udviklet og de forskellige funktioner bliver testes – der bliver lavet dokumentation og en slags manualer.

Denne fase er den fase, som består af de fleste faser.

* Transition (Overdragelsesfasen)
  + Systemet bliver leveret. Brugeren bliver oplært i systemet.   
    Der bliver herefter evalueret på forløbet og processen kan afsluttes.  
    Denne fase består normalt af 1-2 iterationer.

|  |
| --- |
| *Figur 4 UP faseplan* |

## Inception Fase

### System Vision

Kjeld er tilknyttet Kopenhagen Fur, som er en international pels leverandør. Kopenhagen Fur er anerkendt for levering af høj kvalitet pels. De danske minkavlere er kendt for at producere en af de bedste pelskvalitet i verden, som kunderne er villige til at betale mere for. Derfor vil Kjelds minkfarm gerne forsætte med at forbedre deres produktion kvalitet og skræddersy deres produktion efter efterspørgslen på markedet. Kjeld vil gerne være mere konkurrence dygtig fordi hans omsætning ikke kun er afhængig af pelsens kvalitet, med også efterspørgslen på markedet.

Formålet med denne systemvision er at beskrive de overordnede krav til Kjeld samt at kunne afgrænse systemet da vi ikke kommer til at lave et fuldt system med alle use-cases. Mink farms system skal primært kunne hjælpe ham til at holde styre på information omkring de mink i hans farm. Disse informationer kan hjælpe Kjeld. til at forbedre kvalitet på mink produktion, samt at hjælpe ham med at imødekomme efterspørgslen på markedet og derved øge hans rentabilitet.

### Situationsanalyse

**Nu situationen (AS IS):**

**Tekstuel beskrivelse:**

* *Kjeld modtager nye mink(enten ved fødsel eller er købt)*
  + *1: Kjeld finder et bur til minken og skriver minkens oplysninger såsom fødselsår, arv, farve,… på et kort, der hænger på buret*
  + *2: Dyrlægen kommer to gange om året for at tage blodprøver og tjekker for sygdom, han tilbagemelder resultater fra blodprøven(i et brev)*
  + *3: Medarbejderne går rundt og tjekker hvert bur for bidesår. Hvis de finder nogle mink med bidesår, adskiller de mink med sår og flytter minken til en ny lokation (bur)*
  + *4: En gang om året, laver de kvalitets kontrol for at holde styre på information såsom hvilken arter, længden af pelsen, parringsinformation omkring mink, samt hvor de befinder sig. Information bliver skrevet ned*
  + *5: Efter omkring halvandet år og minken har været gennem kvalitetskontrol, er minken klar til pelsning. De pelser minken, registrere mængden af pelsen, samt pelsens type og levere pelsen til Kopenhagen Fur.*

**Problemer (AS IS) - Nu situationen**

Følgende kan også findes i Tabel 1 Problemer, Hypoteser og Løsningsmuligheder og Tabel 2 SWOTAnalyse

* **Ineffektivitet**

Det tager tid at gå rundt for manuelt at udfylde og hænge minkens information på buret

Det tager tid at gå rundt for at læse eller opdatere oplysninger på burene

Der er svært at huske de nye bure lokationer i hovedet, når mink bliver flyttet fra et bur til en anden

* **Fejl**

Der kan opstå fejl på grund af manuel nedskrivning og opdatering af data

* **Information tab**

Muligheden for at glemme ikke nedskrevne informationer, da de kun er i deres hoveder.

Det er svært at få et hurtig overblik af hvilken mink racer, der sidder hvor og hvilken kvalitet de hver især har.

**Forbedringsforslag (To Be)**

* Integreret IT system som kan gøre det muligt at gemme, bruge og opdatere data fra en database
* Integreret IT system som kan gøre det mulige for at se up-to-date antal of mink, deres arv, og hvor de befinder sig, dvs. hurtig adgang til Farm data
* IT system til registrering af mink, bur, mink lokation og til håndtering af minkens kvalitet samt arv information
* Information skal ikke længere skrives manuelt, men i et IT system for at spare tid, undgå redundans og undgå information tab.

**Forbedringsforslag (TO BE):** Aktivitetstabel efter IT

|  |  |  |  |
| --- | --- | --- | --- |
| Hændelse | Aktivitet | Step i aktivitet | Aktør |
| Ny mink modtaget | Registrere mink | Find bur til minken  Opret bur  Registrere arve information, farve, føde år …  Gem information | Kjeld/medhjælper |
| Tage blodprøve | Registrere blod test information | Tage blodprøve fra hver mink  Sende prøve til laboratorium  Modtage blodprøve resultat  Registrere blod prøve resultat i systemet  Flytte mink og registrere sygdom information og deres lokation i systemet | Kjeld/medhjælper |
| Tjek for bidesår | Registrere bidesår information | Kontrollere tilstanden af ​​hver mink  Adskille mink med bidesår  Registrere minkens sygdom information og deres lokation i systemet | Kjeld/medhjælper |
| Syg mink er blevet rask igen | Flytte rask mink tilbage til deres gamle bur | Check på status på mink med bidesår i systemet.  De raske mink blandt dem bliver flyttet tilbage til deres oprindelige bur, ved at bruge sygedom og lokation information fra systemet | Kjeld |
| Mink klar til salg | Levere mink | Efter at minken er blevet flået  Registrere mængden og kvalitet af pels i systemet  Levere pels til Kopenhagen Fur  Modtager meddelelse om at ordre er modtaget. | Medhjælper/  Kopenhagen Fur |
| Se status på farmen | View farm information | Systemet henter info fra hvert bur og viser dem på en side. |  |

### System Bruger Beskrivelse

Systemet vil have to brugergrupper: Ejer og Medhjælper.

|  |  |  |
| --- | --- | --- |
| Navn | Beskrivelse | Repræsentanter |
| Ejer | Har adgang til alle funktioner I systemet og håndtere medhjælperne informationer. | Farm ejer (Kjeld). Har en god erfaring med brugen af en computer og et IT System |
| Medhjælper | Ansvarlig for registreringen af informationer, så som at registrere nye mink, oprette nye bure og registrere overflyttede mink informationer I systemet | En medhjælper på Kjeld’s farm – har lidt erfaring med brugen af en computere og IT systemer |

### Produkt Oversigt

|  |
| --- |
| *Figur 5 Produktet skal kunne håndtere requests fra brugerne, sende svar til brugerne. Produktet bør også være i stand til at sende og modtage information til og fra eksterne IT-system.* |

### Vigtige Features

Det har en stor betydning for Mink Farm, at det nye system skal være brugervenligt.

Eftersom at personalet i virksomheden ikke har den største viden inden for IT, så skal systemet fungere på en måde, så de ansætte ikke skal bruge for meget tid til at lære systemet af kende.

De vil implementere nummer på deres bure, som vil gøre det nemmere for dem at holde styr på lige præcist hvad der står hvor i deres system.

Systemet skal også være sikkert og stabilt, da de ikke føler sig gode med en computer, og bare vil have at det virker som det skal.

En liste over de **Funktionelle krav** der er kritiske for Keld V. Larsens Mink farm:

Systemet skal kunne oprette nye bure med bur lokation.

Systemet skal kunne håndtere mink arter, aldre, farve, ...

Systemet skal kunne holde styr på syge mink information.

Systemet skal kunne holde styr på status på minks lokation, sygdom.. .

**Ikke Funktionelle krav:**

Systemet skal være brugervenligt.

Systemet skal være pålideligt.

Systemet skal fungere korrekt.

Systemet skal vise meningsfulde fejlbeskeder.

### Use-cases

Ud fra vores IT-forundersøgelse har vi kunne tage udgangspunkt i forskellige hændelser, som foregår i virksomheden. Dette resulterede i et use-case diagram.

Vi har 3 use-cases, som er CRUD og CRUD står for Create, Read, Update og Delete. Disse use-cases er "Manage Employees", "Register Hall", "Register Cage" og "Register Disease Report".

Vi har også andre use-cases og de er mere komplekse end CRUD og kræver mere af systemet og muligvis mere af aktøren. Ud fra vores diagram har vi følgende use-cases.

#### USE-case brief beskrivelser

**Manage Mink Info:**

* Medhjælperen registrerer minken og tilføjer den til systemet med div. Informationer så som (hvilken farve minken er, længde på pels, hvor tæt pelsen er, gener, om den har Plasmacytose eller har haft, om der er bidsår).

**View Farm Info:**

* Systemet henter info fra hvert bur og samler info til en side, samt farm information

**Schedule Blood Test:**

* Hvilke bure der skal testes hvornår og af hvem.

**Register Cage (CRUD):**

* Systemet opretter buret og tilføjer dets status og burets informationer.

**Register Hall (CRUD):**

* Systemet opretter hallen og tilføjer hal nummer og hallens informationer.

**Manage Employees (CRUD):**

* System opretter og registrere medarbejdernes information).

**Transfer Diseased Mink:**

* Systemet registrerer de bure hvor der er sygdom med dato og den medarbejder der har foretaget flytningen af de syge mink.

**Transfer Healthy Mink:**

* Systemet modtager blodprøverne og udsender en påmindelse på at der er raske mink blandt de syge og er klar til at blive flyttet tilbage til det gamle bur.

**View Auction Price:**

* Systemet bruges til at vise prisen på auktion priser.

**Update Auction Price:**

* Kopenhagen Fur opdaterer de sidste auktions priser. Det er et eksternt system, udenfor vores system.

**Register Disease Report**

* Systemet bruges til at registrere sygdoms informationer for de bure, hvor der er sygdom i.

#### Use-case prioritering baseret på forretningsværdi og tekniskkompleksitet

**Use-Case**: Use-case nummer

**Aktør**: En rolle, som personerne på arbejdspladsen tager/er

**Mål**: Hvad aktøren skal have IT-systemet til at udføre

**Forretningsværdi**: Vigtighed af use-casen for forretningen; niveauet for værdi de får ud af denne use-case

**Teknisk kompleksitet**: Hvor svært mht. hvor meget tid det vil tage at løse opgave

**Estimering af tid:** Hvormeget tid der er afsat til hver enkel use-case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use-Case ID** | **Mål**  **(Navn)** | **Aktør** | | **Forretningsværdi** | **Teknisk** **Kompleksitet** |
| Medarbejder | Ejer |
| UC1 | Manage Mink Info | X | X | 3 | 2 |
| UC2 | View Farm Info | X | X | 2 | 4 |
| UC3 | Schedule Blood Test | X | X | 3 | 2 |
| UC4 | View Auction Price | X | X | 1 | 2 |
| UC5 | Register Cage | X | X | 3 | 2 |
| UC6 | Register Hall | X | X | 3 | 2 |
| UC7 | Transfer Diseased Mink | X | X | 3 | 4 |
| UC8 | Transfer Healthy Mink | X | X | 3 | 2 |
| UC9 | Manage Employees |  | X | 1 | 1 |
| UC10 | Register Disease Report | X | X | 3 | 3 |

*Værdierne er repræsenterede med 1-4, hvor 4 repræsenterer de mest værdifulde/kompleks use-case og 1 repræsenterer de mindre værdiful/kompleks use-case.*

##### Valg af use-case

Vi har valgt at starte med UC7 da vi i vores Prioritering er kommet frem til at denne use-case er mest komplekse at implementere. Det er fordi der skal være en del sammenspil mellem forskellige elementer i systemet, såsom bur, medarbejder, og sygdom for at implementere denne use-case. En anden grund for at vælge denne use-case er fordi den havde en forretningsmæssig grund for firmaet vi laver programmet i samarbejde med. Den use-case giver mindre tids spild da medarbejderne ikke hele tiden skulle finde eller ringe til Kjeld som ejer firmaet for at få oplysninger der ikke var skrevet ned noget sted. Med denne use-case kan hver medarbejder gå ind og finde de oplysninger i stedet for at skulle spørge.

#### Fully Dressed use-case Beskrivelse af UC7

##### Transfer Diseased Mink

**UC7:** Transfer Diseased Mink

**Omfang og niveau:** Dette omhandler hvad der sker når der skal flyttes en mink fra et bur til et andet på grund af sygdom.

**Primær aktør:** Medhjælper

**Pre betingelser:** Der er tomme bure på minkfarmen

Der er sygdom i et bur og der er registreret medarbejder i systemet som skal foretage overførslen

**Post betingelser:** Minkene er blevet flyttet til et andet bur

Der er blevet registreret, det sygdom der findes i buret, det bur nummer og hvem har udført overførslen

**Basis succes flow:**

1. Medhjælper opretter en ny flytning
2. Medhjælper finder et bur
3. Systemet returnerer oplysningerne på buret
4. Medhjælper finder den bestemte sygdom
5. Systemet returnerer sygdom
6. Medhjælper indtaster sit ID
7. Systemet returnerer medhjælperen
8. Medhjælper gemmer flytningens informationer
9. Systemet registrerer den nye flytning

**Alternativt flow:**

* Systemet går ned

3b Buret findes ikke

4b Sygdom findes ikke

7b Medhjælper findes ikke

**Mockups:**

|  |
| --- |
| *Figur 6 Her ses det valgte mockup til use-casen* |

### Domæne Model

|  |
| --- |
| *Figur 7 Domæne model* |

Figur 7 Domæne model viser delvis domain model for Kjeld’s Mink Farm. Vi starter i denne iteration med UC7 som er Transfer Diseased Mink – UC7 kan ses i: Use-case prioritering baseret på forretningsværdi og tekniskkompleksitet

For at få et overblik over de klasser der skal modelleres for at håndtere UC7 har vi udarbejdet en domænemodel, der viser de konceptuelle klasser som har relevans i vores use-case. Her ses hvilke associeringer der er mellem Transfer, medarbejder, bur og sygdomme.

For at starte en transfer, skal følgende være sandt:

* Der er fundet sygdom i et eller flere bure. Bure har vi visualiseret med Cage domæne.
* Sygdom som er blevet fundet er enten et bidsår eller plasmacytosis. Sygdomme har vi visualiseret med Bite og Plasmacytosis domæner.
* Der er medarbejder der kan foretage overførsler. Medarbejderne har vi visualiseret med Employee domæne.

Dernæst skal der angives bur, sygdom og medarbejderens information på en transfer. Klassen DiseaseReport har vi tilføjet for at kunne registrere sygdom information og delvis hold styre på transfer.

For at kunne oprette en hal, så skal der eksistere en farm, da hallen skal bruge farmens CVR nummer for at oprette en ny hal.

For at kunne oprette en Column, så skal der eksistere en hal, da Column skal bruge hallens nummer for at oprette en ny column.

For at kunne oprette et nyt bur så skal der eksistere et column, da buret skal bruge columns nummer for at oprette et nyt bur.

For at kunne oprette en ny mink så skal alle de attributter minken har, udfyldes. Så skal der eksistere et ledigt bur. Buret kun kan indeholde max 4 mink i hvert bur.

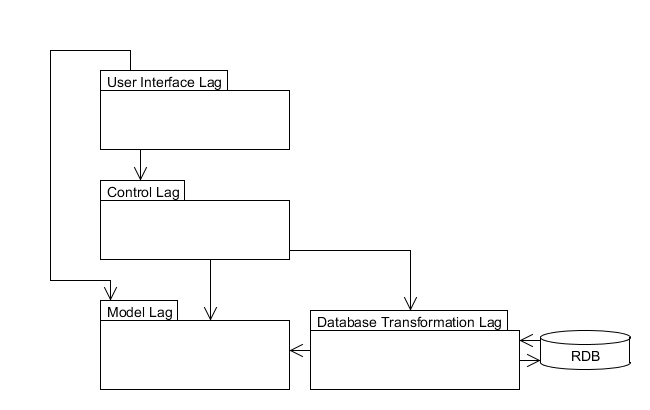
## Elaboration Fase

### Arkitektur

**Patterns:**

Patterns eller Mønstre er kendt løsninger på gentagne design problemer, som med mindre modifikationer kan anvendes i mange sammenhænge. Den fremmer udviklingen af cohesive moduler med minimal kobling. Vi vil referere til nogle design mønstre vi overvejede under vores design.

Vi har valgt at strukturere vores IT-System ved at anvende 4 lags arkitektur. Vi valgte at bruge et lag arkitektur, fordi det vil gøre det muligt at erstatte et lag med minimal indsats og uden bivirkninger i vores system. Denne arkitektur vil også gøre det let at vedligeholde systemet på grund af de lave koblinger mellem lagene, samt at det vil gøre det muligt at genbruge lagene. Vores system er delt op i disse 4 forskellige lag: *User Interface Lag, Control Lag, Model Lag* og *Database Transformation Lag.*



*User Interface Lag:*Dette lag er ansvarlig for håndtering af interaktion mellem aktøren og brugergrænsefladen

*Control Layer:*Dette lag er limlaget mellem User Interface lag og Model lag. Håndtering af use cases sker i dette lag

*ModelLayer:*Dette lag er afledt fra domæne model. Den indeholder klasser fra vores domæne model

*Database Transformation Lag:*Dette lag indeholder klasser, som sikre for håndtering af persistence af objekter i model laget. Den har ansvar for kommunikation med databasen, samt bygning af objekter i model laget.

*Database:*Databasen gemmer objekter der skal holdes persistent.

PS: En arkitektur kan enten være **åben** eller **lukket**. Vores arkitektur anvender en åben arkitektur.

*Åben arkitektur:* Er en arkitektur hvis specifikationer er offentlige, hvilket betyder at alle kan tilgå de forskellige klasser.   
En fordel ved åben arkitektur er at der kan genbruges kode fra forskellige klasser i steder for at skrive det samme igen.  
*Lukket arkitektur:* Er en arkitektur hvis specifikationer er private, hvilket betyder at de ikke kan tilgås af alle og enhver.

### Database Design

Formålet med dette afsnit er at mappe vores domæne model til et relationelt skema, ved hjælp af mapning regler. Vi begynder med at beskrive de mapnings regler vi har brugt, og vise vores databaseskema samt nogle integrity constraints vi har taget højde for. Herefter viser vi hvordan vi har mappede alle klasser i vores domain, samt håndtering af generalisering struktur i vores domain model og til sidst har vi normalisering.

#### Transformation af Domænemodel til database

Her beskriver vi, hvordan vi har omformuleret domæne modellen til relationelle model, ved hjælp af følgende regler

**Tabeller og Nøgler**

Hver klass i vores domæne model er blevet transformerede til tabeller. Klass navne er blevet til tabel navn, attribute navne er blevet til column navn i tabellerne. Vi har valgt de unikke kolonner som primær en nøgle. ex: mfCage, mfEmployee...

**Associering og Aggregering**

1-1 - Opret to tabeller og inkluderede ene sides primærnøgle som fjerne nøgle på den anden tabel.ex: mfCage - mfTransfer

1..n - Opret to tabeller, den ene sides primærnøgle bliver brugt som fremmednøgle på n sides tabel. ex: mfFarm - mfEmployee

**Generalisering**

VI har oprettet Opret to tabeller for de to specialiseringsklasser med deres attributer. Hver tabel også indholder generaliseringsklass attributer. ex:

mfBiteDisease og mfPlasmacytosis.

**Databaseskema**

Et databaseskema bruges til at vise logisk design af en database. Dette databaseskema viser struktur af vores database, den viser alle vores tabeller og relationer mellem tabellerne. De primære nøgler er understreget.



*Figur 8 Database Skema*

**De Integrity constraints vi har taget højde for:**

* *NOT NULL constraint* hjælper til at sikre at et felt altid indeholder en værdi. Det for eksempel betyder, at man ikke kan indsætte en ny employee i mfEmployee tabel uden at tilføj en værdi til navn feltet.
* *Referential integrity constraint* gennemføres ved bruge af en kombination af en primær nøgle og en fremmed nøgle. For eksempel, skal man angive en gyldig cageNo for at tilføje et nyt række til mfTransfer tabel. Det er for at sikre at hver felt i en tabel, der er erklæret en fremmed nøgle, kun indeholde værdier fra forældrenes tabels primære nøglefelt.
* *Primære key constraint* hjælper til at entydigt identificerer hver række i en database tabel. Eksempel er primære nøgle cageNo mfCage tabel, som kan kun indeholde unikke værdier.
* *Foreign key constraint* tillader et nøgle fra en tabel (fremmed nøgle) til at pege på en primær nøgle i en anden tabel. Eksempel er fremmed nøgle cageNo i mfTransfer tabel som peger på cageNo som er en primær nøgle i mfCage tabellen.

Vi har oprettet alle de tilsvarende databasetabeller i vores database. Vi vil få adgang til data i tabellerne ved at bruge SQL, som er både en DML (Data Manipulation Language) og en DDL (Data Definition Language). Følgende supplerende dokumenter er blevet tilføjet som bilag**: Database scripts** til at oprette de **Tabeller** i databasen og **Data Dictionary** som indeholder navnene og typer af alle felt i tabellerne.

**Mapping af Generalisering struktur i vores domain model**

*Problem:*

Vi har en generalisering og specialisering struktur i vores domæne model (det er mellem Disease, Plasmacytosis og Bite). Vi har transformeret dette struktur til tabeller ved at bruge et af de 3 mulige alternativer (de 3 alternativer har hver deres fordele og ulemper).

*Løsning:*

Vi har valgt at lave 2 tabeller (*mfBiteDisease* og *mfPlasmacytosisDisease*).

Den primære grund for at vælge at bruge dette alternativ er, fordi de minker kan kun have en af de to typer af sygdomme ad gangen, som kan enten være plasmacytosis eller bidsår. Vi har derfor valgt, at hver specialiseringsklasse (bidesår og plasmacytosis) skal afbildes i en tabel, som også indeholder generaliseringsklassens(Disease) attributter.

Men dette betyder ikke, at denne metode ikke har nogle ulempe. En ulempe er, at det kræver rettelser i begge specialiseringsklasser når der sker ændringer i generaliseringsklassen, men det er noget vi ikke forventer at det vil forekomme tit.

**Normalization**

Formålet med normalisering er at dekomponere relationer (tabeller) med anomalier. Det hjælper os til at producere strukturerede relationer, der indeholder mindre eller ingen redundans. Dvs. den primær hjælper os til at:

* minimere lagerplads
* reducere redundans
* minimere insertion, deletion og update anomalies

**Functional dependency** bruges til at analysere design kvalitet af en relation(tabel). Anvendes af functional dependency regler kan hjælpe til at transformer en tabel til en tilstand som kaldes **Normal Form**. Normalformer sikre, at anomalier, redundans og inkonsistens er reduceret i en database.

En tabel kan være i en af følgende normale former: 1NF, 2NF, 3NF, ... eller BCNF. Vi anvender kun BCNF i dette projekt. Det er sådan at hvis en tabel er i BCNF, så er det også i 3NF, 2NF og dermed også i 1NF.

**BCNF i mfEmployee tabel**

Ideen bag BCNF er, at hvis en tabel ikke er BCNF kompatibel, så skal de ikke-funktionelt afhængige attributter bruges til at danne nye separat tabeller. Vi vil vise hvordan vi har normaliseret mfEmployee tabel, ved at bruge BCNF. I øjeblikket er mfEmployee tabel ikke i BCNF.

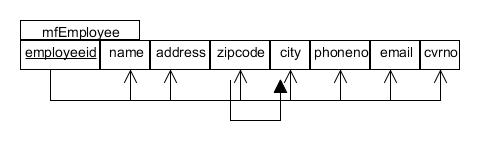
*mfEmployee (employeeid, name, address, zipcode, phoneNo, city, email)*

Kolonnen employeeid er en candidate nøgle, fordi den giver mulighed for unikt at afgøre værdien af ​​de andre kolonner i mfEmployee tabel. Men, zipcode kolonne kan også afgøre værdier i city kolonne.

Selv om zipcode kolonne kan også afgøre værdier i city kolonne, kan det ikke bruges som en candidate nøgle fordi den kun kan afgøre værdier af ​​city attributter og ikke alle de andre attributter.

*Problem:*

Vi kan konkludere, at mfEmployee tabel ikke er i BCNF, fordi city kolonne er funktionelt afhængig af både employeeid kolonne og zipcode kolonne - som ikke er en kandidat nøgle.

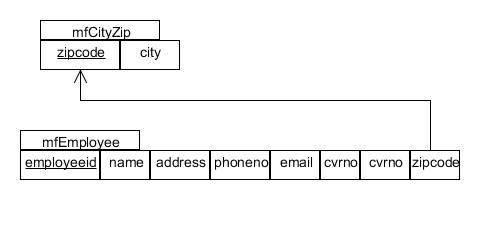


På grund af tidsmangel har vi ikke haft tid at lave vores mfEmployee tabel BCNF kompatibel og valgt at beholde attributterne ”zipcode og city” i samme tabel

*Løsning:*

For at dekomponere denne tabel til BCNF, skal mfEmployee tabel derfor opdeles i to tabeller, hvilket resulterer i to nye skema, der ser sådan ud:

* mfEmployee (employeeid, name, address, phoneNo, email)
* mfcityZip (zipcode, city)



Den ny tabel mfCityZip bruger zipcode som primær nøgle.

Tabellen mfEmployee bruger fremmed nøgle zipCode til at pege på mfCityZips primær nøgle.

### Design Model

#### GRASP PATTERNS

**Creator**

Det omfatter at vælge en creator, som har ansvar til oprettelse af objekter, der kan anvendes i andre dele af programmet.

**Controller**

Dette er de første klasser uden for brugergrænseflade som håndterer events fra UI og redeleger dem til andre lavere niveau klasser.

**Information Expert**

Information Expert anbefaler til at overdrage ansvar til de klasser som har de nødvendige oplysninger til at opfylde opgaven.

**Low Coupling**

Indebærer at vi skal designe klasser med lav kobling, så at de ikke er i høj grad afhænge af hinanden. Overdreven betyder, at hvis en klasse ændres, kan det let påvirke andre klasser, der er afhængige af denne klasse.

**High Cohesion**

Dette betyder, at elementerne i en klasse er stærkt relateret til hinanden. Hver klasse i systemet er designet til at være fokuseret på specifikke ansvar.

### Den mest kritiske use-case

#### System Sekvens Diagram: Transfer Diseased Mink

|  |
| --- |
| Følgende SSD er udarbejdet på baggrund af vores fully dressed use-case: Transfer Diseased Mink |
|  |
| Her ser vi så vores SSD og med den kan man se inden systemet overhovedet begynder at tænke skal en person der arbejder på farmen indtaste nogen data disse data er hvilket bur er skal gøres noget ved og først derefter retunere systemet informationer om det specifikke bur. Det samme er gældende med både sygdommen, medarbejderen og selve flytningen af burets indhold. |

#### Interaktions Diagram: Transfer Diseased Mink

|  |
| --- |
| Følgende Interactions Diagram er lavet ud fra ovenstående System Sekvens Diagram. |
|  |
| Dette er vores interaktionsdiagram og på dette kan man se at selve programmet starter i GUI og dernæst bevæger sig ned til CTRTransfer og herfra kan gå ud til de forskellige andre CTR. Derefter bevæger man sig ned til databasen som fortæller at flytningen sker, til modellaget hvorefter flytningen i databasen bliver udført. |

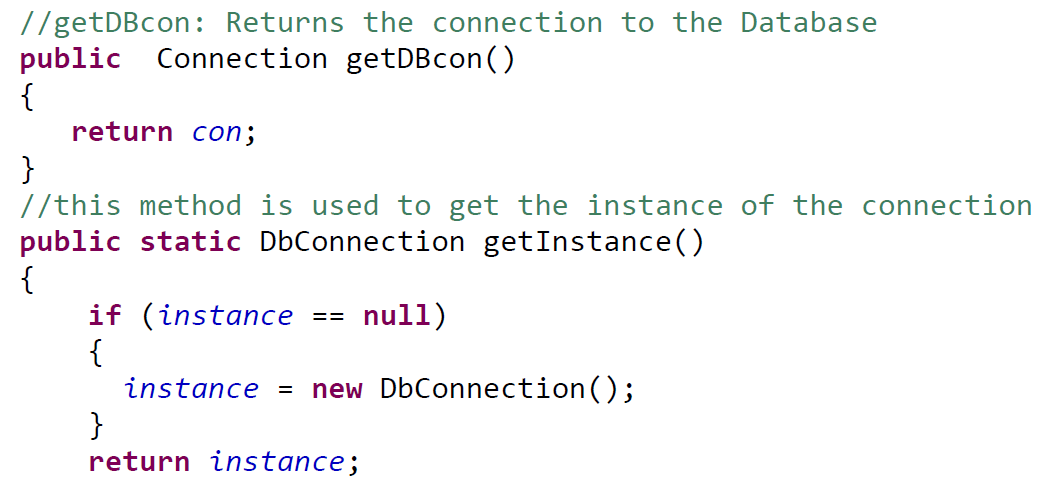
#### Design Klasse Diagram

|  |
| --- |
|  |
| Her kan vi så se vores design klasse diagram for vores use-case Transfer Diseased Mink. I dette kan man se vores program inden det blev til et program. Design klasse diagrammet viser alle de klasser der skal til for at vores program virker i disse klasser kan man også se alle attributterne og hvordan klasserne arbejder sammen. |

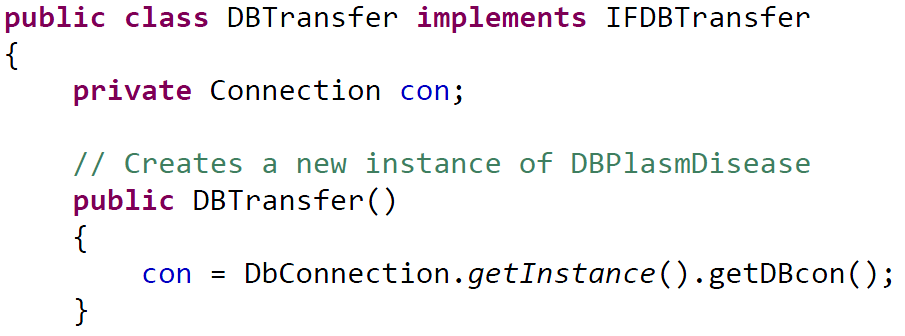
# System Konstruktion

## Singleton Mønstre

Vi har brugt singleton‐mønstre i DBConnection class for at sikre at der kun eksisterer en forbindelse til databasen. Alle klasse som kræver forbindelse til databasen skal bruge denne singletons instans.



Eksempel på, hvordan vi anvendt DBConnection class i DBTransfer konstruktor :



## Implementering

I design fasen valgte vi en åben arkitektur. Her beskriver kort de forskellige værktøjer, som vi har brugt til at implementere systemet.

**SVN**

Vi har brugt svn gennem hele implementation af systemet. Det har givet os mulighed for at gemme alle projekt relaterede filer i et centralt sted, hvor alle gruppe medlemmer kan få adgang til, arbejde på og opdatere eksisterende filer.

**Eclipse**

Hele programmeringen blev lavet ved at anvende Eclipse. Eclipse blev anvendt med svn, og dette har gjort det muligt for gruppens medlemmer til at være i stand til samtidigt at arbejde på koderne.

**Java**

Java blev anvendt som programmeringssprog, fordi den er baseret på OOP(object oriented programming) og det er operativsystem uafhængigt. Java er også det sprog for programmerings kursus for dette semester.

**Microsoft SQL Server/SQL**

Microsoft SQL Server er en relationel database management system, det giver os mulighed for at oprette, gemme og hente data fra vores database, ved hjælp af SQL som både er en DDL(data definition language) og en DML (Data Manipulation Language).

## Programmet

Systemet består af fire pakker:

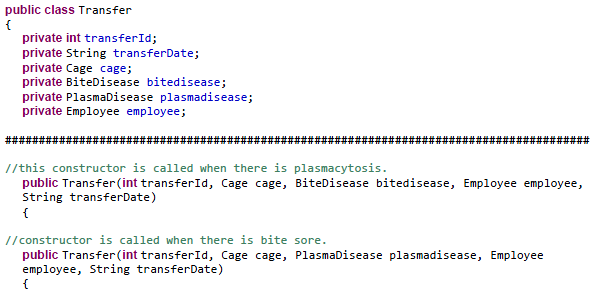
* package GUILayer
* package ControlLayer
* package DBLayer
* package ModelLayer

GUIlayer pakke interagerer med kontrol og model lag pakker. Kontrol lag pakke interagerer med dblayer og model lag pakker. Disse pakker er implementeringen af de fire lag i vores design klassediagram.

Herunder er noget kode udsnit fra Transfer klassen i de forskellige pakker.

**Model Layer:**

Alle variabler i klasserne fra dette lag er defineret som privat og kan kun tilgås ved hjælp af set og get metoder. Klasserne også indeholder forskellige former for en konstruktør, der anvendes til at instantiere objekter af hver klasse.



*Figur 9 Koden ovenfor viser vores Transfer klasse definition i model laget.*

**GUI layer:**

Klasser i dette lag er ansvarlig for oprettelse de views, der er synlig for brugerne af systemet, såsom screen layout og knapper.

|  |
| --- |
| *Figur 10 Her har brugeren indtastet Bur nummeret, Sygdoms ID’et og Medarbejderens ID i GUI.* |

**Control Layer:**

Alle metoder i DB laget, afspejles i kontrol lag. Gennem kontrol lag klasser, får GUI laget adgang til DB laget klasser og modellag klasser.

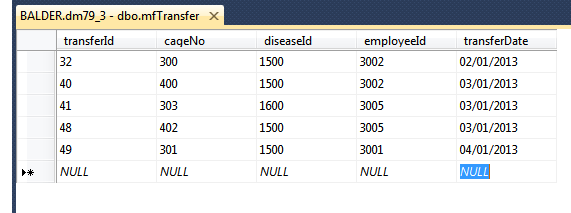
Den indrammede kode viser, hvordan at der bliver skelnet mellem de forskellige sygdomme, enten for den ID’et 1500 eller 1600 alt afhængig af hvilken sygdom det er. Dette bliver gjort så der er styr på hvilken database der skal vælges.

*Figur 11 Ovenstående kode viser kode fra kontrol lag, bruges til at oprette en Transfer objekt i kontrol laget.*

**DBlayer Layer:**

Klasser i dette lag sikre håndtering af persistence af objekter i vores database.

|  |
| --- |
| *Figur 12 Ovenstående kode fra DB laget og bruges til at indsætte en Transfer objekt i databasen.* |



*Figur 13 Ovenstående billede viser vores SQL database, hvor der vises at den tidligere indtastede transfer i GUI'en er blevet tilføjet til databasen*

#### Test

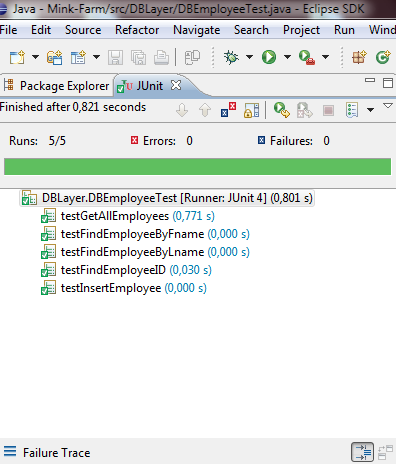
Dette afsnit beskriver, hvordan og hvad vi har testet systemet. Vi har udført 2 typer af tests -Unit test og Usability test.

##### Usability Test:

Målet for usability test var at finde ud af, hvor let det er for en bruger at udføre nogle opgaver, første gang de støder systemet. Unit tests blev udført ved brug af JUNIT imens de enkelte dele af systemet blev udviklet.

##### JUNIT Tests:

For at teste på de forskellige klasser, har vi oprettet JUNIT test cases i Eclipse. Vi har brugt Junit testing til at verificere funktionaliteterne af nogle klasser.



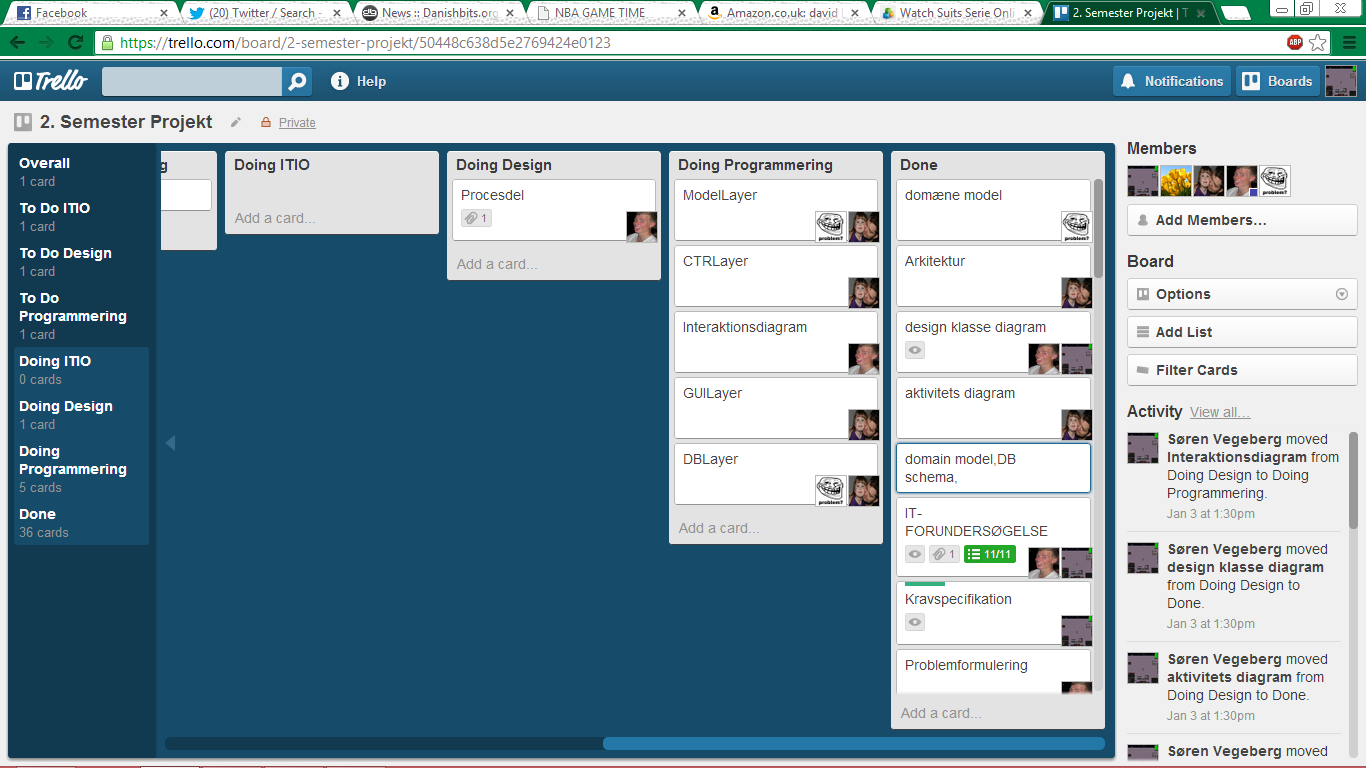
*Figur 14 Her vises et skærmbilleder af resultater af JUnit test case for DBEmployee class. Komplet test koder med deres resultater kan findes på medfølgende CD-Rom*

# Perspektivering

Vi har igennem dette 2. semester arbejdet med Mink Farmen hvor vi har været i kontakt med Kjeld V. Jensen, som er minkfarmer og Kopenhagen Fur, som står for videresalg af minkpelsene.

Vi har optil flere gange prøvet at få et grupperum, hvilket ville have gjort det lettere at koncentrere sig, dette har dog ikke været muligt da – skolen har været under ombygning og der har været meget pladsmangel, så de fleste lokaler har været i brug.

Vi har lige fra starten lavet en tidsplan i Microsoft Projekt, den har dog været mangelfuld, men den er efterhånden blevet opdateret så den har passet til vores projekt, så den indeholder de forskellige opgaver, i de forskellige fag såsom ITIO og System Design, som skulle laves. Vi har også valgt at tilføje antal dage på, så vi ca. har vidst hvornår en opgave skulle/ville være færdig, dog har vi måtte ændre dette op til flere gange.

Som hjælp til at uddele opgaverne har vi valgt at bruge hjemmesiden Trello. Trello er en hjemmeside hvor man kan sætte arbejdsopgaver i gang og sætte de personer der er blevet aftalt skulle lave dette. Det gode ved Trello er at man kan sætte labels på hver ar-bejdsopgave på den måde kan man holde styr på hvad der er blevet lavet hvad der skal laves og hvad der er i gang. Når man er færdig med alle de dele der skal laves for at projektet er færdig kan man få en tidslinje på Trello der fortæller hvem der har gjort hvad og på hvad tid det er gjort på.

*Figur 15 Udklip fra vores Trello board.*

# Konklusion

De overordnede mål med dette projekt var at lære at bruge viden fra ITIO til at analysere, design og implementere et it-system til en mink farmer.

Den første fase af projektet var at analysere og definere de vigtigste krav. Baseret på kravet, har vi udviklet et design, og vi har en delvis implementation af systemet.

Baseret på vores implamentation har vi et it-system med følgende funktioner:

* Muligt at registrere oplysninger om syg mink som er blevet flyttet
* Muligt at registrerer en ny farm med bure, haller, koloner samt mink.
* Det er muligt at styre medarbejder og forskellige sygdom information i farmen

I forbindelse med udviklingen af et delvis system, har gruppen fået kendskab til anvendelsen af UP procesmodel i projektudvikling.

Desuden har vi fået et praktisk kendskab til de udfordringer, der er forbundet med håndtering af tids krav under projektudvikling.

## Underskrift

|  |  |
| --- | --- |
| Faith Oziofu Nielsen: |  |
| Høgni Juul: |  |
| Kaj Viderø Olsen: |  |
| Søren Sand Vegeberg: |  |
| Jakob Lindholm Kaspersen: |  |

# Bilag

## Samarbejdsaftale

**Projektgruppe 3**

|  |
| --- |
| Høgni Juul, Faith Ogini Nielsen, Kaj Viderø Olsen, Søren Sand Vegeberg og Jakob Lindholm Kaspersen |

Gruppen indgår en samarbejdsaftale, som har til formål at fastlægge nogle forskellige rammer for gruppearbejdet.

Denne samarbejdssamtale er lavet med henblik på vores samarbejde under det afsluttende projekt til 2 semester.

Projektet kommer til at indeholde stof fra fagene System Design (SD), Virksomheden (ITIO), Software Konstruktion (SK), og Computer Architecture Operating Systems(CAOS).   
Derfor forventes at hvert gruppemedlem har læst og prøvet at sætte sig ind i de forskellige grundlæggende elementer i faget – som kommer til udtryk i dette tema.

Vi arbejder på klassen fra mødetidspunktet 8.30 til det på skrevne tidspunkt på skemaet – dog kan der opkomme situationer, hvor en person bliver nød til at møde senere eller gå tidligere, dette tager vi som det kommer. Der kan muligvis komme hjemmearbejde hvis det valgte arbejde ikke er blevet færdigt på den pågældende dag, hvor det skulle have været færdiggjort.

Det forventes her at alle i gruppen gør en indsats og at vi alle arbejder professionelt og holder alt internet på et ”low” eks Facebook, fodbold og youtube.

Vi vil stræbe efter at gemme alle vores filer i Dropbox og SVN hvilket gør vores filer tilgængelige til alle i gruppen.   
Vi vil også gøre brug af programmer som Microsoft Projekt, hvilket er et program, som bl.a. hjælper med holder styr på et projekts tidsplan – der ud over vil vi også gøre brug af hjemmesiden Trello – som hjælper med tidsplanlægning og uddelegering.

|  |  |
| --- | --- |
| Navn: Høgni Juul  Tlf: 53501204  E-mail: [hognijuul@gmail.com](mailto:hognijuul@gmail.com)  Skype: hognijuul | Navn: Jakob L Kaspersen Tlf: 22850582 E-mail: [kirsfc@hotmail.com](mailto:kirsfc@hotmail.com) Skype: kirs9000 |
| Navn: Søren Sand Vegeberg  Tlf: 26402498  E-mail: [mail@vegeberg.net](mailto:mail@vegeberg.net)  Skype: vegeberg174 | Navn: Faith Ogini Nielsen  Tlf: 60899038  E-mail: [faiteo2002@yahoo.com](mailto:faiteo2002@yahoo.com)  Skype: faithnielsen |
| Navn: Kaj Viderø Olsen  Tlf: 61994853  E-mail: kajnet@gmail.com  Skype: hellbreed |  |

## Tidsplan

## Regnskab

|  |
| --- |
|  |
|  |

## Data Dictionary

**mfFarm**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Beskrivelser | Key |
| Cvrno | int | Unique number to recognize each farm | Primary key |
| Name | Varchar(20) | Name of the farm | - |
| Address | Varchar(30) | The address of the farm | - |
| Zipcode | Int | The zipcode of the farms location | - |
| City | Varchar(20) | The city where the farm is located | - |
| phoneno | Varchar(50) | The phonenumber of the farm | - |
| email | Varchar(50) | The email address of the employee | - |

**mfHall**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Beskrivelser | Key |
| hallNo | int | Unique number to recognize each column | Primary key |
| farmCVR | int | Unique number to recognize Farm | Foreign key |

**mfColumn**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Beskrivelser | Key |
| colNo | int | Unique number to recognize each column | Primary key |
| hallNr | int | Hall inside a farm | Foreign key |

**mfCage**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Beskrivelser | Key |
| cageNo | int | Unique number to recognize each cage | Primary key |
| columnNo | int | Column inside a hall | - |
| cageType | varchar | The fabric of the cage | - |

**mfMink**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Beskrivelser | Key |
| color | Varchar(20) | The color of each individual mink | - |
| furLength | int | The fur length of each individual mink | - |
| furDensity | Int | The density of each individual mink | - |
| Birthdate | Varchar(50) | Date of birth | - |
| qualityType | varChar(20) | The quality of each individual mink | - |
| cageNr | Int | The cage where the mink is located | Foreign key |

**mfEmployee**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Beskrivelser | Key |
| Employeeid | int | Unique number to recognize each Employee | Primary key |
| fname | Varchar(40) | First name of the employee | - |
| Lname | Varchar(40) | Last name of the employee | - |
| address | Varchar(30) | The address of the employee | - |
| Zipcode | Int | The zipcode of the employee location |  |
| phoneno | Int | The phonenumber of the employee | - |
| email | Varchar(50) | The email address of the employee | - |
| cvrNo | Int | The CVR nr of the farm | Foreign key |
| City | Varchar(30) | The city where the farm is located | - |

**mfBiteDisease**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Beskrivelser | Key |
| diseaseId | int | Unique number to recognize each disease | Primary key |
| diseaseName | Varchar(30) | The name of the diesease | - |

**mfDiseaseRapport**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Beskrivelser | Key |
| cageNummer | int | Unique number to recognize each cage | Primary key |
| diseaseId | Int | Unique number to recognize each disease | - |
| reportDate | Varchar(40) | Date on report date | - |

**mfPlasmaDisease**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Beskrivelser | Key |
| diseaseId | int | Unique number to recognize each disease | Primary key |
| clinicName | Varchar(40) | Name of the clinic | - |
| clinicCVR | Varchar(40) | Unique number to recognize each clinic | - |
| Diseasename | Varchar(40) | Name of disease | - |

**mfTransfer**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Beskrivelser | Key |
| trasferId | int | Unique number to recognize each trasfer | Primary key |
| cageno | int | Unique number to recognize each cage number | - |
| diseaseId | int | Unique number to recognize each disease | - |
| employeeId | Int | Unique number to recognize each employee | - |
| transferDate | Varchar(40) | The date of the transfer | - |

1. <http://www.kopenhagenfur.com/da/om-os/dansk-pelsdyravlerforening-kopenhagen-fur/> [↑](#footnote-ref-1)
2. <http://www.kopenhagenfur.com/da/ansvarlighed/dyrevelfaerd/> [↑](#footnote-ref-2)
3. <http://www.regionsyddanmark.dk/wm306145> [↑](#footnote-ref-3)
4. <http://dinbror.dk/blog/tag/unified-process/> [↑](#footnote-ref-4)