|  |
| --- |
| Førsteårsprøven |
| Ferrari Financing System |
| ? |

|  |
| --- |
| Sofie Blom Jensen, Shahnaz Yahyavi, Martin Tang Olesen  Vejledere: Flemming Koch Jensen, Hans Iversen  Hold: he18dmu-2s17 - Datamatikeruddannelsen  Udannelsesinstitution: Erhvervsakademi Midtvest  01-06-2018  Antal tegn: ? |

Indhold

[Indledning 1](#_Toc513449179)

[Problemstilling 1](#_Toc513449180)

[MUST (Shahnaz) 1](#_Toc513449181)

[Project management 1](#_Toc513449182)

[Udviklingsmiljø 1](#_Toc513449183)

[Iterations- og faseplan (Sofie) 1](#_Toc513449184)

[Fase 1: Inception 3](#_Toc513449185)

[Iteration I1 3](#_Toc513449186)

[Visionsdokument (Sofie) (review Martin) 3](#_Toc513449187)

[Domænemodel (Sofie, review Martin) 4](#_Toc513449188)

[Use case diagram (Sofie, review Martin) 6](#_Toc513449189)

[Use cases (Martin, review Sofie) 7](#_Toc513449190)

[Systemsekvensdiagrammer(Shahnaz) 9](#_Toc513449191)

[Operationskontrakter 9](#_Toc513449192)

[Klassediagram – trelagsarkitektur (Shahnaz) 9](#_Toc513449193)

[Fase 2: Elaboration 11](#_Toc513449194)

[Iteration E1 11](#_Toc513449195)

[Dataordbog(Shahnaz) 11](#_Toc513449196)

[Data Model(Shahnaz) 11](#_Toc513449197)

[Normalisering(Shahnaz) 11](#_Toc513449198)

[Konklusion 14](#_Toc513449199)

[Litteraturliste 15](#_Toc513449200)

[Bilag 16](#_Toc513449201)

[Bilag 1 – Iterations- og faseplan 16](#_Toc513449202)

[Bilag 2 – Visionsdokumentet 17](#_Toc513449203)

[Visionen 17](#_Toc513449204)

[Interessentanalyse 17](#_Toc513449205)

[Feature-liste 17](#_Toc513449206)

[Bilag 3 – Use case diagram, første udkast 18](#_Toc513449207)

# Indledning

## Problemstilling

# MUST (Shahnaz)

Vi hat brugt MUST til at identificere mulige risici for vores system og finde nogle løsninger til dem. De risici kunne være mange ting som manglende viden eller forket forståelse til det færdige produkt.

Derudover MUST kan brugt til at lave en IT-forundersøgelse af Ferrari bil forhandler til at giver dem, det bedste produkt, og undgås misforståelser. Vi skulle forstå hvad forventninger til systemet og hvad koster systemet til sidste, fordi det skal være acceptable til vores kunder.

Vi har også vælger at arbejder med hinanden fordi vi alle sammen præcise og vi har arbejder sammen før.

Vi har lavet en projektplan også sammen, fordi vi synes at den er rigtig godt hjælpemiddel til at give os et øjeblik over hver skal arbejde med hver dag, og beskriver netop hvordan projekten skal forløbe.

Vi planlægger også hvordan og hvornår skal mødes hver dag.

# Project management

## Udviklingsmiljø

## Iterations- og faseplan (Sofie)

Som en del af project management har vi udarbejdet en iterations- og faseplan. Denne er med til at sikre, at vi holder et overblik over de kommende iterationer samt de milepæle der skal nås for hver fase. Den udarbejdes i starten af inception fasen, da den givetvis er et grundlæggende led i project management. Den er dog på dette tidspunkt endnu ikke helt stabil, da vi har behov for at opdatere den løbende. Vores plan indeholder desuden en detaljeret opremsning af de aktiviteter, vi gennemfører den enkelte dag i hver iteration. For hver dag har vi valgt at starte med at se på vores plan for dagen, og vi afslutter desuden dagen med at opdatere planen, samt at føre en log over hvad vi har nået. Det hjælper os til at overholde planen samt reflektere over vores arbejdsproces. Iterations- og faseplanen bør være stabil inden vi går i construction-fasen.

# Fase 1: Inception

## Iteration I1

### Visionsdokument (Sofie) (review Martin)

Vi har valgt at begynde med at udarbejde visionsdokumentet[[1]](#footnote-1) som en af de første artefakter i den første iteration. Visionsdokumentet er fordelagtigt at arbejde med i starten af projektet, da man får indskrænket problemdomænet og formålet med projektet uden dog at begrænse løsningsmulighederne for meget. Visionsdokumentet består af en beskrivelse af visionen, en interessentanalyse og en feature-liste.

Visionen er en kortfattet fremtidsbeskrivelse, der formuleres på en måde der ikke specificerer teknologivalget, med mindre kunden har nogen helt specifikke krav hertil. Den sammenligner ikke det nye system med noget tidligere, men har fokus på fremtiden og løsningen af den nuværende problemstilling. I vores tilfælde er der altså fokus på, at systemet skal samle alle skridt der tages i forbindelse med afgivelse af lånetilbud, når Ferrari skal sælge en bil med finansiering. Visionsdokumentet tager også højde for, at Ferrari er interesseret i et intuitivt brugerinterface uden forsinkelser, en effektivisering af virksomhedens nuværende proces og en minimering af tab af salg, hvilket de har oplevet som en konsekvens af den nuværende arbejdsgang. Visionen indeholder også dele af vores it-forundersøgelse, da vi har fået en case udleveret er vi nødsaget til at bruge elementerne fra case. Normalt vil mange af disse punkter have dukket op igennem netop denne analyse

Interessentanalysen er en analyse af alle der har interesse i løsningen, vi når frem til. Der kan være tale om brugere, kunder, lovgivere, ejere af understøttende systemer og så videre. Med andre ord: interessenter der anvender eller betaler for produktet, tilbyder services til os som udviklere eller på anden måde regulerer problemdomænet. Interessentanalysen tager højde for sådanne interessenters konkrete ønsker, krav og behov. Vores interessentanalyse tager således udgangspunkt i forhandlerens kunder, dem der er interesseret i at købe en bil, bilsælgerne, kontorassistenter og økonomimedarbejdere samt salgschefen og den bank, forretningen giver finansieringstilbud i samarbejde med. Vi henviser til Bilag 2 for en mere detaljeret gennemgang af interessentanalysen og de enkelte interessenters interesser og behov.

Feature-listen er et element i visionsdokumentet, der skaber overblik over systemets funktioner. Det er således en liste over alt hvad systemet ud fra problemdomænet skal kunne udføre overordnet set, og er med til at danne grundlag for den videre udviklingsproces. Vores feature-liste indeholder de funktioner, der er behov for i forbindelse med afgivelse af lånetilbud hos bilforhandleren. Vi har valgt at medtage kreditværdighedstjek hos RKI, indhentning af aktuel rentesats hos banken, beregning af rentesats ud fra givne oplysninger fra banken og RKI, registrering af et lånetilbud til en kunde samt eksport af et lånetilbud med en tilbagebetalingsplan.



Figur 1 - Domænemodelnotation

### 

### Domænemodel (Sofie, review Martin)

I den første iteration har vi valgt at udarbejde et første udkast til domænemodellen. Domænemodellen er en måde at visualisere og analysere problemdomænet på et forholdsvis tidligt stadie i projektet. Modellen viser relationerne mellem forskellige koncepter i problemdomænet, og vi udarbejder den ud fra problemformuleringen og de use case beskrivelser vi har på nuværende tidspunkt. Vi tegner domænemodellen så den ligner UML-klassediagrammer, dog med en del variationer i notation. Kort sagt er notationen opbygget således at kasserne[[2]](#footnote-2), der i et klassediagram ville repræsentere klasser nu snarere repræsenterer koncepter, der ikke nødvendigvis skal afspejles som klasser i det færdige system. Pilene viser association mellem de forskellige koncepter, samt om der er tale om multipliceret. Vores færdige domænemodel ses i *Figur 2*. I vores domænemodel har vi valgt at medtage Salgschef, Bank, RKI, Kunde, Bilsælger og Lånetilbud som koncepter i problemdomænet. Her er det sælgeren der kommer til at interagere med systemet som primær aktør, derfor har han associationer til alle andre koncepter i problemdomænet.

INDSÆT DOM HER

Figur 2 - Domænemodel

Sælgerens relation til både banken og RKI består i at han indhenter oplysninger fra dem, der er nødvendige for at udarbejde det endelige lånetilbud. Derfor viser modellen også, at der kan være flere sælgere, men der er altid kun tale om én bank og ét RKI register. Sælgerens relation til salgschefen er, at han arbejder under ham. Ud fra problemformuleringen ved vi, at der kun er én salgschef. Bilsælgerens relation til det enkelte lånetilbud er, at det er ham der opretter og afgiver det. Han kan naturligvis oprette mange lånetilbud, da de vil være unikke for den enkelte kunde. Associationspilen mellem kunden og sælgeren går begge veje, da de har kontakt med hinanden og begge kan tage initiativ til denne kontakt. En sælger kan traditionelt sagtens have kontakt med flere forskellige kunder. Kunden kan også have kontakt med flere sælgere, men vi forstår ud fra problemformuleringen, at der er behov for en kontrol af at den enkelte kunde kun modtager ét lånetilbud, ligegyldigt hvor mange sælgere han har været i kontakt med i forbindelse med købet.

((TILFØJ HVIS DER SKAL STÅ NOGET MERE OM SALGSCHEFEN))

### Use case diagram (Sofie, review Martin)









**Kommunikation**

****

**System boundary**

Figur 3 – Use case-diagramnotation

Use case-diagrammet giver overblik over funktionelle krav, hvorfor den er fordelagtig at begynde på i den første iteration. Den bør være stabil inden overgang til construction-fasen, da det er en del af milepælen der skal nås indenfor elaboration. På nuværende tidspunkt behøver den dog ikke være stabil eller færdiggjort, da vi ikke har identificeret alle use cases endnu. Den anvendes blot som et værktøj til os som udviklere til at holde overblik over de use cases vi har identificeres på nuværende tidspunkt. Use case-diagrammet viser den primære aktør, der interagerer med systemet, og de understøttende aktører, der bidrager til opfyldelse af målet med hver af de identificerede use cases. En mere detaljeret beskrivelse af de enkelte use cases og deres formål findes under afsnittet ”Use cases”.

Identifikation af use cases er en del af Elementary Business Process (EBP), hvilket repræsenterer en virksomheds aktiviteter på det mest elementære niveau. En use case udgør således som udgangspunkt en EBP, idet det kan defineres som én opgave, der udføres af én bestemt person i forbindelse med én form for forretningshændelse. Denne opgave vil føre til en værdi og nogen data som er målbare for virksomheden. En sådan EBP består så af flere trin, der tilsammen vil skabe denne værdi. Det kan derfor være svært at identificere use cases, da der kan opstå tvivl om hvilket niveau de bliver defineret ud fra. For eksempel: er ”opret låneaftale” en passende use case? Eller består den af for mange trin? Eller: ”Start Programmet”, vil det være en passende use case, eller består den af for små skridt. Man bør altså have fokus på hvor mange skridt en use case vil indeholde, og hvilken værdi den vil skabe. Men som sagt giver use case-diagrammet blot et overblik over de identificerede use cases, og afspejler således ikke disse detaljer[[3]](#footnote-3). De er blot nødvendige i forbindelse med identifikationen af use cases.

Vores use case diagram er forholdsvis begrænset i omfang, og vi har kun på nuværende tidspunkt få primære aktører at fokusere på, nemlig sælgeren og salgschefen, og derfor har vi valgt kun at lave ét diagram. Hvis der var mange aktører med mange use cases til hver, ville det være fordelagtigt at dele diagrammet op, således at der var ét diagram pr. aktør. I vores tilfælde giver det dog mest mening at beholde de identificerede use cases i ét diagram, som vi herefter kan opdatere og udvide efterhånden som processen skrider frem i de kommende iterationer. Notationen vi har benyttet i forbindelse med udarbejdelse af use case-diagrammet ses i *figur 3*. i Bilag 3 ses vores første udkast til et use case-diagram. I udarbejdelsen af use case diagrammet har vi desuden haft fokus på at de forskellige use cases og aktører alle sammen eksisterer i domænemodellen. Det er endnu en måde at sikre sig, at de use cases vi identificerer, er passende og forbliver indenfor projektets scope. Som det ses, er alle aktørerne, salgschef, sælger, bank og RKI repræsenteret som koncepter i domænemodellen. Desuden passer alle use case-titlerne til forholdet mellem de forskellige koncepter i domænemodellen; for eksempel er ord som ”kreditvurdering” og ”rentesats” kendte både i domænemodellen og use case-diagrammet. Det er desuden værd at bemærke, at alle use cases i det første use case-diagram er repræsenteret som Concrete use cases, idet de startes af en aktør og realiserer aktørens ønskede mål med casen[[4]](#footnote-4). Vores use case diagram vil blive nærmere beskrevet i elaboration afsnittet, da det som sagt er i denne fase den skal være stabil.

### Use cases (Martin, review Sofie)

Use cases spiller en stor rolle i UP, det er dem der driver projektet fremad og det er ud fra vores use cases vi vælger det næste skridt. Ved hjælp af use case-diagrammet fik vi hurtigt udvalgt de første use cases vi vil tackle. En god hovedregel i UP er at tackle den use case med størst risici involveret først, netop fordi et af de primære formål med inception fasen er at afdække om vi overhovedet kan lave dette system. Vi udvalgte hurtigt 3 use cases, som vi i fællesskab var helt enige om var kernen i dette system, og dermed her de største risici ligger. Den store use case som vi har valgt at kalde ”FFS-UC3 - Udregn lånetilbud”[[5]](#footnote-5), indeholder alle de tre trusler vi har identificeret som værende de store; vi har derfor valgt at tackle de 3 trusler som værende hver deres use case. Den første use case, kaldet ”FFS-UC1 - Tjek kreditværdighed”[[6]](#footnote-6), blev så til en underfunktion af UC3. Det samme gør sig gældende for ”FFS-UC2 - Indhent aktuel rentesats”[[7]](#footnote-7). Vi har valgt at beskrive disse use cases formelt; vi kunne også have brugt et uformelt use case format, men vi mener, at eftersom dette er kernen af systemet, vil det hjælpe fremadreddet, at de er beskrevet på det formelle niveau. Det formelle format er netop godt, fordi de funktionelle krav bliver beskrevet på en formel og utvetydig måde. Målet er at få beskrevet hovedscenariet og alle variationer af dette.

Det er vigtigt at vi har et **unikt id** til alle vores use cases, da vi laver mange andre artefakter ud fra disse, og det er vigtigt at holde sporbarhed hele vejen tilbage til den konkrete use case. **Titlen**, der står efter vores unikke id, prøver vi så vidt muligt at holde som en beskrivende kommando, der indfanger use casens formål - den holdes på bydeform. **Afgrænsningen** af use casen fortæller hvad der berøres af use casen, i mindre systemer som dette vil afgrænsningen meget naturligt være systemet selv; dette er også tilfældet for alle vores use cases. **Niveauet** indikerer use casens placering i hierarkiet, den typiske og mest brugte her er "brugermål", men vores UC1 og UC2 er underfunktioner til UC3 og har derfor fået niveauet ”underfunktion” i stedet. Den **primære aktør** beskriver den person, der interagerer med systemet i den konkrete use case; som nævnt i afsnittet ovenfor bestræber man sig på at det kun er én bruger pr. use case. Det er vigtigt at bemærke, at vi forsøger at beskrive rollen på aktøren så præcist som muligt, dvs. betegnelsen ”bilsælger”, som er vores aktør, er væsentligt bedre end blot at kalde ham en ”bruger”.

**Interesser og interessenter** beskrives også i de enkelte use cases. Her trækker vi en rød tråd tilbage til vores visionsdokument og udvælger de interessenter, som har interesse i den konkrete use case. **Forudsætninger** dækker over krav der skal være opfyldt før use casens udførelse kan gå i gang. Det kan være andre use cases som skal være afviklet først, som vi ser i vores UC3. Det er vigtigt at man ikke nævner trivielle forudsætninger, da det ikke har nogen værdi for os som udviklere – eksempelvis ”computeren er tændt” eller ”brugeren er logget ind” hvis de ting siger sig selv. **Succesgaranti** dækker over de forhold der skal være opfyldte ved use casens afslutning. De gælder primært for hovedscenariet, men alternative scenariers succesgarantier kan også beskrives. disse succesgarantier bør opfylde alle vores interessenters interesser. For vores use cases var det ret simpelt og ligefrem at beskrive vores succesgarantier, da de bare er opfyldt når den funktion er fuldendt korrekt.

**Hovedscenariet** beskriver det oftest forekommende scenarie. Det beskrives uden variation og skal som hovedregel opfylde alle interessenternes interesser. På grund af størrelsen af vores UC1 og UC2 er der ikke mange trin i disse, men hovedscenarier kan ofte blive på mange trin i use cases der er mere omfangsrige. **Varianter** er hvor man beskriver alternative scenarier, der kan give en variation fra hovedscenariet - både successcenarier men også fejlhåndteringsscenarier. Disse varianter består af 2 trin. Første trin er betingelsen, her beskrives betingelsen samt hvordan den opdages. Trin 2 er håndteringen. Her beskrives de trin, der skal tages i denne variation, hvilket kan lede til en fortsættelse til hovedscenariet eller specifikt vælge at afslutte use casen. De primære varianter, vi har brugt i vores UC1 og UC2, er fejlhåndteringer. **Teknologier og dataformer** dækker over eventuelle krav der kan være stillet fra kunden. Vi bør som hovedregel prøve at begrænse disse mest muligt for ikke at ligge os fast på teknologivalg så tidligt i processen. Krav her kan i værste tilfælde blive en trussel for os senere, som vi så skal håndtere og dermed gøre systemet dyrere for kunden. I vores tilfælde er vi blevet stillet visse krav til brug af visse API'er, derfor er disse medtaget i vores UC1 og UC2. **Ikke-funktionelle krav**, som hører til use casen, eller som afdækkes mens use casen udføres, tilføjes her; de bør dog også blive tilføjet til den supplerende kravspecifikation senere. Et af de ikke funktionelle krav, som går igen i både UC1 og UC2, er kravet om at vores kald til RKI og bank ikke må påvirke brugbarheden af vores brugergrænseflade, mens de foretages. **Hyppighed** beskriver hvor tit en use case forekommer. I vores tilfælde er det ikke noget, der reelt vil ske hurtigt efter hinanden - vi har dog skrevet ofte fordi vi mener vores system skal kunne understøtte at man laver disse kald inden for kort tid (for eksempel mange gange på én dag). Det sidste punkt på den formelle use case er **diverse**, hvor man beskriver det, som ikke passer ind under de andre punkter. Vi kan også tilføje kendte eller eventuelt uløste problemstillinger, man har opdaget undervejs. Vi har ikke haft noget til diverse posten i nogle af vores use cases og den er derfor ikke taget med på den formelle beskrivelse af use casen.

### Systemsekvensdiagrammer(Shahnaz)

Systemsekvensdiagrammer viser at hvordan en bruger kommunikation med systemet i forhold til en konkrete usecase scenarier, den også fokuserer på hvad systemets respons er, ved en bestemte stimuli. I SDD kigger vi på systemet ligesom en blak box, og vi viser alle de funktioner som kommer fra en usecase på systemet. SDD er rigtig godt til at analysere systemet og viser enlige hvad systemet gøre. Den er også meget godt at viser rollerne(aktør) i systemet. Vi bruger SSD til at identificerer system operationer.

### Operationskontrakter

### Klassediagram – trelagsarkitektur (Shahnaz)

Trelags-arkitektur er en model til beskrivelse af de blokke et program er opbygget af. Modellen anvendes blandt andet fordi det gør det nemmere at overskue kommunikationen mellem lagene og kommunikationen med brugerene og med andre IT-systemer. Vi bruger trelagsarkitektur til at styrer på vores system og vi sætter klasser i klassediagram på plads til at vi kan selv se hver opgaven til enkelte klasse i vores system.

Udover det når men bruger interfaces og trelagsarkitektur til at gøre vores system mere brugervenlig til andre udvikler i fremtid.

Vores trelagsarkitektur er lagene defineret på følgende måde:

* Veiwleg: Det øverste lag der håndterer modtagelse og præsentation af data. Dette lag er kendetegnet ved at være ”tæt” på brugeren af programmet.
* Logiklag: Det midterste lag der håndterer udvekslingen af data mellem præsentationslaget og datalaget.
* Datalag: Det nederste lag der opbevarer og håndterer data. Dette lag er også kendetegnet ved at være ”tæt” på computeren.

Et billede, der indeholder tekst, tavle

Beskrivelse, der er oprettet med meget høj sikkerhed

trelagsarkitektur som vi selv tegner den

# Fase 2: Elaboration

## Iteration E1

### Dataordbog(Shahnaz)

Data ordbog beskriver alle væsentlige termer og forkortelser i problemdomænet. Den er også giver eksempler på dem. På den måde systemudvikler kan forstå forretning sprog og forstå bedre hvad henleder systemet om. Den beskriver en general beskrivelse om en koncept og beskriver også en konkrete eksempler på den.

### Data Model(Shahnaz)

Data modellering er en proces, der bruges til at definere og analysere datakrav, der er nødvendige for at understøtte forretningsprocesserne inden for rammerne af tilsvarende informationssystemer i organisationer. Processen med datamodellering involverer derfor professionelle datamodellerer, der arbejder tæt sammen med forretningsmæssige interessenter, samt potentielle brugere af informationssystemet.

Data modellering teknikker og metoder bruges til at model data på en standard, konsistent, forudsigelig måde for at styre det som en ressource. Brugen af datamodelleringsstandarder anbefales kraftigt for alle projekter, der kræver et standardmiddel til at definere og analysere data i en organisation.

Data modellering kan udføres under forskellige typer projekter og i flere faser af projekter. Datamodellerne er progressive; Der er ikke noget som den endelige datamodel for en virksomhed eller en applikation. I stedet skal en datamodel betragtes som et levende dokument, som vil ændre sig som reaktion på en skiftende forretning. Datamodellerne skal ideelt set opbevares i et lager, så de kan hentes, udvides og redigeres over tid.

Dette er en del af etableringen af en informationssystemstrategi, der definerer en overordnet vision og arkitektur for informationssystemer. Informationsteknologi er en metode, der omfatter denne tilgang.

I systemanalyse oprettes logiske datamodeller som led i udviklingen af nye databaser.

### Normalisering(Shahnaz)

Normalisering af en database, er en teknik som sikrer at rettelser i databasen, kan foretages med mindst muligt indflydelse på det oprindelige system. Målet er at minimere redundant data. Det vil sige at samme oplysning er gemt flere steder. Med normalisering bliver det lettere at foretage rettelser i databasen (så skal man kun rette det ét sted).

lidt en balancegang i forhold til performance, men det skal være velovervejede designbeslutninger og ikke dovenskab der skal være styrende for om man vælger at gå på kompromis med normalformerne. Normalisering er godt i teorien, og er absolut en god tommelfingerregel.

Problemerne med redundans kan deles op i to grundargumenter:

* **Performance**: Dataudtræk kan blive langsomme, da den redundante data medfører at der bliver brugt unødvendige ressourcer (IO, memory, netværk og cpu).
* **Maintainability**: Den samme opdatering skal foretages mange steder.

**Vores tabeller til FFSDB:**

**Låntilbud:** låntilbudid, Dato, FørsteBeløb, afdrag, kundenavn, KundeAddresse, Kundetlf, KundCPR, Kundebedømmelse, renteset, bilnavn, bilpris, bilsælgernavn, chef, creditRating

**1.NF:**

1NF er nok til at oprette en database, men i lavere normal form øges risikoen for uregelmæssigheder, når data opdateres.

**I vores database**

I lånetilbud tabel vi har data redundans i den løsning vi laver ikke flere tabel, men bare laver flere rækker til den. den skaber dog mere redundans.

**2.NF**

1. Tabellen skal opfylde alle krav for 1. normalform.
2. Hvis en tabel har en sammensat nøgle, skal alle felter, der ikke indgår i nøglen, afhænge af den samlede nøgle.
3. Alle non-prime attributter skal være fuldt funktionelt afhængige af primærnøglen. (dvs. ingen partiel funktionel afhængighed).

Hvis der kun findes en enkelt primær nøgle, er tabellen allerede i 2. normalform, hvis der derimod er en sammensat primærnøgle, kan den stadig bringes til 2. normalform, ved at splitte tabellen op i separate tabeller.

**I vores database**

Alle attributter skal være fuld afhængig PK som den er låntilbudid her, så vil laver flere taber.

**Låntilbud**: låntilbudid, Dato, FørsteBeløb, afdrag, Kundetlf, RKI, Renteset, Bilid, bilsælgerid

**Kunde**: navn, Adresse, tlf., CPR., creditRating

**Bil**: Bilid, pris, navn

**Bilsælger**: id, navn, chef

**3.NF**

1. Tabellen skal opfylde alle krav for 2. normalform.
2. Der må ikke findes felter uden for primærnøglen, som er indbyrdes afhængige.
3. Ingen transitive funktionelle afhængigheder mellem non-prime attributter.

**I vores database**

Vil har ikke det her problem.

# Konklusion

# Litteraturliste

# Bilag

## Bilag 1 – Iterations- og faseplan

## Bilag 2 – Visionsdokumentet

### Visionen

Systemet samler alle skridt der tages i forbindelse med afgivelse af lånetilbud ved køb af virksomhedens produkter(Ferrari). Systemet skal have et intuitivt brugerinterface som reagerer uden forsinkelse. Systemet bidrager til virksomhedens drift ved at effektivisere processen ved afgivelse af lånetilbud. Det skal kunne minimere tab af salg grundet bortkomne formularer, og kunne tilgås fra alle steder, af hensyn til salgschefens forretningsrejser.

### Interessentanalyse

Regional Ferrari-forhandler: Interesseret i at øge virksomhedens salg. Interesseret i muligheden for at godkende låneplaner uden at være fysisk tilstede i virksomheden.

Kunde: Kunden har interesse i at få en hurtig og korrekt udmelding om lån kan godkendes.

Bilsælger: Bilsælgeren har interesse i at vide om kunden allerede er registreret, for at undgå at der gives tilbud til samme kunde flere gange. Han har interesse i at processen bliver nemmere og hurtigere, da det gør det muligt at fokusere på kundekontakt og salg.

Kontorassistent: Kontorassistentens arbejdsbyrde lettes ved implementering af systemet.

Økonomimedarbejder: Økonomimedarbejderen har interesse i at processen bliver effektiviseret, da det gør hans arbejde lettere og hurtigere at udføre.

Salgschef: Salgschefen er interesseret i at øge virksomhedens salg ved en mere effektiv proces.

Banken: Banken er interesseret i virksomhedens finansielle situation, der forventes forbedret ved implementering af systemet. Banken øger også sit eget salg, når virksomhedens salg forøges.

### Feature-liste

Effektiv kommunikation med brugeren gennem en grafisk brugergrænseflade

Kreditværdighedstjek hos RKI

Indhentning af aktuel rentesats hos banken

Beregning af rentesats ud fra givne oplysninger

Registrering af et lånetilbud til en kunde

Eksport af et lånetilbud med en tilbagebetalingsplan

## Bilag 3 – Use case diagram, første udkast





1. Se Bilag 2 - Visionsdokumentet [↑](#footnote-ref-1)
2. Se illustration 1 - domænemodelnotation [↑](#footnote-ref-2)
3. Se afsnittet Use cases for flere detaljer. [↑](#footnote-ref-3)
4. I modsætning til Abstract use cases, der ses som en underfunktion til andre use cases og ikke kan stå alene. Dette vil dog blive revideret senere. [↑](#footnote-ref-4)
5. Se bilag ????????????? [↑](#footnote-ref-5)
6. Se bilag ?????????????? [↑](#footnote-ref-6)
7. Se bilag ????????????? [↑](#footnote-ref-7)