Opgaver

Minimumskrav til dokumentation

Et separat dokument – gerne som pdf-dokument, der indeholder:

- Navn på udvikler, dato og et navn på projektet
- UML diagram inkl. use cases + E/R diagram
- En kort teknisk beskrivelse af projektet med begrundelser for de anvendte teknikker.
- Logbog over udviklingen af projektet
    - Ændringer
    - Problemer/Udfordringer
    - Husk: Det er bedre med dokumentering af problemer end ingenting.

Dokumentationen må gerne være mere uddybende end beskrevet – men det skal ikke være en roman.

Hvis I anvender flow charts til at analysere opgaven, må de meget gerne inkluderes i dokumentationen.

Læs HELE opgaven godt igennem og stil spørgsmål så tidligt som muligt (hvis der er noget, I er i tvivl om).

Denne opgave er individuel, men I må hjælpe hinanden eller spørge om hjælp.

Husk at dette fag er på avanceret niveau.

Database opgave: ”Min” case

Du har arbejdet med et database case projekt på H1, der nu skal udbygges. Det er selvfølgelig helt OK at ændre på den oprindelige struktur (forhåbentlig ser du andre muligheder efter gennemgang af nye temaer).

Alle principper fra første del forventes anvendt i denne opgave.

Din case fra H1 (biludlejning, dyreklinik eller egen virksomhed/spil) skal udbygges som beskrevet i nedestående eksempler fra dyreklinikkens databasen: (tilsvarende eksempler kan lave for biludlejning, egen virksomhed/spil)

1. Dyreklinik

- *Dyrlægen har brug for at løse flg. problemstillinger:
    1. Slå behandlingshistorik op (jf. udskrift 1) evt. i et datointerval.
    2. Fakturerer behandlinger til kunder (jf. udskrift 2).
    3. Slå statistik op for en given behandlingstype og i et givent datointerval, samt se omsætning.
- Du opfinder selv priser for behandlinger, men dyrlægen skal senere kunne ændre dem, uden det får indflydelse på historikken for tidligere behandlinger og prisen på det tidspunkt.
- Der SKAL udarbejdes USE CASES, E-R diagram, og det SKAL overholde 3. normalform (normalisering af en database). Afleveres som en samlet pdf eller billede.
- Du skal implementerer Stored Procedures til at udføre dine USE CASES. Dine Stored Procedures skal indeholde:
    - Parametre og variabler
    - Kontrolstruktur (Logik: if, else if, else)
    - Afleveres i SQL som tekst
    - Returnere værdi om resultatet er OK eller ej
- Skal indeholde mindst en Trigger funktion – En påmindelse, automatisk opdatering eller noget tilsvarende.
- Indeksering skal anvendes, så der opnås optimal performance.
- Der skal tænkes sikkerhed ind i løsningen
    - Bruger adgang
    - Undgå utilsigtede SQL Injektion
- Skal indeholde avanceret SQL kommandoer (f.eks. aggregatet, single row og/eller Group funktions).

*Kunne også være et af følgende eksempler

2. Biludlejningen

- Ekspedienten:
    1. Slå historikken op for en bestemt bil
    2. Faktura til kunder
    3. Slå statistik op for et bestemt bilmærke/type inden for en bestemt periode for at se omsætningen.
- Sæt selv priserne for leje af bilerne, forsikring, brændstof mv., men ekspedienten skal senere kunne ændre dem, uden det får indflydelse på historikken for tidligere udlejninger og prisen på det tidspunkt.

3. Virksomhed "Salg af dimser"

- Bogholder:
    1. Slå historikken op for en bestemt vare
    2. Fakturer til kunder
    3. Slå statistik op for en gruppe af vare inden for en bestemt periode og se omsætningen og/eller statistik for en gruppe sælger for at se, hvor meget har de hver især har solgt for.
- Sæt selv priserne for de forskellige vare, men bogholderen skal senere kunne ændre dem, uden det får indflydelse på historikken for tidligere køb af varer og prisen på det tidspunkt.

4. Spil

- Spiller/Spil udbyder:
    1. Se hvilke features han/hun har til rådighed
    2. Fakturer til spiller
    3. Slå statistik op for spil resultater/køb
    4. Login spiller/spil udbyder
- Sæt selv priserne for de forskellige features, men spil udbyderen skal senere kunne ændre dem, uden det får indflydelse på historikken for tidligere køb og prisen på det tidspunkt.

Eller noget helt fjerde…

Husk igen at databasedelen skal minimum være på et avanceret niveau.

Bilag til database opgaven: (Dyreklinik)

Udskrift 1

Rapport over behandlingshistorik

Patient ID	Navn	Type	Alder	Ejer navn	dato	Behandling procedure
246	Fiddo	HUND	12	Peter kvist	JAN 13/2018	01 - Rabies vaccination
					MAR 27/2016	10 - Undersøgelse og behandling for skade
					APR 02/2015	05 - Hjertekardiogram test
298	Plet	HUND	2	Kirtsen F. Hansen	JUN 21/2016	08 - Hvalpe vaccination 2/3
					JAN 10/2016	07 - Hvalpe vaccination 1/3
341	Misser	KAT	4	Torben Jensen	AUG 19/2015	23 - Kastration (Kat)
					JAN 23/2014	02 - Vaccination - 8 uger
519	Pipper	FUGL	2	Kirsten F. Hansen	JUN 21/2016	20 - Årligt tjek
					JUN 21/2016	12 - Skylning af øjne

Udskrift 2

FAKTURA

Andersens dyrehospital
Jens Juelsvej
5000 Odense C

DATO: JUN 21/2016

Fru. Kirsten F. Hansen
Smedevænget 12
5230 Odense M

Patient	BEHANDLING	BELØB
Plet	Hvalpe vaccination 2/3	1.238,00
Pipper	Årligt tjek	500,00
	Skylning af øjne	345,00
	I alt:	2.083,00
	Moms (DK: 25%)	520,75
	TOTAL	2.603,75

Evt. en fælles database og OOP opgave:

Der tages udgangspunkt i case opgaven for databasen fra de forgående sider.

Altså en case for en dyreklinik, biludlejning eller egen virksomhed/spil.

- Læs og udfør alle punkterne fra før, men denne gang skal løsningen præsenteres i en **GUI** til databasen baseret på OOP.
- Det er vigtigt, at den bagvedliggende funktionalitet er fuldt objektorienteret og overholder præmisserne for OOP.
- Løsningen skal indeholde et **Class Library** (CL), som du kan anvende i et nyt **WPF** (**W**indows **P**resentation **F**oundation) eller i et **WinForms** projekt med reference til dit **CL**.
  WPF eller WinForms projektet udgør din GUI og dit Class Library udgør din Business Logik.
- Det er vigtigt, at du tænker over dit OOP design, herunder SKAL du gøre brug af Interface og/eller Abstrakt klasser.
- Du skal også anvende en eller flere af de følgende metoder: Overskrivning, abstrakte metoder og/eller polymorfi.

Hvis der er tid nok:

- Implementering af et asynkront forløb
- Lambda udtryk og/eller eventhåndtering i form af delegates

Husk

- At dokumentere opgaven undervejs (log bog). Skriv kort, men dagligt.
- Versionsstyring er ikke et krav men en god ide også at udføre dagligt.
- OOP kan være på rutineret, avanceret eller ekspert niveau, men databasen skal som minimum være en **avanceret** løsning.

I dette forløb er der to muligheder:

1. Aflevering af et separat OOP- og database projekt (2 afleveringer)
2. Aflevering af en samlet opgave, der indeholder både OOP og database i sammen projekt (afleveringen skal løfte målene for begge fag)

Uanset hvilken opgave/opgaver I vælger at løse, skal alt dokumenteres som beskrevet.

Vurdering af opgave:

	Niveauet, som opgaven løses på	*Personlige kompetence, der knytter sig til niveauet.*
Rutineret	Rutinemæssig eller kendt situation. • Alene og i samarbejde. • Planlægge og gennemføre en opgave eller aktivitet. • Løse et problem. • Selvstændigt sætte sig ind i mere komplicerede problemstillinger.	• Viser fleksibilitet og omstillingsevne.
Avanceret	Ikke-rutinesituationer. • Alene eller i samarbejde. • Vurdere et problem. • Planlægge, løse og gennemføre en opgave eller aktivitet. • Løse et problem.	• Tager selvstændigt ansvar. • Viser initiativ. • Lægger vægt på kvalitetssans og kreativitet.
Ekspert	Bruge allerede opnåede kompetencer. • Løse komplekse arbejdsopgaver. • Argumentere for valgte løsning og opståede problemer. • Bruge opnåede kompetencer i en ny kontekst. • Arbejde med overblik og deltagelse i arbejdspladsens innovative processer.	• Planlægge, tilrettelægge, udføre og evaluere arbejdsprocesser. • Vurdere og begrunde behovet for at forbedre arbejdsprocesser. • Kommunikere om sin faglighed i alle relevante sammenhæng.