Programmering og udvikling af små systemer samt databaser

EKSAMENSOPGAVE EFTERÅR 2021

Forord

Denne opgavebeskrivelse er udarbejdet af Nicolai Jacobsen og Regitze Sdun til brug ved eksamen i faget Programmering og udvikling af små systemer for foråret 2022.

Eksamensprojektet skal udarbejdes i grupper af 2-4 personer.

I må gerne diskutere og arbejde sammen med andre grupper. Husk dog, at såfremt I benytter kode fra hinanden, eller tutorials skal dette angives med en kildehenvisning.

I skal igennem arbejdet med jeres eksamensprojekt vise de kompetencer frem, som I har opnået igennem kurset. Vi forventer derfor, at I kan leve op til alle fagets læringsmål.

Eksamensopgaven er en udvidelse af eksamensopgaven fra efteråret, hvor I nu skal inkorporere det, som I har lært om objektorienteret programmering og databaser i jeres løsning af det oprindelige problem. Det betyder at de tidligere krav også skal revurderes så de opfylder et objektorienteret paradigme.

I vil kunne finde inspiration og løsninger til store dele af jeres eksamensprojekt igennem de uge- og godkendelsesopgaver, som I allerede har udarbejdet.

Spørgsmål omkring formalia til opgaven bedes lagt på Canvas, så alle kan få glæde af svaret. Såfremt spørgsmålet omhandler personlige forhold, som for eksempel sygdomsperioder, skal disse sendes til studieadministrationen.

Opgavebeskrivelse

I har startet jeres egen virksomhed, hvor I gerne vil lave en konkurrent til Den Blå Avis (DBA). I er trætte af de dårlige søgeegenskaber og det kedelige user interface, så det vil I gøre noget ved. Jeres MVP blev en success og I har fået en god investering på plads, I har derfor ansat 1-3 nye udviklere på jeres projekt og er nu en gruppe på 2-4.

I skal derfor arbejde videre med jeres web-applikation, men med udvidet funktionalitet. Disse kan ses under "funktionelle krav", oven i dette skal de foregående krav også opdateres så de er objektorienteret, så I lettere kan udvide applikationen i fremtiden. Dertil skal dataen også gemmes i en normaliseret database, så data let kan tilgås.

I skal derfor udvikle app'en i Node.JS som server og Express til endpoints. App'en skal have en frontend, som brugerne kan navigere igennem. Samt et datalag som gemmer al information om brugere og annoncer. Det betyder, at I skal udvikle tre dele:

- Klienten (JS, HTML, CSS)
- Server/API (Node.JS & Express)
- Storage i T-SQL hvori jeres brugere og annoncer kan bo.

Kravene for disse enkelte dele fremgår under Tekniske krav til besvarelsen.

Funktionelle krav til besvarelsen

Tidligere krav fra efteråret (skal opdateres til at fungere med jeres nye database):

1. App'en skal tillade en bruger at oprette en profil
2. App'en skal tillade en bruger at slette sin egen profil
3. App'en skal tillade en bruger at opdatere sin egen profil
4. App'en skal tillade brugeren at logge ind
5. App'en skal tillade at hvis en bruger er logget ind kan de forblive logget ind.
6. App'en skal gøre det muligt at en bruger kan oprette en vare med en varekategori, billede og pris.

7. App’en skal gøre det muligt at en bruger kan slette en vare du har oprettet.
8. App’en skal gøre det muligt at en bruger kan opdatere en vare du har oprettet.
9. App’en skal gøre det muligt for en bruger at logge ud
10. App’en skal kunne vise en bruger en tabel over de varer, som de har til salg.
11. App’en skal have en kategori menu hvor man kan klikke og få præsenteret alle varer til salg inden for en given kategori.
12. Lave udførlige unit tests til 1 krav (vælg mellem krav 1-9). Dette betyder at alt funktionalitet for kravet skal være unit testet.

Udvidede krav:

13. App’en skal give admin mulighed for at opdatere en brugerprofil
14. App’en skal give admin mulighed for at slette en bruger
15. App’en skal give en admin mulighed for at opgradere en bruger til “guld”. Guldbrugeren’s annoncer har første prioritet og bliver vist i toppen af listen fra krav 11 herefter vises annoncer fra normale brugere.
16. En admin skal have mulighed for at se brugsstatistikker af app'en herunder
    a. En tabel af antallet af annoncer pr bruger
    b. Det totale antal annoncer
17. App’en skal give en bruger mulighed for at “følge” annoncer. En bruger skal altså kunne klikke “følg” på en annonce og så skal disse annoncer kunne vises på brugerens profil. (Mens annoncerne stadig er aktive.)
18. App'en skal give brugerne mulighed for at filtrere deres søgning af annoncer. Dette filter ***skal*** virke på følgende egenskaber:
    a. Lokation
    b. Hvor lang tid annoncen har været oppe
    c. Pris
    d. Definer selv en ekstra parameter at filtrere på og beskriv, hvorfor det er en god ide.

Formelle krav til besvarelsen

1. Rapporten skal afleveres som pdf via Digital Eksamen
2. Alle sider i rapporten – inklusiv forsiden - skal være fortløbende nummereret
3. Antallet af sider skal fremgå af forsiden
4. Rapporten for Programmering og udv. af små systemer samt databaser må højst fylde 30 normalsider. Alt, hvad der ligger ud over disse sider, vil ikke blive taget i betragtning ved bedømmelsen. Se regler for optælling af sider på my.cbs.dk. Er den kortere er det også fint, I bliver bedømt på indhold, ikke længde
5. Antallet af normalsider dokumenteres ved f.eks. udskrift af rapport fra tekstbehandlingsprogrammets statistik-funktion
6. Figurer i rapporten skal være tydelige og læselige
7. Rapporten skal være gennemlæst og ikke indeholde stavefejl
8. GitLog skal vedhæftes afleveringen, som dokumentation for udviklingsprocessen. Vi forventer, at I løbende har arbejdet i forskellige branches og har foretaget løbende commits af jeres kildekode
9. Jeres kildekode skal vedhæftes afleveringen som en zip-fil. Sørg for at koden kan køres og at det er dokumenteret, hvordan man kører koden
10. Andre bilag såsom UML-diagrammer skal også vedhæftes i zip-fil'en

Tekniske krav til besvarelsen

I skal udarbejde tre dele til jeres system.

1. Klienten (JS, HTML, CSS)
2. Server/API (Node.JS & Express)
3. Storage (SQL server)

De tre dele skal være forbundet og trække data fra hinanden, hvilket vil give brugeren mulighed for at interagere med klienter, som sender en forespørgsel til serveren, der afslutningsvist henter eller opdaterer data i databasen.

1 - Klienten

I har mulighed for at lave en klient, som benytter et af de mange frontend-frameworks der findes til Node.JS, eller I kan lave jeres klient i helt simpelt HTML og CSS. Vi sætter ikke krav herom, men forventer, at I begrunder jeres valg og kommer med argumenter herfor.

Vi bruger ikke ressourcer på at evaluere jeres brugerinterface i forhold til brugervenlighed og det grafiske udseende, hvorfor I ikke bør bruge meget tid på denne del.

2 - Serveren

I skal udarbejde serveren i Express med Node.js. Arbejdet går på at udarbejde et API, som kan benyttes i forbindelse med jeres klient. API'ets hovedfunktion er at forbinde klienten og storage og dermed være mellemled (backend) imellem de to.

Applikationen udstiller et HTTP-interface, som kan give svar på HTTP-requests og sende svar tilbage. Dataudvekslingsformatet skal være JSON.

I skal lægge vægt på at opbygge et velgennemtænkt modellag, som danner grundlaget for jeres applikation. Herudover skal I følge normal god kodeskik og dokumentere jeres kode med kommentarer og en fornuftig filstruktur. Koden skal være nem at vedligeholde for andre efter aflevering, og koden skal naturligvis være versioneret ved brug af Git.

3 - Storage

For at opfylde kravene til projektet er det nødvendigt at gemme noget data. Dette data forventes gemt i en normaliseret T-SQL database sådan, at det er nemt at udvide med mere metadata senere - skulle dette være nødvendigt.

De valg I gør jer for storage skal dokumenteres i jeres rapport. Dette omfatter f.eks. et Entity-Relationsdiagram: Altså hvordan I har valgt at strukturere jeres data.

Materialer og gode råd

Som baggrund for at gennemføre opgaven anvendes denne opgavebeskrivelse samt fagets pensum. I kan derudover søge oplysninger i andre offentligt tilgængelige kilder (husk at angive kilde med APA-modellen).

Der opfordres til, at I benytter forløbets forskellige muligheder for at diskutere projektet og de problemstillinger I møder undervejs. Dertil opfordres I til at sparre med hinanden og tilbyde reviews af hinandens løsninger. Ofte ender man med at se sig blind i det man selv laver, hvorfor det kan være nyttigt med et sæt friske øjne.

I er velkommen til at poste tekniske problemer på Canvas, hvor vi gerne vil hjælpe. Vi forventer dog, at I også hjælper hinanden og byder ind i forhold til hinandens problemer.

Det er vigtigt, at I holder jer for øje, hvad dette fag fokuserer på og dermed undlader elementer, som for dette fag ikke er relevante.

I skal lægge vægt på refleksion og diskussion, hvilket bør vægte højere end beskrivelse. Såfremt I benytter teori er det derfor vigtigt, at I ikke blot beskriver teorien, men reflekterer over brug af teorien og dermed konkretiserer teorien til lige netop jeres projekt.

Rapporten

Nedenstående er minimumskravene listet til de enkelte del-leverancer:

Kravspecifikationen & arkitekturen er det endelige produkt. I afsnittet om tekniske krav, skal I lave en tabeloversigt over hvilke krav, der er opfyldt og hvilke der ikke er. Herefter skal I gå hvert krav systematisk igennem, én for én og beskrive hvordan de er opfyldt brug gerne visualisationer fra frontenden. Hertil

opfordres I til at angive løsningovervejelser for de tekniske krav I ikke har opfyldt.

Som bilag til rapporten skal I vedlægge en lille video af jeres applikation (som form af et link), som viser at alle krav I skriver er opfyldt, rent faktisk er opfyldt. Denne video skal uploades til vimeo eller youtube og kun linket lægges med i rapporten.

Klassediagrammer vil være den grundlæggende model for jeres applikation. Det er samtidig jeres model af problemområdet, som dermed er en analyse af kravene til det system I har udviklet. Vi forventer, at I argumentere for den valgte model og kritisk forholder jer til eventuelle forretningsmæssige begrænsninger heri. Klassediagrammet skal være lavet efter gældende regler for UML og følge formatet heri. Ud over et UML klassediagram kræver vi også et sequence diagram af 1 af de udvidede krav.

Tests er en vigtig del af softwareudvikling og derfor skal I også teste jeres program. Da tests er meget tidskrævende må I vælge et punkt fra den funktionelle krav specifikation, som I skal unitteste så grundigt som muligt. Dette skal dokumenteres i rapporten.
I skal kun dokumentere hvordan I unittester, beskrivelser af hvordan man debugger og andre typer tests gives der ikke point for.

Entity Relationship Diagram viser hvordan jeres database er designet. Diagrammet giver et hurtigt overblik over, hvordan I har valgt at gemme jeres data. Det forventes derfor, at I vedlægger et ER diagram i rapporten samt argumenterer for, hvorfor det ser ud, som det gør (Hint: det er oftest en god ide, at det er normaliseret til 3nf).

Løsningsovervejelser skal være en tungtvejende del af jeres opgave. Vi forventer at høre omkring de overvejelser I har gjort jer i forhold til strukturen for jeres respektive applikationer, samt hvilke begrænsninger som eventuelt måtte være.

Procesevalueringen har til formål, at I forklarer jeres læring i forløbet. I dette afsnit kan I for eksempel beskrive 3-5 hændelser, der er opstået undervejs i jeres projektforløb og hvordan I valgte at løse disse hændelser metodisk.

Rapporten skal først og fremmest beskrive løsningen, dvs. produktet og de dertilhørende designvalg, men ikke processen af at nå frem til produktet bortset fra procesevalueringsafsnittet.

Eksamen i Programmering og udv. af små systemer samt databaser

Eksamen udføres af fagets underviser med intern censur, og bedømmes efter 7-trins skalaen. Karakteren gives efter en samlet bedømmelse af rapporten, jeres kodebase og jeres mundtlige eksamination.

Bedømmelsen af rapporten vil lægge vægt på følgende:

- Disposition og rapport-systematik
- Jeres kodebase og konstruktionen heraf, samt dokumentation af jeres arbejdsproces ved brug af git
- Jeres forståelse for tekniske og forretningsmæssige begrænsninger af jeres implementering, samt refleksion over de valg I har truffet heraf
- Beskrivelse af jeres forudsætninger, valg af perspektiv(er) og redegørelse for jeres valg
- Begreber og definitioner (deres præcision, konsistens og relevans)
- Teori- og metodeanvendelse
- Tabeller og figurer (deres informationsværdi, konsistens og relevans)
- Anvendt litteratur (omfang og dybde)
- Kildekritik (både teori og data)
- Forklaringer: redegørelse for årsag og virkning
- Procesevalueringens kobling til pensum
- Bilag (deres kvalitet og relevans)
- Rapportens omfang (den skal holde de formelle retningslinjer)
- Sprog, kildehenvisninger og struktur
- Evt. afskrift / direkte reproduktion uden kildehenvisninger

Der gives karakter efter fagets læringsmål, som kan ses i kursuskataloget.