Programmering og udvikling af små systemer samt databaser

EKSAMENSOPGAVE 2020

Forord

Denne opgavebeskrivelse er udarbejdet af Henrik Thorn til brug ved eksamen i faget Programmering og udvikling af små systemer for efteråret 2020.

Eksamensprojektet skal udarbejdes individuelt, men I må gerne diskutere og arbejde sammen. 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 på nær et enkelt:

- designe og foretage simple forespørgsler og opdateringer i en database

Dette læringsmål vil vi gå i dybden med på næste semester. I må derfor gerne vælge database som storage allerede nu, men vi forventer først, at I kan arbejde med dette fra næste semester.

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

Du er netop blevet ansat i en startup, som vil skabe en konkurrent til Tinder. De har derfor bedt dig om, at designe og implementere en første udgave af en dating-app. Selve app'en skal virke meget lig Tinder, men man har forhåbninger om på sigt, at kunne forbedre selve matching-algoritmen og dermed skabe en konkurrencefordel.

Brugerne vil ikke skulle betale for brugen af App'en, men der vil på et senere tidspunkt blive lavet nye features, som potentielt vil kunne virke med micro-transactions.

Din projektleder har bedt dig lave en MVP (Minimum Viable Product), som kan bevise investorer om, at forretningsideen er værd at investere på i. Du skal derfor udvikle en app i Node.JS, som benytter MVC-frameworket som overordnet struktur. App'en skal være objektorienteret og skal have en frontend, som brugerne navigerer efter. Det betyder, at du skal udvikle tre dele:

- Storage
- API
- Frontend

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

Funktionel kravspecifikation:

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 gøre det muligt for en bruger at vælge like eller dislike for en foreslået profil
6. App'en skal give brugeren en notifikation, såfremt begge profiler har liket hinanden
7. App'en skal gøre det muligt for en bruger at logge ud
8. App'en skal kunne vise en liste over en aktuels brugers matches
9. App'en skal kunne vise en fuld profil for et potentielt match
10. App'en skal give brugeren mulighed for at fjerne et match igen
11. App'en skal give admin mulighed for at se liste over aktuelle brugere i systemet

12. App'en skal give admin mulighed for at opdatere en brugerprofil
13. App'en skal give admin mulighed for at slette en bruger

Matching Algoritme

Du skal udvikle en matching-algoritme, som viser de personer der indenfor en vis radius befinder sig på en person. Derudover skal du sortere de personer, som bedst passer til den aktuelle bruger. Dette skal du gøre ud fra personens interesser, så brugeren først får vist de andre brugere med flest matchende interesser. Det er din opgave, at designe algoritmen, så den tager flest mulige parametre med ind - men så den stadigvæk performer hurtigt og effektivt.

Formelle krav til besvarelsen

Rapporten afleveres via Digital Eksamen. Alle sider i rapporten – inklusiv forsiden - skal være fortløbende nummereret. Antallet af sider skal fremgå af forsiden.

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.

Antallet af normalsider dokumenteres ved f.eks. udskrift af rapport fra tekstbehandlingsprogrammets statistik-funktion.

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.

Jeres kildekode skal vedhæftes afleveringen som en zip-fil, som også indeholder eventuelle andre relevante bilag. Dette kunne for eksempel være nogle af jeres UML-diagrammer.

Tekniske krav til besvarelsen

I skal udarbejde tre dele til jeres system, som vil følge three-tier-model.

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

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 opdatere 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 bruger meget tid på denne del.

2 - Serveren

I skal udarbejde serveren i Node.JS med Express som middleware. Arbejdet går på at udarbejde et API, som kan benyttes i forbindelse med jeres klient. API'ets hovedfunktion er at forbinde klienten og databasen og dermed være mellemled (backend) imellem de to. Det betyder, at I skal opbygge en applikation som er baseret på MVC.

Applikationen udstiller et HTTP-interface, som kan give svar på HTTP-requests og sende svar tilbage. Dataudvekslingsformatet skal være JSON eller XML, men vi vil klart anbefale jer JSON.

I skal lægge vægt på at opbygge et velgennemtænkt model-lag, 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

I skal implementere storage for de oplysninger, som jeres API behandler. I kan vælge mellem følgende storage-muligheder, som også er gennemgået i undervisningen:

- Fil-storage
- Database
- IndexDB
- localStorage

I skal i jeres rapport argumentere for valget af jeres storage-type og herigennem forklare fordele og ulemper ved den valgte model.

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, at I giver sparring til de hinanden og tilbyder 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.

Vejledningen giver jer en mulighed for at få feedback på den endelige eksamensopgave, som I skal aflevere. Projektet løber over hele semesteret og for at gøre det så virkelighedsnært som muligt,

kan I forvente, at der kan komme tilføjelser til de endelige krav til opgavebesvarelsen i løbet af semesteret.

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. Et eksempel herpå er elementer fra projektledelse, som I med rette bør notere til dette fag i stedet.

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 reflektere over brug af teorien og dermed konkretisere teorien til lige netop jeres projekt.

Rapporten

I nærværende eksamensopgaven er der beskrevet forskellige aktiviteter, som det forventes, at jeres gruppe tager højde for i den endelige rapport. Nogle aktiviteter er specifikt forbundet med processen, hvor andre aktiviteter er forbundet med rapporten. Det kan være svært at adskille processen fra produktet, idet processen afleder produktet. Nedenstående er minimumskravene listet til de enkelte del-leverancer:

Kravspecifikationen & arkitekturen er det endelige produkt. I skal beskrive systemet og systemets funktioner samt hvilke tekniske krav, der er til systemet og ikke mindst, hvordan disse krav opfyldes. I kan med fordel benytte use-cases til denne beskrivelse. Såfremt I bruger use-cases skal disse overholde UML-notationen.

Klassediagrammet 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.

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. Derudover forventer vi, at I har gjort jer overvejelser omkring eventuelle algoritmer, som har særlig tyngde i jeres projekt.

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.

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 og jeres kodebase.

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 kursuskateloget.