Programmering og udvikling af små systemer samt databaser

GODKENDELSESOPGAVE 3

Forord

Denne opgavebeskrivelse er udarbejdet af Henrik Thorn til brug ved faget Programmering og udvikling af små systemer samt databaser på 1. semester af HA(it.) studiet. Opgaven er stillet som godkendelsesopgave 1.

Opgaven tæller som én aflevering ud af de ott godkendelsesopgaver i faget. Du skal have godkendt fem ud af otte opgaver for at kunne gå til eksamen i faget.

Målet for opgaven er at sætte dig i stand til at løse den opgave, som du vil blive mødt med til den fire timers skriftlige stedprøve. Du kan derfor se disse godkendelsesopgaver, som din mulighed for at følge dit eget niveau løbende igennem faget - samt modtage feedback på dit resultat.

Godkendelsesopgaver

Opgave 1 - Individuel opgave
Opgave 2 - Individuel online quiz
Opgave 3 - Individuel opgave
Opgave 4 - Individuel online quiz
Opgave 5 - Individuel opgave
Opgave 6 - Individuel opgave
Opgave 7 - Individuel opgave
Opgave 8 - Individuel online quiz

Du vil i denne opgave blive testet for følgende kompetencer, som følger den undervisning vi har været igennem på nuværende tidspunkt af kurset:

- Klasser og objekter
- MVC
- HTTP
- UML
- Events
- GIT

Deadline for aflevering er 18. oktober 2020.

Opgavebeskrivelse

Du skal i denne opgave designe og implementere et API, som kan benyttes til en dating-app. API'et skal følge MVC, men skal ikke indeholde en frontend.

- User
- PaymentUser
- FreeUser
- CreditCard
- Interest
- Match
- Image

Du skal ikke implementere en frontend for dit API, ligesom du ikke skal implementere storage. Du må derfor gerne hardcode de brugeroplysninger, som dit system indeholder - og det er denne information, som bliver opdateret, slettet mm.

Der findes to undertyper af User, som er hhv. PaymentUser og FreeUser. Førstnævnte har betalt for brugen af jeres service, hvorfor de har registeret et CreditCard. I skal lave nedarvning mellem User og de to undertyper.

Du skal implementere CRUD-endpoints for følgende entiteter:

- User
- Interest
- Match

Din rapport skal omhandle dine løsningsovervejelser, samt begrundelse for dine valg. Dette kræver, at du tænker forskellige alternativer igennem og dermed reflekterer over hvorfor en løsning er bedre end en anden. Du må gerne have kodeeksempler som en del af din rapport, hvilket kan bruges til at understøtte dine forklaringer og reflektioner. Hertil skal din rapport indeholde et klassediagram, som følger UML-notationen.

//LAV KLASSEBESKRIVELSER

Formelle krav til besvarelsen

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

Afleveringen for Programmering og udv. af små systemer samt databaser må højst fylde fem 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.

Jeres kildekode skal vedhæftes afleveringen, som en zip-fil der afleveres som et bilag til selve rapporten der afleveres som en PDF.

Rapporten skal være skrevet på dansk, imens I opfordres til at alt jeres kode skrives på engelsk.

Tekniske krav til besvarelsen

Du skal løbende lave commits af din kode i GIT og du skal have implementeret en master-branch og en udviklingsbranch (develop).

Dine endpoints skal understøtte JWT-tokens, hvorfor du skal implementere en login-funktion for brugeren. Du kan se mere herom her:

https://tutorialedge.net/nodejs/nodejs-jwt-authentication-tutorial/