**Programmering og udvikling af små systemer og databaser**

**Skriftligt produkt**

|  |
| --- |
| Laura Bøjer  Studienummer: 144984  HA(it.) - Copenhagen Business School |
| Afleveret d. 17. oktober 2020  8.963 anslag og 5 sider |

#### Indholdsfortegnelse:

[**Indledning** 2](#_Toc57818945)

[**Analyse** 2](#_Toc57818946)

[**Perspektivering** 2](#_Toc57818947)

[**Konklusion** 2](#_Toc57818948)

[**Litteraturliste** 2](#_Toc57818949)

#### Indledning

#### Analyse

#### Perspektivering

#### Konklusion

#### 

#### Litteraturliste

Læringsmål, som skal opfyldes:

1. Redegøre for og konstruere udvalgte delprodukter, der fremstilles i et objektorienteret systemudviklingsforløb
2. Redegøre for grundlæggende begreber indenfor programmering
3. Anvende et objektorienteret programmeringssprog til at konstruere mindre applikationer
4. Designe og implementere et simpelt brugerinterface
5. Anvende de teknikker og værktøjer, der er introduceret i faget, til at analysere og løse et komplekst problem gennem udvikling af et mindre administrativt program (analyse, design, konstruktion, test og dokumentation)
6. Reflektere over valgte implementeringer, samt optimeringer heraf

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

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 tillade at hvis en bruger er logget ind kan de

forblive logget ind.

6. App’en skal gøre det muligt for en bruger at vælge like eller

dislike for en foreslået profil

7.App’en skal give brugeren en notifikation, såfremt begge

profiler har liket hinanden

8. App’en skal gøre det muligt for en bruger at logge ud

9. App’en skal kunne vise en liste over en aktuels brugers matches

10. App’en skal kunne vise en fuld profil for et potentielt match

11. App’en skal give brugeren mulighed for at fjerne et match igen

Formelle krav:

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. Er den kortere er det også fint, I bliver bedømt på indhold, ikke længde.

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:

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 (Simpel fil storage & local storage (local storage er

koblet op med frontenden))

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.

De valg I gør jer for storage skal dokumenteres i jeres rapport. Dette omfatter f.eks. den valgte json struktur - Altså hvordan i har valgt at strukturere jeres data.

De valg I gør jer for storage skal dokumenteres i jeres rapport. Dette omfatter f.eks. den valgte json struktur - Altså hvordan i har valgt at strukturere jeres data.

- Store datasæt + sensitive data = ikke den bedste løsning.

https://www.youtube.com/watch?v=6iZiqQZBQJY

sync vs. no sync (readFile/writeFile)

- venter på at alle data er loadet

get request kan behandle små dele data, men kører gennem URL'en, hvorfor det ikke er optimalt at logge ind/oprette bruger via get request

Post sker "usynligt", behind the scenes og er derfor mere optimal

testet med postman, da browsere kun kan modtage get requests

post()

LocalStorage:

https://www.youtube.com/watch?v=NxVCq4p0Kb0 (video omkring at gemme objekter i local storage)

Do not store sensitive user information in localStorage

It is not a substitute for a server based database as information is only stored on the browser

localStorage is limited to 5MB across all major browsers

localStorage is quite insecure as it has no form of data protection and can be accessed by any code on your web page

localStorage is synchronous, meaning each operation called would only execute one after the other

dev//bruges kun under udvikling og ikke senere i serveren

npm start serveren med nodemon (så vi ikke skal reloade ed hver ændring)

require / document is not defined = problemer med at forbinde html, api og controllers