Eksamen for PG6301 Web utvikling og API design, 2023

Dette er en praktisk eksamen som skal evaluere din mestring av teknologiene og konseptene som har blitt gjennomgått i faget: React, Express, Mongodb, Heroku, Jest, OpenID Connect og Web Sockets. At det er en praktisk eksamen innebærer at du evalueres basert på hva du lager og hvordan det fungerere og ikke på teoretisk kunnskap du har tilegnet deg.

For å demonstrere dette skal du lage en applikasjon der brukere kan snakke sammen i chatrom

Læringsmålene eksamen skal demonstrere er som følger:

1. Lage en app med parcel, express, concurrently, prettier, Jest
2. Sette opp en fungerende React app med React Router, håndtering av loading state og feilhåndtering
3. Sette opp en fungerende Express app inkludert express Routes i egen fil
4. Kommunikasjon mellom klient og server med GET, PUT og POST inkludert feilhåndtering
5. Deployment til Heroku
6. Lagring, henting og oppdatering av data i Mongodb
7. Login med OpenID Connect (både Google og Microsoft Entra ID)
8. Web Sockets
9. Test coverage på 50-70%

- For å oppnå A må alle 9 av disse være dekket
- For å oppnå B må 8 av disse være dekket
- For å oppnå C må 6-7 av disse være dekket
- For å oppnå D må 4-5 av disse være dekket
- For å oppnå E må 2-3 av disse være dekket og applikasjonen må enten kjøre på Heroku eller `npm install && npm test && npm start` må kjøre uten feil og gi en brukbar applikasjon

Om du kun leverer kode identisk med koden fra forelesningen teller det som et halvt poeng. Om du løser punktet delvis teller det delvis. Om du har mestret noe, men har fått problemer med å få det til å virke på eksamen kan du dokumentere det i README.md for å få delvis uttelling. Vesentlig mangler i kodestil, ustabil oppførsel eller fraværende eller problematisk CSS layout trekker ned.

Innleveringen skal være en ZIP-fil med din kildekode som eksportert fra Github som et vedlegg i Wiseflow.

README-fila skal innehold link til Heroku slik at programmet kan testes ut og til GitHub repository

Viktig:

- Eksamen må besvares i Github Classroom med følgende link: https://classroom.github.com/a/pgC2zHhI
- Du mister tilgang til GitHub 23. november, kl 09:00
- Lever en ZIP-fil (ikke RAR, 7z eller tilsvarende). Den beste måten er å utvikle på Github og velge "Code" > "Download ZIP" i ditt repository på github.com
- README-fila på Github og i ZIP-fila må inneholde link to Github repository og Heroku-applikasjonen
- Genererte filer som `.node_modules`, `.parcel-cache`, `dist` og `coverage` må ikke være sjekket inn på Github eller i ZIP-fila
- Les oppgavebeskrivelsen grundig og sjekk at du ikke har oversett noen funksjonelle krav
- Du trenger ikke å løse alle krav for å få en ok karakter, prioriter hvilke krav du vil løse og dokumenter eventuelle forenklinger du har utført i README.md

Samarbeid under eksamen:

Du kan be andre om hjelp under eksamen, men du må angi i README.md kode du ikke har skrevet eller du har delt med andre om det ikke skal regnes som plagiat. Du vil bli evaluert på egen kode du leverer og bør angi hvilken kode du ikke selv har utviklet

Oppgavebeskrivelse

Du skal implementere en webbasert chat-løsning med brukerprofil

Når brukeren kommer til nettsiden skal de få en presentasjon av siden og mulighet til å logge seg inn med minst to identity providere. Du kan benytte Google, Microsoft Entra ID eller implementere Facebook eller Github om dere foretrekker.

En innlogget bruker kan chatte med andre brukere og se tidligere chatmeldinger

Det anbefales at du starter med å lage en plan for hvordan du skal løse innleveringen. Du kan legge planen i README-fila om du vil

Funksjonelle krav:

- [ ] Anonyme brukere skal ikke kunne se chatloggen, men skal kunne logge seg inn
- [ ] Brukere kan logge seg inn. Det anbefales at du implementerer at brukerne logger seg inn med Google, men andre mekanismer er også akseptabelt
- [ ] En bruker som er logget inn kan se på sin profilside (userinfo fra Google)
- [ ] Innloggede brukere skal kunne sende chatmeldinger
- [ ] Meldinger som sendes inn skal lagres i Mongodb
- [ ] Innloggede brukere skal kunne se chatmeldinger umiddelbart. Bruk websockets for å hente oppdateringer
- [ ] Chatmeldinger skal inneholde navnet på brukeren som skrev dem. Navnet skal hentes fra identity provider (Google, Entra ID)
- [ ] Når bruker logger seg inn skal websiden hente eksisterende meldinger fra MongoDB
- [ ] Kreves for A/B: Brukere som har logget på med Entra ID kan opprette chatroom med tittel og beskrivelse. Brukere må når de skrive chat meldinger velge chatroom for meldingen.
- [ ] Kreves for A/B: Systemet skal hindre brukere fra å opprette to chatroom med samme tittel
- [ ] Kreves for A: Brukeren som opprettet et chatroom skal kunne endre tittel og beskrivelse
- [ ] Kreves for A/B: Brukeren kan legge inn navn og bio på sin brukerprofil. Brukere kan se andres brukerprofil
- [ ] Kreves for A/B: Brukere kan endre navn og bio i sin brukerprofil
- [ ] Brukere skal forbli logget inn når de refresher websiden
- [ ] Alle feil fra serves skal presenteres til bruker på en pen måte, med mulighet for brukeren til å prøve igjen

Må-krav til teknisk løsning

- [ ] Besvarelsen skal inneholde en README-fil med link til Heroku og GitHub
- [ ] `npm run dev` skal starte server og klient. Concurrently og parcel anbefales
- [ ] `npm test` skal kjøre tester. Testene skal ikke feile
- [ ] Koden skal ha konsistent formattering. Prettier og Husky anbefales
- [ ] Nettsidene skal ha god layout med CSS Grid og horisontal navigasjonsmeny. Brukeren må kunne navigere overalt uten å bruke "back" eller redigere URL
- [ ] Nye chatmeldingene må sendes til klienten via websockets. Endringer i brukerprofil, historiske meldinger og chatroom kan hentes og oppdates med GET/POST/PUT
- [ ] Serveren validerer at brukeren er logget inn

- [ ] Innleveringen skal være i form av en ZIP-fil. Maks størrelse på fila er 1MB
- [ ] Chatmeldinger skal lagres i MongoDB
- [ ] Applikasjonen skal deployes til Heroku
- [ ] Applikasjonen skal ha tester for visning og brukerhandlinger i React og for API i Express