Eksamen 2024

Velkommen

Da er det tid for å bli enda tryggere på JavaScript, Next.js, Api, Hono og Databaser. Anbefaler å komme i gang med en gang. Sett dere inn i oppgavene og fordel dem mellom dere. Husk at oppgavene er ment for å gi mye læring.

Link til kode

https://www.loom.com/share/83eee115f6c14f54952f3bb2fe290a2e?sid=447cb023-0c53-428e-8f0a-b94aaf88e7c9

Generell info

Gruppeeksamen

- Gruppene består av 3-4 personer og må opprettes selv i Canvas
- Innlevering skjer i Inspera innen eksamensfristen
- Gruppen må levere refleksjonsnotat
- Studenter som ikke har gruppe ved eksamensstart blir tildelt en gruppe

Veiledning (endringer kan forekomme)

Det blir mulighet for veiledning underveis i eksamensperioden.

- Avklaringer gjøres fortløpende på Discord. Send en melding til meg så svarer jeg så fort jeg kan. Informasjon som gjelder alle legges til egen FAQ på denne siden.
- Annen veiledning blir gjennomført **digitalt**, fortrinnsvis via Discord / Zoom.

    - Hver gruppe har maks 20 minutter i løpet av eksamensperioden
- Booking av avtale må **skje seinest 12.00 to dager før via Discord til Tommy eller Marius.** Dette da begge er i full jobb og må planlegge med arbeidsgiver. På Discord er begge relativt fleksible. Tidsmessige kan 09-16 være utfordrende, men vi kan også kveld og helger.

Forutsetningen for å få hjelp er:

- At dere har klart for dere hva det trengs bistand til
- At problemet / det dere trenger bistand til forklares til veileder på Discord i forkant
- At siste versjon av koden er tilgjengelig på Github

Hvis bug / feil:

- At vi får screenshot av eventuell feilmeldingen
- At du har prøvd å løse feilen (Google, console.log for å utforske dataen du jobber med, JSON.stringify(), lest dokumentasjon, YouTube, Ulearn, en venn m.fl)

Discord

Det har blitt opprettet en kanal i Discord (#prosjekt-spørsmål) for eksamen. Her kommer det informasjon som er relevant for alle.

- I denne kanalen kan dere også få hjelp til relevante spørsmål.
  Forutsetningen for å få hjelp er:
    - At problemet forklares til oss på Discord
    - At siste versjon av koden er tilgjengelig på Github slik at vi har tilgang til koden
    - At vi får screenshot av feilmeldingen
    - At du har prøvd å løse feilen

Plagiat

Kode du låner / kopierer fra diverse ressurser, må kommenteres. Dette for å unngå at hele gruppen stryker grunnet plagiat. Trenger ikke å henvise til kode brukt fra kurset.

Kode fra CodePilot og / eller ChatGPT skal også kommenteres.

```
// SRC: kilde: ...url/
```

GitHub

GitHub skal brukes og dere bør

- Bruke branches
- Commit ofte
- Samle all kode som virker i `main`

Prosjektdokument / Rapporten

Dette dokumentet skal fylles ut av gruppen og leveres sammen med eksamen. Link til dokumentet finner du her.

Absolutte krav

- At prosjektdokumentet er levert
- Applikasjonen må være kjørbar (at det fungerer å starte prosjektet)
- Masterbranchen inneholder alle deltakerne sin kode (at alt er merget med master eller main)
- At det er få eller ingen "røde" errors i Google Chrome consolen
  - Unntatt om det er hydreringsmissmatch grunnet random verdier
- At alt som skal leveres inn er levert inn til fristen
- At sensor har tilgang til GitHub-prosjektet
- At alle deloppgaver som gir en ønsket karakter er bestått
- At en oppgave med lavere oppgavenummer er bestått
  - Eks. Oppgave 1 må bestås før Oppgave 2 kan bli vurdert
- At du behersker bruk av tredjeparts biblioteker, men samtidig kan skrive egen logikk der det er hensiktsmessige
- At du kan lage komponenter uten bruk av UI-biblioteker
- At du benytter Typescript for å type enkle typer på props i komponenter

    - String
    - Number
    - Boolean
    - Array
        - [1,2,3]
        - [”Lars”, “Ole”, “Karoline”]
    - Record
        - { id: 1, title: “Title 1”, authors: [{ id: 1, name: “Stian”}, {id: 2, name: “Mona”}]}
    - Array med Records
        - [{ id: 1, title: “Title 1”}, {id: 2, title: “Title 2”}]
    - Enkle funksjoner
- At alle sider har et minimum av design som gjør det enklere for sensor å vurdere innholdet

Eksempel på akseptansekrav

> Skal være mulig å legge til en kommentar på et produkt

Dette kriteriet har flere krav som må oppfylles for å få full pott

- Du kan ha en `frontend` med `dummydata`, men ingen `API`. Kriteriet er da delvis oppfylt.
- Du kan ha en `backend` som kan testes med f.eks `Postman`, men ingen `frontend`. Kriteriet er da delvis oppfylt.
- Du kan ha en `frontend` og `API`, men de kommuniserer ikke sammen. Kriteriet er da delvis oppfylt.
- Du kan ha en `frontend` og `API`, men oppfyller ikke krav til `design`, `validering` og `errorhåndtering`. Kriteriet er da delvis oppfylt.
- Du kan ha alt det nevnte, men har mangler nevnt under. Kriteriet er da delvis oppfylt.

    - Enkle typer
    - Tester (unit, integrasjon)
    - Konsistent kodebase
    - Fornuftig oppdeling av kode
    - Gode abstraksjoner
    - +++

Karakter

Oppgavene står i prioritert rekkefølge. Sensor setter karakter etter hvilken grad akseptansekravene er oppfylt. Helhetsvurderingen legges til grunn hvis det er få mangler på et akseptansekrav / deloppgaver. Endringer kan forekomme.

Krav til karakter

- F-E
    - Oppgave 1.1, 1.2, 1.3
    - Oppgave 2.1 og 2.2
    - Funksjonaliteten må virke slik den er beskrevet
    - Grad av mangler og kvalitet avgjør karakter F-E
- F-D
    - Oppgave 1.2 og 1.3
    - Oppgave 2.1, 2.2 og 2.3
    - Grad av mangler og kvalitet avgjør karakter F-D
- F-C
    - Oppgave 1.2 og 1.3
    - Oppgave 2.1, 2.2, 2.3 og 2.4
    - Grad av mangler og kvalitet avgjør karakter F-C
- F-B
    - Oppgave 1.2, 1.3 og 1.4
    - Oppgave 2.1, 2.2, 2.3, 2.4, 2.5, 2.6 og 2.7

    - Grad av mangler og kvalitet avgjør karakter F-B
- F-A
    - Oppgave 1.2, 1.3, 1.4 og 1.5
    - Oppgave 2
    - Grad av mangler og kvalitet avgjør karakter F-A

Levering av eksamen

Når dere skal levere gruppeeksamen må dere

- Lage en gruppe i Inspera
    - Inviter alle deltakerne til denne gruppa
- Grupperapporten / refleksjonsnotat som baserer seg på malen dere har fått. Ved behov kan også individuelle rapporter leveres. Denne rapporten skal inneholde link til GitHub repo deresLaste opp en .zip fil med
    - (Hvis relevant) tekstdokumenter som inneholder:
        - Antagelser dere har gjort i oppgaven som sensor bør kjenne til
        - Linker til kode dere har kopiert fra eksterne kilder (stack overflow, google, ++) (for å unngå plagiat) og hvor i koden dere har brukt det
        - Informasjon sensor må kjenne til for å starte prosjektet (.env variabler e.l)
        - Annet du tenker sensor bør kjenne til (f.eks Wireframes fra IA e.l)
    - Hele prosjektet **UTEN** node_modules, lastes opp som egen .zip.
    - Slett oppgaver dere ikke har gjort / oppgaver som ikke er nødvendig for karakteren du sikter mot
- Inviter **hiofkonto@gmail.com** til deres **GitHub** repo med **alle rettigheter**
- Krav til kode
    - Se hver enkelt oppgave
    - NB: Det er ingen garanti at man får en gitt karakter selv om man har levert inn de oppgavene som kreves. Kvaliteten på koden må også være god nok.

Oppgave 1

En student har jobbet hele sommeren med nytt UI for en mikro LMS. Dessverre ble ikke studenten helt ferdig med alt samme. Din oppgave er å ferdigstille løsningen slik at alt fungere som det skal.

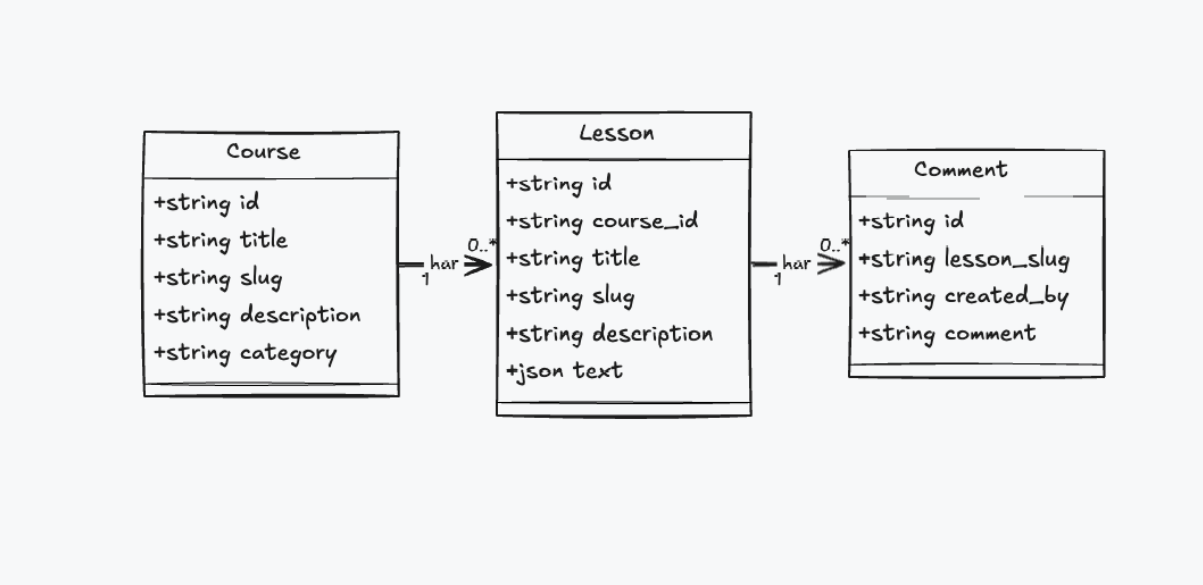
Oppgaven er delt opp i flere deler hvor ulike deler kvalifiserer til ulike karakterer (se over)

Link til kode

Data og design

Vedlagt er wireframes studenten fikk før implementasjonen startet. Studenten fikk også et forslag til et diagram som viser relasjonene i dataen.

Wireframes.pdf



Dette er akseptansekriteriene kunden har satt:

Rydde i koden

- Kunden har god teknisk innsikt og ser at dagens kode bør omorganiseres, justeres og optimaliseres for å sikre at fremtidig utvikling blir enklere

Navigering / routing / dynamiske url-er

- Kunden ser at ikke alt virker som det skal når siden lastes inn. Forsiden (siden for login) er nå tom og de ulike sidene knyttet til kurs, leksjoner og lage nytt kurs vises ikke / kan ikke navigeres til

Backend / frontend

I dag er all informasjon statisk, noe som gjør det vanskelig å lage samt vise kurs.

- Kunden har ingen store forventninger til at backend er lagdelt da dette er en liten applikasjon, men utover det opptatt av at koden har god kvalitet.
- Kunden foretrekker Hono med Node som backend-server. Når det kommer til valg av database er SQLite å foretrekke

  Kurs og leksjoner

  - Kunden ønsker at dagens data skal være tilgjengelig i applikasjonen
  - Kunden ønsker at fremtidige kurs og leksjoner lagres og vises i løsningen
  - Kunden ønsker at kurs kan slettes. Da skal også tilhørende leksjoner og kommentarer slettes
  - Kunden har ingen krav til at et kurs kan endres, men optimalt sett kan kategorien på kurset endres
  - Kunden foretrekker at filtreringen kun gjøres i frontend slik det gjøres i dag

  Kommentarer

  - Kunden ønsker at kommentarer som allerede finnes lagres i databasen
  - Kunden har ingen krav om at fremtidige kommentarer lagres, men optimalt sett gjøres dette

Bedre teksteditor

- Dagens kode bruker textarea for å lage tekst til leksjonene. De ønsker at f.eks TipTap brukes isteden, men kun enkleste versjonen.

Testing

- Kunden hadde krav om at det å opprette et nytt kurs er godt testet. De ønsker i hovedsak end til ende tester og har hørt mye bra om Playwright. De la ved en omfattende oversikt som de opplever er dekkende for hva som må testes

1.1 (planlegge)

Dokumentasjonen kan lages som Markdown i VSCode og legges i mappen `documentation` (må lage mappen selv i oppgave_1). Dokumentasjonen skal være kortfattet.

1. Skal dokumentere hvilke api-endepunkter (ressurser) som skal brukes
2. Skal til hvert api-endepunkt dokumentere hvilke verb som er tilgjengelig. Hva slags forespørsler skal de håndtere.
3. Skal til hvert api-endepunkt dokumentere responsen og statuskoden for de ulike verbene. Hva slags data skal returneres når det går riktig / feil.
4. Skal dokumentere hvilke sider (urler) som skal benytte de ulike APIene og grovt hva som kan gjøres på den enkelte siden. Hvilke sider i "app" skal opprettes og grovt hva som kan gjøres på de ulike sidene.

1.2 (endringer frontend)

Gjøre nødvendige tilpasninger i frontend slik at kunden blir fornøyd. I hovedsak innebærer dette `Rydde i kode` og `Navigering / routing / dynamiske url-er` nevnt over

1.3 (backend)

Sikre at kravene nevnt i `Backend` ivaretas. I hovedsak CRUD operasjoner med hensiktsmessige abstraksjoner, database og kvalitet i koden.

1.4 (editor)

Kunden ønsker at dagens funksjonalitet virker som den skal, men vil ha en mulighet for å bruke en annen og bedre editor ved behov.

Optimalt sett ønsker de å kunne veksle mellom dagens løsning og kravene nevnt i `Bedre teksteditor`. I fremtiden kan det hende de vil ha andre editorer og det er derfor viktig at det blir enkelt å få til dette.

I tillegg til implementasjon ønsker kunden at endringene i koden dokumenters slik at nye utviklere forstår hvorfor og hvordan dette ble ivaretatt

1.5 (testing)

Kunden foretrekker at så mange som mulig av dagens testkriterier blir ivaretatt og i henhold til kravet `Testing`.

Oppgave 2

En liten forening har mistet kontroll over hvem som er påmeldt på ulike arrangementer. De har nå ikke oversikt over hvilke arrangementer de har og hvem som er påmeldt de ulike. Det vet heller ikke hvem som har betalt. Din oppgave er å hjelpe de slik at de får bedre kontroll.

Link til kode

Oppgave

Din oppgave er å lage en prototype for "arrangement med booking"-løsning.

Data og design

Kunden er åpen for forslag og vil gjerne se noen enkle skisser som ivaretar kravene under.

Dette er akseptansekriteriene kunden har satt:

Bestiller / kundeside

Sette opp landingsside

- Hver arrangement skal ha en landingsside med unik slug (urlen til siden. eks `dette-er-en-url`) / eller `id-en` til arrangementet
- Skal ved visning av arrangementet ha med nødvendig data for at kunde skal vite hva slags arrangement det er. Eksempel på data er tittel på

arrangementet, beskrivelse av arrangementet, `slug`, når arrangementet er, lokasjon, type arrangement, antall plasser, pris m.m

Styre visningen av arrangementer

- Bestiller skal kunne se arrangementer per mnd og år
- Bestiller skal kunne filtrere på type arrangement
- Bestiller skal se om arrangementet er ledig / fullbooket / e.l hvis aktuelt

Sette opp skjema for å melde seg på et arrangement

- Skjema ber om nødvendig informasjon om bestilleren
- Optimalt sett ønsker kunden at det er mulig å legge til flere personer på samme bestilling

Regler skjema

- Skjema tar hensyn til om det er plass på arrangementet, men tillater venteliste om det er satt opp som et alternativ på arrangementet
- Skjema viser hva total pris vil bli for bestillingen (om det er pris)
- All data er påkrevd og kritisk data må valideres

Admin

Administrere arrangementer

Når kunden skal jobbe med arrangementer har de tre måter å løse dette på.

1. De kan lage en **mal** for arrangementer de ønsker å gjenbruke (eks. mal for styremøter som har et sett av regler og forhåndsutfylt data)
2. De kan lage et arrangement basert på en mal de har laget. Ved opprettelse av arrangementet kan deler av malen overstyres. Selve malen endres ikke.
3. De kan lage et arrangement fra scratch

- Skal kunne lage maler for arrangementer som går igjen

    - Malene skal kunne settes opp med ulike regler. Mulige regler som kunden ønsker er for å redusere antall feil ved publisering av arrangementer er.

      > 💡 Mulige regler er nevnt under. Start med noen regler til å begynne med og se hvor langt du kommer

        1. Bestemme om en mal ikke tillater andre arrangementer på samme dag. Hvis tilfelle kan den ikke benyttes på en dato hvor det allerede finnes et arrangement.
        2. Bestemme at en mal skal låses til en ukedag(er). Regelen skal da ikke gjøre det mulig å opprette denne på andre dager enn de som er valgt. Denne verdien kan ikke overskrives hvis satt på malen.
            - F.eks Fredagsjoggen-mal kan kun være på fredager
        3. Bestemme om en mal skal være privat. Den skal da ikke vises i listen. Kunden må da ha linken til denne for å se arrangementet.
        4. Bestemme om en mal har begrenset antall plasser
        5. Bestemme om en mal har en fast pris
        6. Bestemme om en mal er gratis
        7. Bestemme om en mal skal ha venteliste
- Skal kunne lage et nytt arrangement fra scratch eller ta utgangspunkt i en mal.
    - Et nytt arrangement må ivareta regler nevnt over, men er utover det kun basert på basis-dataen et arrangement har.
        - Eksempelvis kan vi ikke opprette et arrangement på samme dag om regel `1` er satt
- Skal kunne endre data på et arrangement etter opprettelse. Regler må ivaretas.
- Skal kunne slette et arrangement
- Skal ikke kunne endre eller slette en mal som er i bruk

Administrere påmeldinger

- Skal kunne administrere påmeldinger til et arrangement i admin-panelet
  - Godkjenne påmeldingen / flere påmeldinger
  - Avslå påmeldingen / flere påmeldinger
  - Slette en påmelding
  - Se påmeldinger
  - Se påmelding
  - Legge til manuelle påmeldinger

Statistikk

Statistikk

- Laste ned et excel ark som viser alle påmeldinger og tilhørende informasjon per år
- Antall påmeldt per arrangement
- Antall nye påmeldinger siste måned

2.1 (planlegge)

Dokumentasjonen kan lages som Markdown i VSCode og legges i mappen `documentation` (må lage mappen selv i oppgave_2). Dokumentasjonen skal være kortfattet.

1. Skal lage enkle low-fi skisse basert på kravene over
2. Skal dokumentere hvilke api-endepunkter (ressurser) som skal brukes
3. Skal til hvert api-endepunkt dokumentere hvilke verb som er tilgjengelig. Hva slags forespørsler skal de håndtere.
4. Skal til hvert api-endepunkt dokumentere responsen og statuskoden for de ulike verbene. Hva slags data skal returneres når det går riktig / feil.
5. Skal dokumentere hvilke sider (urler) som skal benytte de ulike APIene og grovt hva som kan gjøres på den enkelte siden. Hvilke sider i "app" skal opprettes og grovt hva som kan gjøres på de ulike sidene.

6. Skal dokumentere hvordan filtreringen skal foregå og løses i frontend og backend
7. Skal dokumentere datamodellen og bakgrunnen for denne modellen
8. Skal dokumentere hvordan løse det å opprette / gjenbruke en mal
9. Skal dokumentere databasemodellen og nødvendige relasjoner

2.2 (frontend del 1)

I denne delen av frontend skal du lage grensesnittet for `Bestiller / kundeside`. Tar ikke høyde for noen regler nevnt i kravene over.

2.3 (backend del 1)

I denne delen av backend skal du lage endepunkter for `Bestiller / kundeside`. Tar ikke høyde for noen regler nevnt i kravene over. Data må lagres i database.

2.4 (backend og frontend del 1)

I denne delen skal du koble sammen backend og frontend. Tar ikke høyde for regler nevnt i kravene over.

2.5 (frontend del 2 - admin)

I denne delen av frontend skal du lage grensesnittet for `admin`. Aktuelle regler må ivaretas.

2.6 (backend del 2 - admin)

I denne delen av backend skal du lage endepunkter for kravene som omhandler `admin`. Aktuelle regler må ivaretas.

2.7 (backend og frontend del 2 - admin)

I denne delen skal du koble sammen backend og frontend for `admin`. Aktuelle regler må ivaretas.

2.8 (lagdele backend for admin)

I denne delen skal du kun lagdele den delen som er koblet til oppgave `2.6`. Aktuelle regler må ivaretas.

2.9 (statistikk)

I denne delen skal du implementere nødvendig logikk for å sikre at ett av kravene under `statistikk` ivaretas.

FAQ

FAQ