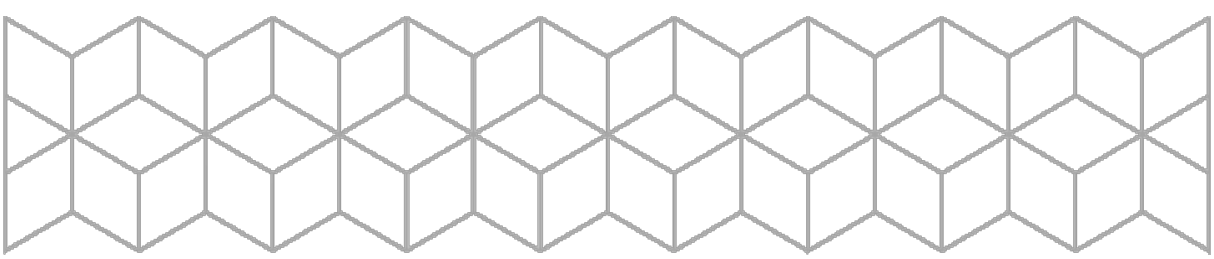
Høgskolen i Østfold

SENSORVEILEDNING

Emnekode:	ITM30617
Emnenavn:	Utvikling av interaktive nettsteder
Eksamensform:	Todelt eksamen Del 1: Prosjekt Del 2: Skriftlig
Dato:	Del 1: Innlevering 19.05.2023 Del 2: Skriftlig 25.05.2023
Faglærer(e):	Marius Akerbæk Ann-Charlott Karlsen
Eventuelt: Prosjektet blir gjort som gruppe eller individuelt hvor alle hjelpemidler er tillatt Skriftlig er individuell med ingen hjelpemidler tillatt	



Eksamen

Nettsted og individuell skriftlig eksamen

Individuell sluttkarakter settes på bakgrunn av to deleksamener. Hver deleksamen må være bestått for å få hele emnet bestått. Karakterskala A-F.

Deleksamen 1: Prosjekt

Studentene skal utvikle et nettsted i henhold til eksamensoppgaven. Prosjektet leveres digitalt med en link til et Github repository i insperia. Nettstedet kan leveres som gruppeoppgave med 3-4 studenter i hver gruppe eller individuelt:

Nettstedet teller 60 % av den samlede karakteren for emnet. Det gis individuell karakter A-F.

Deleksamen 2: Individuell skriftlig eksamen, 1 time

Skriftlig eksamen teller 40 % av den samlede karakteren. Det gis individuell karakter A-F.

Del 1: Prosjekteksamen

Studentene kan velge ambisjonsnivå for eksamen, i prosjektmappen informerer studentene om hvilket ambisjonsnivå de har valgt å gå for. Alle kravspesifikasjoner som beskrevet i oppgaveteksten må våre nådd for å oppnå valgt ambisjonsnivå (Karakter).

Innledende oppgavetekst

I eksamensoppgaven skal dere lage en applikasjon som fungerer som et spillbibliotek og -butikk (tenk ala Steam Links to an external site. eller PS Store

Links to an external site.). Stikkord for funksjonalitet som skal finnes i applikasjonen er butikk, liste over dine spill og dine favoritter.

I eksamen-mappen i kursets GitHub-repository

Links to an external site. ligger noen skisser og bildefiler som kan brukes i eksamensoppgaven, games.js som inneholder innholdsressurser for minstekrav og karakter D-oppgaven, samt en sass-partial med fonter og farger som dere valgfritt kan benytte i oppgavene som krever SASS.

Skissene i denne mappen er ment som veiledende og for en forståelse av noen av kravene i oppgaven, men dere stiller fritt til å designe applikasjonen som dere ønsker. Dette inkluderer mer/annet innhold (men ikke mindre), mer/utfyllende funksjonalitet etc. PS: Fokuser på å oppnå kravene for hver karakteroppgave før dere legger til ekstra innhold/funksjonalitet. Kravene nevnt i oppgaven må oppnås for å nå ønsket karakter.

Oppgaven vil være delt inn etter mulig karakteroppnåelse, hvor utnyttelse av mer komplekse teknologier gir bedre karakter.

Minstekrav (karakter E)

Minstekrav som tilsvarer en E hvis alle kravene er møtt. Minstekravene gjelder alle oppgavene og alle kravene må være møtt for å få godkjent. Er et av minimumskravene ikke møtt uansett ambisjonsnivå og antall møtte krav ved andre nivåer er oppgaven ikke godkjent og tilsvarer en F.

Det er maksimalt mulig å oppnå en karakter E ved gjennomføring av kun minstekrav.

- Appen skal installere når sensor kjører npm install
- Appen skal starte uten feilmeldinger etter npm install er utført (test dette før innlevering!)

Komponenter og routing

Applikasjonen skal bruke React, og ha minimum fire komponenter:

- Dashboard (forside), som har tre seksjoner; Gameshop, My Games og My Favourites (se skisse .)

- MyGames (som lister opp mine spill)
- MyFavourites
- GameShop

Disse komponentene skal linkes til ved hjelp av Routing fra menyen i toppen av applikasjonen, samt fra de respektive knappene på dashboardet (se skisse i GitHub-mappe). Vi anbefaler å bruke rutene

- / (forsiden, altså dashboard)
- /gameshop
- /mygames
- /favourites

Innhold

For minstekrav skal innholdet i filen *games.js*

Links to an external site. brukes. I games.js exporteres to const-variabler med innhold; **store**, som inneholder spill til butikkdelen, og mygames, som inneholder spill til eget bibliotek/favorittliste.

OBS: For å få hentet inn disse separat i koden må de importeres som variabler:

import {store, mygames} from "./games"

Krav til innhold

- I dashboard skal
    - seksjonen Gameshop hente ut tre vilkårlige spill fra store-arrayen i games.js, og vise disse i henhold til skissen*. Knappen BUY skal lede brukeren til kjøpslenken fra spillobjektet i games.js.

    - seksjonen My Games-library hente ut fire spill fra games-arrayen i games.js, og viser disse i henhold til skissen*
    - seksjonen My Favourites hente ut to spill fra games-arrayen i games.js som er favoritter, og vise disse i henhold til skissen*
- **skissen* viser til dashboard_minstekrav.png fra eksamens-mappen i github-repositoriet
- MyGames skal vise alle spillene i games-arrayen i games.js
- MyFavourites skal vise alle spill som er favoritter i games-arrayen i games.js
- GameShop skal vise alle spillene som ligger i store-arrayen i games.js. Merk at disse skal ha en knapp med mulighet for å kjøpe som linker til kjøpslenken.

Studentene må selv holde kontroll over hvilke props som må sendes til de ulike komponentene.

HTML

- Vi forventer at applikasjonen er bygget med semantisk, godt strukturert HTML-kode
- Vi forventer at applikasjonen designes responsivt med CSS

Karakter D-krav

Gjennomføre oppgaven ved å følge kravspesifikasjon som beskrevet i oppgaven for å oppnå karakter D, alle kravspesifikasjonene må være gjort for å oppnå karakter D ved ambisjonsnivå D. Hvis minstekravene fra oppgave E ikke er møtt er oppgaven ikke godkjent og tilsvarer karakter F. Hvis ikke alle kravspesifikasjonene for ambisjonsnivået er møtt går man ned en karakter. Ved ambisjonsnivå D tilsvarer ikke nådde kravspesifikasjoner E-F
I tillegg til kravene og oppgavene nevnt i dette avsnittet, gjelder også minstekrav for karakter D.

Komponenter og routing

I tillegg til komponenter fra minstekrav, skal applikasjonen ha komponentene

    - GameCard, som viser et spillkort
    - GamePage, som viser et spill med all informasjon
- Et spill i dashboard, My Games-librariet og My Favourites skal kunne klikkes på (enten hele spillkortet, eller med en knapp/lenke til et spill). Tips: lag en ny property i objektene i games-arrayen kalt "slug" med en pen variant av tittelen du kan bruke som parameter i URL/Routing for å peke til ett spill. Når linken åpnes, brukes GamePage for å vise et enkelt spill (må hentes fra arrayen) med all informasjon knyttet til dette spillet fra games-arrayen i games.js.
- Vi anbefaler å bruke ruten /game/gametitle-as-slug for ett enkelt spill

Karakter C-krav

Gjennomføre oppgaven ved å følge kravspesifikasjon som beskrevet i oppgaven for å oppnå karakter C, alle kravspesifikasjonene må være gjort for å oppnå karakter C ved ambisjonsnivå C. Hvis minstekravene fra oppgave E ikke er møtt er oppgaven ikke godkjent og tilsvarer karakter F. Hvis ikke alle kravspesifikasjonene for ambisjonsnivået er møtt går man ned til en tilsvarende karakter for møtte kravspesifikasjoner. Ved ambisjonsnivå C tilsvarer ikke nådde kravspesifikasjoner D-F

C-kravene bygger på Minstekrav og karakter D-krav.

Eksternt API; Rawg

For å oppnå karakter C erstattes ressursfilen fra minstekrav med å bruke et eksisterende API; Rawg.io

Links to an external site.. (dokumentasjon og anskaffelse av API-nøkkel: rawg.io/apidocs

Links to an external site.) Du skal ikke bruke games.js-filen fra minstekrav/Karakter D-kravene dersom du går for Karakter C eller høyere.

Funksjonaliteten beskrevet under minstekrav og Karakter D-krav er fortsatt gjeldende, men oppgaven skal bruke asynkronitet og API-kall mot Rawg for spill-innhold. For å gjøre det mer oversiktlig når innhold hentes fra API:

- For GameShop, hent ut de tre nyeste spillene for visning i dashboard. Hent ut de 10 nyeste for visning på /gameshop (når du klikker linken "Visit Shop")
- For MyGames, hent 4 spill fra en valgri sjanger som vises i seksjonen My Games i dashboard, og 20 spill fra samme sjanger på My Games-biblioteksiden (/mygames)
- For MyFavourites, lag funksjonalitet som gjør følgende:
    - På visning av ett spill (, ha en knapp "Legg til favoritter".
    - Klikk på knappen "Legg til favoritter" skal lagre spillet i en array i en state kalt favourites.
    - MyFavourites skal hente/vise spill fra favourites-staten.

For karakter C og høyere forventer vi at studentene selv tar ansvar for refaktorering (bruk av komponenter og props på en hensiktsmessig måte).

OBS: Husk en attribution-link til APIet skal ligge i footer-seksjonen av applikasjonen (krav for bruk av API-et).

HTML

- Vi forventer at applikasjonen designes responsivt med CSS ved hjelp av SASS (for enklest mulig integrasjon av sass, se Adding a sass stylesheet fra create-react-app.dev

Links to an external site.)

Karakter B-krav

Gjennomføre oppgaven ved å følge kravspesifikasjon som beskrevet i oppgaven for å oppnå karakter B, alle kravspesifikasjonene må være gjort for å oppnå karakter B ved ambisjonsnivå B. Hvis minstekravene fra oppgave E ikke er møtt er oppgaven ikke godkjent og tilsvarer karakter B. Hvis ikke alle kravspesifikasjonene for ambisjonsnivået er møtt går man ned til en tilsvarende karakter for møtte kravspesifikasjoner. Ved ambisjonsnivå B tilsvarer ikke nådde kravspesifikasjoner C-F

For å oppnå karakter B (og A) skal deler av innholdet styres gjennom Sanity.

- Opprett et nytt Sanity-prosjekt, og koble dette til React-applikasjonen.
- Lag datamodell og -innhold for

    - spill (må kunne lagre en tittel, slug, API-id, timer spilt, sjangere og om det ligger i favorittlisten)
    - sjanger
- Legg inn minimum 10 spill i Sanity. Disse utgjør basen for "My Games"-biblioteket i Appen. Bruk feltet API-id for å lagre spillets id (eksempel; i APIet har spillet "Pocket City 2" id-en 957651).
- I dashboard, legg til opptelling av antall spill for My Games-biblioteket og Favourites. Se skissen dashboard_gradeAB.png i GitHub-repositoriet. Tips: GROQ-metoden *count*.
- På en side for ett spill, bruk feltet API-id for å hente ekstra informasjon (i tillegg til informasjon som ligger i Sanity) som skal vises om spillet. Ekstra informasjon er
    - Bilde
    - Rating
    - Oppsummering/plot
    - Stikkord (tags)
    - Utviklere (developers)
    - Utgiver (publisher)
    - Utgivelsesår (release)
    - Plattformer (platforms)
    - Kjøpsmuligheter (stores)

Karakter A-krav

Gjennomføre oppgaven ved å følge kravspesifikasjon som beskrevet i oppgaven for å oppnå karakter A, alle kravspesifikasjonene må være gjort for å oppnå karakter A ved ambisjonsnivå A. Hvis minstekravene fra oppgave E ikke er møtt er oppgaven ikke godkjent og tilsvarer karakter A. Hvis ikke alle kravspesifikasjonene for ambisjonsnivået er møtt går man ned til en tilsvarende karakter for møtte kravspesifikasjoner. Ved ambisjonsnivå A tilsvarer ikke nådde kravspesifikasjoner B-F

For A-krav gjelder også alle kravene nevnt under B-krav.

- Lag en innholdstype "Bruker" i Sanity.
- Lag innlogging som sjekker bruker mot Sanity og logger inn denne. Sjekk på e-postadresse er tilstrekkelig (ikke krav til passord). Bruk en state eller localStorage.
- Alt spillinnhold må kunne knyttes til brukerkontoen. Applikasjonen skal kun hente innhold knyttet til den innloggede brukeren.
- Brukere skal kunne ha ulike favoritter. Datamodellen må modifiseres slik at favoritter lagres på bruker, ikke på spill.
- På spillsiden, bruk npm-pakken react-tagcloud

  Links to an external site. sammen med stikkord (tags) og gamescount-nøkkelen fra APIet for å lage en ordsky av stikkord.

Vurdering

- En god mappestruktur med fornuftige mappe- og filinndeling og -navngiving vil telle positivt i vurderingen. Fjern unødvendige filer (du trenger strengt tatt bare index.js

og app.js + minimum en css/sass-fil). PS: ved opprydning, husk å fjerne importer fra index.js og app.js.

- I tillegg til kravene beskrevet til funksjonalitet og teknologi i denne oppgaveteksten, vil gode prinsipper for webutvikling telle positivt (semantisk HTML, god bruk av CSS, universell utforming, responsivitet).
- Bruk av branches og commits vil vurderes.
- Implementering av Layout (ikke et krav) i routingen vurderes positivt.
- Refaktorering (fornuftig bruk/gjenbruk av komponenter og props) er en vesentlig del av vurderingen.
- Feilmeldinger (error og warnings) bør på det sterkeste unngås. Rydd opp i koden før innlevering.

Hele oppgaveteksten:

Eksamen skal leveres som en gruppeoppgave, ihht gruppeinndelingen Arbeidsgrupper.

LES HELE OPPGAVETEKSTEN NØYE FØR DERE BEGYNNER. Hvis dere eksempelvis vil gå for A-krav, bør dere lese de andre kravene først, siden disse bygger på hverandre! Selv om de bygger på hverandre, vil det være mye unødvendig arbeid om man vet at man går for B/A-karakter og koder seg gjennom minstekravene uten å se ulikhetene mellom oppgavene. Vi anbefaler sterkt å lese og planlegge oppgaven før dere begynner å kode.

Innledning

I eksamensoppgaven skal dere lage en applikasjon som fungerer som et spillbibliotek og -butikk (tenk ala Steam

Links to an external site. eller PS Store

Links to an external site.). Stikkord for funksjonalitet som skal finnes i applikasjonen er butikk, liste over dine spill og dine favoritter.

I eksamen-mappen i kursets GitHub-repository

Links to an external site. ligger noen skisser og bildefiler som kan brukes i eksamensoppgaven, games.js som inneholder innholdsressurser for minstekrav og karakter D-oppgaven, samt en sass-partial med fonter og farger som dere valgfritt kan benytte i oppgavene som krever SASS.

Skissene i denne mappen er ment som veiledende og for en forståelse av noen av kravene i oppgaven, men dere stiller fritt til å designe applikasjonen som dere ønsker. Dette inkluderer mer/annet innhold (men ikke mindre), mer/utfyllende funksjonalitet etc. PS: Fokuser på å oppnå kravene for hver karakteroppgave før dere legger til ekstra innhold/funksjonalitet. Kravene nevnt i oppgaven må oppnås for å nå ønsket karakter.

Oppgaven vil være delt inn etter mulig karakteroppnåelse, hvor utnyttelse av mer komplekse teknologier gir bedre karakter.

Se DEMO for forventet minimumskrav (E)

Minstekrav (karakter E)

Det er maksimalt mulig å oppnå en karakter E ved gjennomføring av kun minstekrav.

- Appen skal installere når sensor kjører npm install
- Appen skal starte uten feilmeldinger etter npm install er utført (test dette før innlevering!)

Komponenter og routing

Applikasjonen skal bruke React, og ha minimum fire komponenter:

- Dashboard (forside), som har tre seksjoner; Gameshop, My Games og My Favourites (se skisse

- Links to an external site.)
- MyGames (som lister opp mine spill)
- MyFavourites
- GameShop

Disse komponentene skal linkes til ved hjelp av Routing fra menyen i toppen av applikasjonen, samt fra de respektive knappene på dashboardet (se skisse i GitHub-mappe

Links to an external site.). Vi anbefaler å bruke rutene

- / (forsiden, altså dashboard)
- /gameshop
- /mygames
- /favourites

Innhold

For minstekrav skal innholdet i filen *games.js*

Links to an external site. brukes. I games.js exporteres to const-variabler med innhold; **store**, som inneholder spill til butikkdelen, og mygames, som inneholder spill til eget bibliotek/favorittliste.

OBS: For å få hentet inn disse separat i koden må de importeres som variabler:

```
import {store, mygames} from "./games"
```

Krav til innhold

- I dashboard skal
    - seksjonen Gameshop hente ut tre vilkårlige spill fra store-arrayen i games.js, og vise disse i henhold til skissen*. Knappen BUY skal lede brukeren til kjøpslenken fra spillobjektet i games.js.
    - seksjonen My Games-library hente ut fire spill fra games-arrayen i games.js, og viser disse i henhold til skissen*

    - seksjonen My Favourites hente ut to spill fra games-arrayen i games.js som er favoritter, og vise disse i henhold til skissen*
- **skissen* viser til dashboard_minstekrav.png fra eksamens-mappen i github-repositoriet
- MyGames skal vise alle spillene i games-arrayen i games.js
- MyFavourites skal vise alle spill som er favoritter i games-arrayen i games.js
- GameShop skal vise alle spillene som ligger i store-arrayen i games.js. Merk at disse skal ha en knapp med mulighet for å kjøpe som linker til kjøpslenken.

Studentene må selv holde kontroll over hvilke props som må sendes til de ulike komponentene.

HTML

- Vi forventer at applikasjonen er bygget med semantisk, godt strukturert HTML-kode
- Vi forventer at applikasjonen designes responsivt med CSS

Karakter D-krav

I tillegg til kravene og oppgavene nevnt i dette avsnittet, gjelder også minstekrav for karakter D.

Komponenter og routing

- I tillegg til komponenter fra minstekrav, skal applikasjonen ha komponentene
    - GameCard, som viser et spillkort
    - GamePage, som viser et spill med all informasjon
- Et spill i dashboard, My Games-librariet og My Favourites skal kunne klikkes på (enten hele spillkortet, eller med en knapp/lenke til et spill). Tips: lag en ny property i objektene i games-arrayen kalt "slug" med en pen variant av tittelen du kan bruke som parameter i URL/Routing for å peke til ett spill. Når linken åpnes, brukes GamePage for å vise et enkelt spill (må hentes fra arrayen) med all informasjon knyttet til dette spillet fra games-arrayen i games.js.
- Vi anbefaler å bruke ruten /game/gametitle-as-slug for ett enkelt spill

Karakter C-krav

C-kravene bygger på Minstekrav og karakter D-krav.

Eksternt API; Rawg

For å oppnå karakter C erstattes ressursfilen fra minstekrav med å bruke et eksisterende API; Rawg.io

Links to an external site.. (dokumentasjon og anskaffelse av API-nøkkel: rawg.io/apidocs

Links to an external site.) Du skal ikke bruke games.js-filen fra minstekrav/Karakter D-kravene dersom du går for Karakter C eller høyere.

Funksjonaliteten beskrevet under minstekrav og Karakter D-krav er fortsatt gjeldende, men oppgaven skal bruke asynkronitet og API-kall mot Rawg for spill-innhold. For å gjøre det mer oversiktlig når innhold hentes fra API:

- For GameShop, hent ut de tre nyeste spillene for visning i dashboard. Hent ut de 10 nyeste for visning på /gameshop (når du klikker linken "Visit Shop")
- For MyGames, hent 4 spill fra en valgri sjanger som vises i seksjonen My Games i dashboard, og 20 spill fra samme sjanger på My Games-biblioteksiden (/mygames)
- For MyFavourites, lag funksjonalitet som gjør følgende:
    - På visning av ett spill (, ha en knapp "Legg til favoritter".
    - Klikk på knappen "Legg til favoritter" skal lagre spillet i en array i en state kalt favourites.
    - MyFavourites skal hente/vise spill fra favourites-staten.

For karakter C og høyere forventer vi at studentene selv tar ansvar for refaktorering (bruk av komponenter og props på en hensiktsmessig måte).

OBS: Husk en attribution-link til APIet skal ligge i footer-seksjonen av applikasjonen (krav for bruk av API-et).

HTML

- Vi forventer at applikasjonen designes responsivt med CSS ved hjelp av SASS (for enklest mulig integrasjon av sass, se Adding a sass stylesheet fra create-react-app.dev

- Links to an external site.)

Karakter B-krav

For å oppnå karakter B (og A) skal deler av innholdet styres gjennom Sanity.

- Opprett et nytt Sanity-prosjekt, og koble dette til React-applikasjonen.
- Lag datamodell og -innhold for
    - spill (må kunne lagre en tittel, slug, API-id, timer spilt, sjangere og om det ligger i favorittlisten)
    - sjanger
- Legg inn minimum 10 spill i Sanity. Disse utgjør basen for "My Games"-biblioteket i Appen. Bruk feltet API-id for å lagre spillets id (eksempel; i APIet har spillet "Pocket City 2" id-en 957651).
- I dashboard, legg til opptelling av antall spill for My Games-biblioteket og Favourites. Se skissen dashboard_gradeAB.png i GitHub-repositoriet. Tips: GROQ-metoden *count*.
- På en side for ett spill, bruk feltet API-id for å hente ekstra informasjon (i tillegg til informasjon som ligger i Sanity) som skal vises om spillet. Ekstra informasjon er
    - Bilde
    - Rating
    - Oppsummering/plot
    - Stikkord (tags)
    - Utviklere (developers)
    - Utgiver (publisher)

    - Utgivelsesår (release)
    - Plattformer (platforms)
    - Kjøpsmuligheter (stores)

Karakter A-krav

For A-krav gjelder også alle kravene nevnt under B-krav.

- Lag en innholdstype "Bruker" i Sanity.
- Lag innlogging som sjekker bruker mot Sanity og logger inn denne. Sjekk på e-postadresse er tilstrekkelig (ikke krav til passord). Bruk en state eller localStorage.
- Alt spillinnhold må kunne knyttes til brukerkontoen. Applikasjonen skal kun hente innhold knyttet til den innloggede brukeren.
- Brukere skal kunne ha ulike favoritter. Datamodellen må modifiseres slik at favoritter lagres på bruker, ikke på spill.
- På spillsiden, bruk npm-pakken react-tagcloud

- Links to an external site. sammen med stikkord (tags) og gamescount-nøkkelen fra APIet for å lage en ordsky av stikkord.

Huskeliste til innlevering

- Alle hjelpemidler er tillatt under utviklingen.
- Sørg for at alle pakker dere har brukt (eksempelvis react-router-dom) ligger i devDependencies i package.json. For å sikre dette allerede fra dere installerer dem, bruk flagget --save-dev. Eksempel:

  ```
  npm install --save-dev react-router-dom
  ```

- Attribution-link til APIet skal ligge i footer-seksjonen av applikasjonen
- Gjør GitHub-repositoriet public! Sett sensorene (brukernavn *toremake* og *ackarlse* som collaborators)
- Husk å merge alle branches og all kode inn til main-branchen før innlevering!
- For dere som velger A/B-kravene: Legg sensorene til som administratorer på Sanity-databasen (inviter marius.akerbak@gmail.com og ann.c.karlsen@hiof.no).
- Et kort dokument (i rot-mappen i repositoriet med navnet "eksamensdokument_gruppeX") som inneholder
    - Hvilken vanskelighetsgrad gruppen har gått for (minstekrav, D, C, B eller A)
    - Hvilken student som tilhører hvilket GitHub-alias (eks: toremake = Tore Marius Akerbæk)
    - Redegjørelser/forutsetninger dere gjør dersom oppgaveteksten oppleves uklar eller ikke setter klare rammer eller begrensninger (dere kan ta egne, velbegrunnede valg!)
    - Redegjøre for potensielle utfordringer som ikke rakk/ble klart løst
    - Kilder til ALL dokumentasjon dere har benyttet under arbeidet med eksamen
- Kommenter kode som avviker fra syntax/metodikk undervist i kurset, og spesielt der dere har benyttet ekstern hjelp (StackOverflow, YouTube, blogger etc.), inkludert link til kilden.

Innlevering

Innlevering er en link til GitHub-repositoriet som skal leveres i Inspera. Merk - ingenting skal leveres i Canvas for eksamensoppgaven.

Vurdering

- En god mappestruktur med fornuftige mappe- og filinndeling og -navngiving vil telle positivt i vurderingen. Fjern unødvendige filer (du trenger strengt tatt bare index.js og app.js + minimum en css/sass-fil). PS: ved opprydning, husk å fjerne importer fra index.js og app.js.
- I tillegg til kravene beskrevet til funksjonalitet og teknologi i denne oppgaveteksten, vil gode prinsipper for webutvikling telle positivt (semantisk HTML, god bruk av CSS, universell utforming, responsivitet).
- Bruk av branches og commits vil vurderes.
- Implementering av Layout (ikke et krav) i routingen vurderes positivt.
- Refaktorering (fornuftig bruk/gjenbruk av komponenter og props) er en vesentlig del av vurderingen.
- Feilmeldinger (error og warnings) bør på det sterkeste unngås. Rydd opp i koden før innlevering.

Del 2: Skriftlig eksamen

Skriftlig flervalgs-eksamen som varer en time, det blir valgt ut 45 tilfeldige spørsmål fra en spørsmålsbank. Spørsmålene og svaralternativene og hva som er riktig svar er tilgjengelig for sensor i eksamenssystemet Insperia.

Det blir gitt et poeng per riktig svar, det blir ikke gitt poeng ved feil svar eller ingen svar. Det er ikke minuspoeng for feil svar