Eksamensdokument for Gruppe 2

Innholdsfortegnelse:

Medlemmer med alias:

Navn: Emil Berglund — **GitHub Alias**: EmilB04

Navn: Andreas B. Olaussen — **GitHub Alias**: Andolaus

Navn: Sebastian W. Thomsen — **GitHub Alias**: Thomsen97

Navn: Ida K. Tollaksen — **GitHub Alias**: Idatol

Lenker og invitasjon:

GitHub Lenke: https://github.com/EmilB04/UIN2025_eksamen_gruppe2

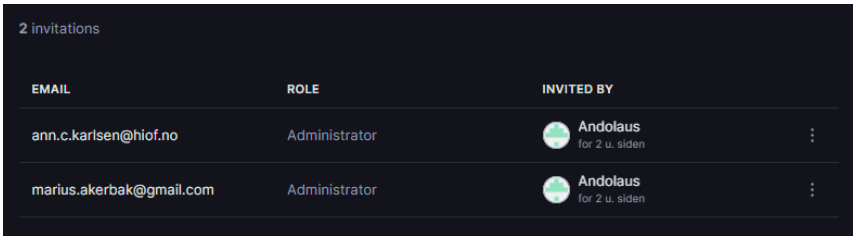
Sanity organization:

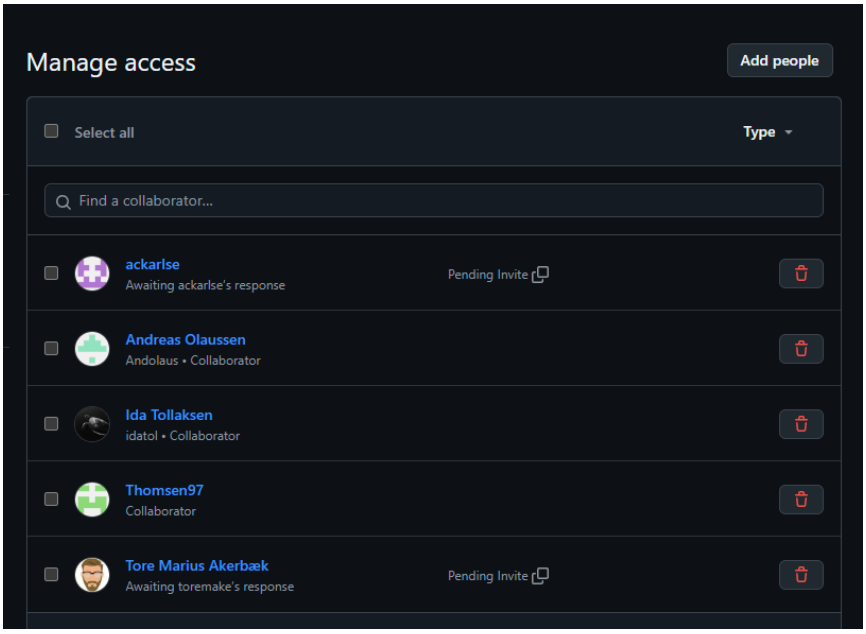
https://www.sanity.io/organizations/oa4H42Px9/project/zfrfh6h7?orgId=oa4H42Px9

CloudFlare Live Server: https://uin2025-eksamen-gruppe2.pages.dev/

Ann-Charlott og Marius har begge blitt invitert til GitHub og Sanity-organisasjonen med følgende informasjon.

Navn	E-post	GitHub Alias
Ann-Charlott Beatrice Amundsen Karlsen	ann.c.karlsen@hiof.no	ackarlse
Tore Marius Akerbæk	marius.akerbak@gmail.com	toremake





Mappestruktur front-end:

Billettlyst/

src/

api/

- Filer relatert til Ticketmaster API-et

assets/

- Ekstra ressurser, som bilder og figurer

componentes/

- Ulike brukbare komponenter

pages/

- Ulike navigerbare sider

sanity/

- Fetch funksjoner mot sanity

styles/

- Stil for komponenter og sider.

App.jsx

main.jsx

Innlogging på Min side:

Følgende informasjon finner dere også i Sanity back-end, men for enklere tilgang og sjekking er det listet opp ulike innlogginger med forberedt innhold.

Person	***E-post***	***Passord***
Emil Berglund	emilbe@hiof.no	Test1234
Ida Tollaksen	idakto@hiof.no	Test12345
Sebastian Thomsen	sebastwt@hiof.no	Test1234
Andreas Olaussen	andreabo@hiof.no	#MariusOgAnnCharlottErBest123#

Vanskelighetsgrad:

Hele gruppen syntes front-end development er særdeles gøy å holde på med, og vi ble fort enige om å sikte høyt på denne innleveringen. Ved at alle i gruppa oppnådde en høy karakter i Webutvikling og syntes konseptet med et dynamisk nettsted var gøy, siktet vi fra start på karakter A eller B.

Ved å jobbe med oppgaven jevnt og trutt, endret heller ikke synet vårt seg noe angående valgt nivå. Vi opplevde i stor grad at vi fikk til det som krevdes av de ulike nivåene, og at vi klarte å gjenskape demo-videoen slik den ble presentert.

Totalt sett har vi i Gruppe 2 derfor gått for karakter A eller B.

Redegjørelse for forutsetninger:

Når det kommer til prosjektet vårt, har vi gjort flere redegjørelser og valg underveis basert på hvordan vi tolker oppgaveteksten. Under vil dere kunne lese hva vi har tolket og gjort, i tillegg forstå dere på eventuelle fravik/avvik.

Min side/Dashboard:

Avvik fra oppgavetekst

Under utviklingen av Dashboard har de ulike karakterkravene fungert som en sjekkliste for måten vi har arbeidet på og jobbet oss «oppover». Det har derimot vært ting på denne «listen» som vi har vært uenige i når det kommer til helhetlig løsning og resultat.

Under karakter D står det følgende: *EventCard - En komponent som presenterer et enkelt arrangement i kortformat. Denne skal brukes der arrangementer listes opp, f.eks. i: Home, Dashboard og CategoryPage.*

«EventCard» er bevisst ikke brukt på Dashboard da vi mener denne ikke passer inn på en *Min Side*-oversikt med tanke på hvordan vi har utført dashboard etter senere karakterkrav. *Event Card* brukes andre steder på nettstedet vårt og inneholder blant annet et bilde, tittel, sted og dato. Hvis dette skulle inne på en oversiktsside som

Dashboard er det vår mening at hvert tidligere kjøp -og ønskelisteelement ville tatt unødvendig mye plass.

Vi har derfor valgt å presentere disse elementene på listeform, for en oversiktlig og lett utvidbar måte. Ved å lage det på listeform kan man utvide løsningen ved å begrense antall elementer på listen, før man må «bla» til neste side, for eksempel.

Videre har vi også gått for en løsning med brukerverifisering, hvor innholdet som vises kun skal være relevant til innlogget bruker. Vi opplevde her at vår fremgangsmåte ble enklere å gjennomføre.

Selv om vi tok dette valget, føler vi at vi har fått vist kompetanse innen bruk av komponenter og props ellers, og håper vårt synspunkt også gir litt mening i satt kontekst.

Autentisering og brukerdata

- Brukeren må være innlogget for å kunne få tilgang til «Min side»
- Er man logget inn, får man tilgang til informasjon om innlogget bruker, og elementer som ønskeliste og tidligere kjøp.
- Er man ikke logget inn, blir man ført til en innloggingsside med funksjonell autentisering, og får ikke sett noe av de overnevnte elementene.
- Informasjon om brukeren (inkludert ønskeliste og tidligere kjøp) hentes fra Sanity.
- Detaljer for ønskeliste-elementer og tidligere kjøp hentes fra Ticketmaster sitt API.
- Innloggingsstatus og id lagres i LocalStorage ved innlogging, og slettes ved utlogging.
- Hver bruker har følgende relevante felt lagret, i tillegg til personlig informasjon som navn, alder og bilde:
    - wishlist: en liste med referanser til event-objekter.
    - previousPurchases: en liste med referanser til event-objekter.
    - Friends: en liste med referanser til andre brukere i systemet.

Venner og felles arrangementer

- Brukere i Sanity har et eget felt med referanser til alle sine venner.
- Venner til innlogget bruker listet opp på «Min side»

    - Vennelisten viser opptil 3 felles arrangementer per venn for enkel oversiktlighet, og opplyser om dette med en tekststreng.
    - Hvis en venn ikke har noen arrangementer til felles, vises også dette med en tekststreng.
- Trykker man seg inn på et element i ønskelisten eller tidligere kjøp, får man først og fremst en detaljert oversikt over arrangementet hvor data hentes direkte fra Ticketmaster. Videre kan man også se hvilke personer som har dette på ønskelisten sin, eller hvem som har kjøpt billetter tidligere.

Struktur på event-data

- 10 event-objekter lagres i Sanity med følgende felt:
    - API ID (referanse til Ticketmaster)
    - Bilde (Ikke nødvendig/funksjonelt, men for bedre oversiktlighet i Sanity)
    - Tittel
- Disse brukes som objekter i ønskeliste og tidligere kjøp, og kobles opp til brukere i Sanity ved bruk av referanser.

- Detaljene til disse ulike eventene hentes med API-kall fra Ticketmaster.
- Ved API-kall skal blant annet følgende hentes ut:
    - API-ID (Ikke nødvendig)
    - Tittel/navn
    - Dato
    - Sted
    - Tidspunkt
    - Sjanger
    - Bilde/Poster

Henting av data

- Måten vi henter ut detaljer og info er som følger:
    - Vi går gjennom relevante eventer i Sanity og henter ut det første feltet til hver event, som er API ID.
    - Videre bruker vi en metode for å søke etter TicketMaster eventer ved å bruke API ID som ble hentet fra Sanity.

    - Til slutt returnerer vi funnene.
    - Herifra kan vi hente ut all informasjon som er tilbudt av API-et.
- Det benyttes en hjelpefunksjon getApiIdBySanityId(id) for å bruke Sanity *_id* til å hente Ticketmaster sin event-ID.
- Deretter brukes getEventById(apiId) for å hente ulik informasjon fra Ticketmaster som nevnt over.
- For å unngå CORS-feil og API-throttling (HTTP 429), hentes dataene sekvensielt med for…of og await.

Brukeropplevelse og fallback

- Ettersom mesteparten av data hentes fra API, kan lastetiden være ulik/ujevn. Det er derfor implementert en «loading»-indikator mens dataene laster.
- Dersom et event ikke finnes eller feiler under lasting, håndteres dette med en feilmelding i konsollen, men resten av eventene lastes fortsatt.
- Hvis en bruker ikke har noen events i ønskelisten eller kjøpshistorikken, vises passende tomtilstand-meldinger.
- Siden er responsiv og fungerer på både mobil, nettbrett og pc.
    - Gjøres med media-queries og utnyttelse av flex-box

EventCard og FestivalPassCard

I karakter C-kravet står det at man skal "tilpasse EventCard eller lage et tilsvarende komponent" for å vise arrangementer fra en storby. Der valgte vi å tolke dette som at det var tillatt å lage én gjenbrukbar komponent som kunne brukes flere steder, så lenge den inneholdt de elementene som ble beskrevet: navn, bilde, by og land og dato. Vår EventCard ble derfor laget som et fleksibelt komponent. Dette så vi på som det mest intuitive og beskrivende måten å løse det på, og vi mener det ga oss en renere og mer gjenbrukbar kodebase.

Når det kommer til visning av festivalpass på EventPage (som beskrevet under karakter krav C, EventPage), står det at man skal "bruke EventCard til å liste ut tilgjengelige festivalpass". Her tok vi et bevisst valg om å heller lage et eget komponent, FestivalPassCard, fordi denne delen trengte egne knapper for kjøp og ønskeliste, og ikke

bare informasjon. Vi mener det ville vært mer klønete å bruke EventCard her og prøve å tilpasse det med ekstra logikk. I tillegg valgte vi det vi opplevde som den mest intuitive og ryddige løsningen.

Med andre ord, vi har brukt EventCard som det "tilsvarende komponentet" der det var naturlig og brukte FestivalPassCard på EventPage altså på siden med de ulike festivalpassene. Vi mener dette har gjort koden mer strukturert og oversiktlig, og for oss var det en bedre og mer forståelig måte å løse kravet på enn å bruke samme komponent til alt.

Redegjørelse for mulige utfordringer:

Min side/Dashboard:

Et problem jeg (Emil) møtte på under utviklingen av dashboard var å få opp passende tekst/innhold når man trykket inn på et ønskeliste-element eller et tidligere kjøp. Selv om vist informasjon/detaljer om eventet skulle være likt på begge sidene, skulle teksten som informerte hvilken venner som tidligere har kjøpt eventet, eller har elementet på ønskelisten være forskjellig.

Jeg løste dette problemet ved å sende med en prop, «pagetype» som sier om elementet er «wishlist» eller «previous-purchase» og viser passende tekst deretter i venne-seksjonen. Prop-typen baseres på om valgt event ligger i ønskeliste -eller tidligere kjøp-seksjonen på dashboard-siden.

Videre støtte jeg på et problem da jeg skulle laste inn informasjon om ulike eventer fra API-et. Jeg fikk støtt og stadig feilen «429» som betydde at jeg overbelastet API-grensen ved å kalle for mange ganger på API-et. Dette løste jeg ved å dele opp API-kallene i flere, men mindre kall, slik at det ble fordelt utover, som igjen ledet til mindre overbelastning.

Events:

Et problem jeg (Ida) møtte på under utviklingen av EventCard komponentet var å hente ut og vise data fra Ticketmaster API-et. Det å jobbe mot et API på denne måten var veldig nytt for meg, og det krevde at jeg satte meg godt inn i dokumentasjonen til API-et.

Jeg løste dette nettopp ved å sette meg godt inn i dokumentasjonen og i tillegg eksperimentere med ulike spørringer mot API-et. For å forstå hva som faktisk ble hentet ut brukte jeg mye console.log mye. Dette ga meg en god oversikt i terminalen over hvilke data som ble hentet ut fra API-et. På denne måten klarte jeg å hente ut riktig informasjon på riktig sted når det kom til byggingen av EventCard komponentet.

CategoryPage:

Et problem vi (Ida og Sebastian) støtte på under utviklingen av filtreringsfunksjonaliteten i CategoryPage var at det ikke ble returnert noen events i København, selv om vi visste at det var events der. Til å begynne med trodde vi at det skyldtes ø-en i «København», men etter å ha undersøkt med hjelp av console.log fant vi ut at det ikke var dette som var problemet.

Det viste seg at Ticketmaster registrerer København som ulike bydeler: København S, København N, København V og København K. Dette gjorde at filtrering kun på «København» ikke ga noen treff. Vi løste dette ved å hardkode filtrering lokalt, der vi sjekker om bynavnet til eventene inkluderer «København» uavhengig av sone. På denne måten fikk vi en robust filtrering som fungerer godt for visning av events for hele København.

Løsningen med å hardkode filtreringen for København er ikke den mest fleksible, men fungerte godt for vårt behov. Vi opplevde kun dette problemet med København, Oslo og Stockholm fungerte som forventet. Vi valgte derfor å håndtere dette lokalt, spesielt fordi København vises i demo-videoen. Løsningen fungerer fint nå, men kan videreutvikles for bedre gjenbrukbarhet.

Et annet problem som oppsto når jeg (Sebastian) holdt på med kategorisidene var angående ønskeliste-knappen. Sånn den først var implementert var at den kun ble laget ved hjelp av en useState, men da fungerte den kun så lenge man ikke refreshet nettsiden, da ble den nullstilt. Ettersom jeg var usikker på om dette var innenfor kravene, så valgte jeg å lagre det i localStorage, slik at eventene man hadde lagt i ønskelisten forble dersom man skulle refreshe nettsiden. Men da dukket det opp et nytt problem. Nå ble den lagret på tvers av brukere. Så dersom man la til noe i ønskelisten uten å være innlogget eller på for eksempel Emil sin profil, så ville den også vise at den var i ønskelisten dersom man logget ut eller logget inn på en annen profil.

For å fikse dette tenkte jeg at man kunne legge til at den også var knyttet til en bruker. Etter å ha prøvd dette uten hell, spurte jeg ChatGPT, og fikk da en fungerende ønskeliste-knapp som lagrer det i localStorage basert på hvilken bruker man er på, og dersom man ikke er logget inn blir det lagret i en gjesteønskeliste.

Det siste problemet jeg (Sebastian) hadde var angående henting av data. Det sto i oppgaveteksten at et godt utgangspunkt for henting var find/suggest endepunktet i API-et. Der fant jeg ut at dersom jeg brukte dette, fikk jeg kun hentet 5 events, og det var ikke støtte til filtrering av f.eks dato. Derfor har jeg gått for løsningen at ved første innlasting når man velger en kategori så brukes suggest, og når man søker eller filtrerer så hentes events rett fra events endepunktet, og attraksjoner og spillesteder blir hentet videre med embedded fra disse eventene. Da vil attraksjoner og spillesteder matche de eventene man får opp.

HomePage:

En utfordring som jeg (Andreas) hadde under denne oppgaven var API-kallet knyttet til festivalene på HomePage. Jeg brukte først kun /events endepunktet for å hente festivaldata, som førte til at jeg fikk festivalpass istedenfor selve attraksjonen. Jeg så ikke at dette var feil før vi hadde kommet ganske langt ut i prosjektet. Jeg så at navnet til eventet var feil i forhold til demovideoen og jeg begynte å lage egne metoder i HomePage for å kutte teksten på overskriftene for å få det til å matche demovideoen. Jeg skjønte fort at det kanskje var noe feil med apikallet og jeg leste mer i dokumetnasjonen til

ticketmaster apiet, samt brukte api exploreren flittig for å finne ut hva som var riktig.

Jeg kom etter hvert frem til at jeg måtte bruke /attractions-endepunktet i stedet for å hente arrangementet for å få festivalene riktig. Da jeg gjorde dette oppstod det et større problem hvor vi ikke lenger kunne komme inn på EventPage, hvor jeg brukte mye tid på å finne ut av hvordan jeg kunne gjøre to apikall i samme metode. Da jeg løste dette, fungerte det å komme inn på EventPage igjen.

Arbeidsmetodikk og fordeling:

Oppstart:

Når det kom til å sette opp prosjektet vårt, ble det gjort av Emil. Han satte opp et GitHub repository og Sanity organization og inviterte deretter resten av gruppa. Emil satte videre opp et tomt React prosjekt, installerte avhengigheter og la til grunnleggende struktur slik at hele gruppa kunne komme i gang.

Utvikling og GitHub struktur:

Videre har vi hovedsakelig arbeidet på forskjellige områder på nettstedet vårt, og bestemte oss kjapt for å benytte forskjellige branches. Under finner dere en grov oversikt over hvem som har gjort hva.

Vi har etter beste evne prøvd å utnytte GitHub for hva det er verdt, med å ha gode og strukturerte commit-meldinger, benytte oss av branches for å unngå å overlappe og overskrive hverandre. I tillegg har vi benyttet oss av pull-requests hvor vi flere ganger har bedt hverandre å se over hverandres endringer, for å forsikre oss om at det vi merger er noe alle er enige i.

Oppsettet vårt i Github består av en main branch, en develop-branch og eventuelle undergrener/arbeidsgrener. Disse arbeidsgrenene baserer seg på develop, og henter dermed data derifra.

Vi har valgt å benytte main som en innleveringsbranch, så ingenting har blitt endret der direkte utenom readme-filen.

Valget bak egne branches til hver oppgave eller hvert område kommer av oversiktlighet. Ved å gjøre det sånn, vet man alltid hvem som gjør hva, og man slipper som sagt å tenke på å overskrive andres arbeid.

Grov arbeidsfordeling:

HVEM	HOVEDOMRÅDE(R)	TILLEGGSOMRÅDE
EMIL	Min side/Dashboard	Navbar
ANDREAS	HomePage/EventPage	ArtistCard
SEBASTIAN	CategoryPage	Event filtrering
IDA	EventPage/HomePage	Event filtrering

Bruk av KI:

Det er brukt KI for å løse/forbedre deler av oppgaven vår.

KI har hovedsakelig blitt brukt til å forbedre noe vi allerede har implementert, men også til å lage metoder for oss. Metodene KI står 100% for, er derimot metoder som ikke er avgjørende for funksjonalitet, men gjør totalopplevelsen bedre.

Eksempelvis har Github Copilot laget metoder for å formatere tid og dato fra engelsk format til norsk. Dette forbedrer lesbarheten og fjerner unødvendig forvirring som kan oppstå ved uvant format.

Videre har KI blitt brukt til å forbedre API fetching, ved å foreslå metoder for å unngå API-overbelastning. Her skrev den ikke koden for oss, men viste et eksempel på hvordan det kunne gjøres. Derifra kunne vi implementere en liknende løsning i vårt prosjekt.

I tilfellene hvor KI er brukt, er dette dokumentert direkte i koden, med prompt og svar/resultat.

Kilder:

Font Awesome. (2025). *Classic*. Font Awesome. https://fontawesome.com/icons/packs/classic

Itamar Haim. (2025, 23. feb). *What Does The rel=”noopener noreferrer” Tag Mean?* Elementor. https://elementor.com/blog/noopener-noreferrer/

Sanity. (2018, 10. jan). *List Previews*. Sanity. *https://www.sanity.io/docs/studio/previews-list-views*

Mozilla. (u.å.). *Spread syntax (...)*. MDN Web Docs. Hentet 15.mai. 2025 fra https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax

Webtricks LMS. (2025). *Utvikling av interaktive nettsteder*. Webtricks.blog. https://lms.webtricks.blog/kurs/uin

W3Schools.com. (2025). *How TO - Custom Scrollbar*. W3schools.com. https://www.w3schools.com/howto/howto_css_custom_scrollbar.asp