Eksamensdokument for Gruppe 2

Innhold

[Medlemmer med alias: 1](#_Toc198493415)

[Vanskelighetsgrad: 2](#_Toc198493416)

[Redegjørelse for forutsetninger: 2](#_Toc198493417)

[Redegjørelse for mulige utfordringer: 2](#_Toc198493418)

[Min side/Dashboard: 2](#_Toc198493419)

[Kategorisider (Sebastian): 3](#_Toc198493420)

[Arbeidsmetodikk og fordeling: 3](#_Toc198493421)

[Oppstart: 3](#_Toc198493422)

[Utvikling: 4](#_Toc198493423)

[Grov arbeidsfordeling: 4](#_Toc198493424)

[Bruk av KI: 5](#_Toc198493425)

[Kilder: 5](#_Toc198493426)

# 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

**GitHub Lenke:** <https://github.com/EmilB04/UIN2025_eksamen_gruppe2>

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

# 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 syntes 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:

# 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.

* **Måtte refreshe siden for å oppdatere Logg inn og Min side.**

## Kategorisider (Sebastian):

Et av problemene som oppstå når jeg 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.

Et annet problem var angående filtrering av byer. Slik vi hadde satt opp filtreringen fungerte alt som det skulle, unntatt filtrering på København. Jeg fant da ut at dette var på grunn av i API-et så var København delt inn i forskjellige bydeler, som København S, København N, osv... Dette fikk jeg ikke til å fikse selv, så her måtte jeg også spørre ChatGPT om hvordan filtreringen vi har implementert kan hente og ta i bruk underkategoriene av København dersom brukeren velger å filtrere på København. Dette fungerer nå slik vi har sett for oss oppgaven ønsker det.

Det siste problemet jeg 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.

# 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:

Videre har vi hovedsakelig arbeidet på forskjellige områder på nettstedet vårt, og bestemte oss kjapt for å benytte forskjellige branches. Under finner du 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 | - | - |
| Sebastian | Kategorisidene | Henting av filtrerte data |
| Ida | - | - |

# Bruk av KI:

Det er brukt KI for å løse/forbedre deler av oppgaven vår. I tilfellene hvor KI er brukt, er dette dokumentert direkte i koden, med prompt og svar/resultat.

# Kilder:

Font Awesome. (2025). *Font Awesome*. Fontawesome.com; Font Awesome. <https://fontawesome.com/icons/packs/classic>

**‌**

**NB: Husk å kommentere kode som avviker fra kurset, eller ved bruk av KI.**