Dokumentasjon

Webapplikasjon

Gruppe 10

Mahamoud Ibrahim, Mariam, Jørgen, Shvan, Bardh Arifi

Gruppe

Levere samlet som gruppe. Dere må derfor være enige om det som skrives her.

Hvordan ble oppgavene fordelt?

Fordel 0-3 poeng per rute. Ingen skal ha 0 og alle kan ikke ha 3 over alt.

0 = Ingen
1 = Minimalt
2 = Moderat
3 = Mest

	Prosjekt*	Backend	Frontend	Testing	Annet**
Mahamoud Ibrahim	2	3	2	1	1
Maream	2	3	2	1	1
Jørgen	2	2	2	3	1
Shvan	3	2	2	1	1
Bardh Arifi	3	1	3	1	3

** Prosjektkolonnen er knyttet til prosjektstyring (Trello, GitHub, Discord m.m).*
*** Hvis annet spesifiser hva*

Annet her betyr “Design”

Hvordan har samarbeidet fungert?

Vi føler at samarbeidet har fungert veldig bra. Alle gruppemedlemmer har hatt god initiativ, jobbet sammen og møtt opp felles for å jobbe sammen flittig gjennom eksamensperioden. Vi tok hensyn til forskjellige timeplaner, da flere av gruppemedlemmene hadde flere eksamener under mappeinnleveringseksamensperioden.

Hvilke antagelser gjorde dere i oppgaven?

Oppgave 2:

- Kravet: "*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.".* Antagelsen vi tok for å fylle dette kravet er at en bruker ikke skal kunne oppdatere dagene som en mal er opprettet med. Vi lar derfor ikke bruker oppdatere dette feltet, selv når ingen arrangementer bruker malen.

- I oppgaven antok vi først at en "participant" representerte en individuell person, og designet tabellen deretter. Senere endret vi denne antagelsen til at en "participant" skulle representere en individuell bestilling som kunne inkludere flere deltakere. Derfor oppdaterte vi tabellen med et participants-felt for å håndtere dette. Vi vurderte også en mer kompleks løsning med en egen user-tabell og mange-til-mange-relasjoner, men valgte bort dette grunnet tidsbegrensninger og manglende krav.

Hvilkes deler av oppgaven trenger utvidet forklaring?

Oppgave 1:

- Vi hadde vanskeligheter med fordeling av komponenter i oppgave 1. For eksempel: Vi var usikre på om det kun var nødvendig å separere komponentene som var i alt.js, eller om vi skulle refaktorere og bygge egne komponenter med samme funksjonalitet. Denne usikkerheten førte til ekstra og unødvendig tidsbruk. Hadde vært bedre om dette ble oppklart fra starten av istedenfor gjennom FAQ i senere tid.

Hvilke problemer oppstod underveis og hvordan ble de løst?

- I oppgave 1 så skulle vi bruke kode fra All.js og rydde det opp, men det var vanskeligere enn det vi har forventet, spesielt når vi skal overføre det til typescript (siden det er en javascript fil).
- Det vanskeligste delen i oppgave 2, var å forholde oss til regler iht til opprettet mal. Vi går mye fram og tilbake, og må eventuelt være sikker på at det er validering på både frontend og backend
- Vi hadde problemer med useState under rendering. Når du gjør en endring i en useState, så vil ikke endringene være umiddelbar og det fører til problemer når man har inputs, for da så kan det hende at noen inputer blir "uncontrolled".

- Vi opplevde problemer med sammenligning av datoer fra DatePicker (showCorrectDatepicker) i forbindelse med implementeringen av regel 1 (ingen andre kan ha arrangement på samme dag). Problemet oppsto når vi forsøkte å sammenligne datoene ved å konvertere datoen fra DatePicker til ISO-string, hente kun datoen uten tidsinformasjon og deretter utføre sammenligningen. Gjennom feilsøking i konsollen oppdaget vi at DatePicker alltid valgte feil dato, nemlig én dag tidligere enn den datoen som skulle være ugyldig. For å løse dette justerte vi logikken ved å bruke: `prevDate.setDate(prevDate.getDate() + 1);` til å forskyve datoen én dag fremover. Dette viste seg å løse problemet.

Forklare ulikheter mellom wireframe og endelig innlevering.

Alle wireframe bilder finnes i mappen documentation/skisser

Navbar:

Vi valgte å ikke lage en “About Us” og “Contact”-side da dette ikke var et krav fra oppgaven.

Footer:

Vi fjernet mesteparten av det som var planlagt for footer, og erstattet det med teksten “Webapplikasjoner - Eksamen høsten 2024 - Gruppe 10”

Admin Arrangementoversikt (/admin/events) - Kalt ArrangementOversikt.jpg

Første tanke var at vi skulle sette opp en form for tabell ved å bruke table html tags. På grunn av brukervennlighet, og lettere mediaqueries, valgte vi heller å lage kort fremfor en tabell. Slik blir det lettere for nettsideeieren å navigere rundt i events med store og tydelige knapper fremfor å bruke en arrangementtabell som kan være begrenset til å se detaljer om arrangementet. En slik tabell ville også vært vanskeligere å tilpasse mobil, og er dårlig for SEO.
Fjernet filtrering når vi merket at dette ikke var et krav.

Arrangement Detaljer (/event/{eventId}) - kalt EventDetailedPage.png

Vi valgte bort fra bruk av bilder, da dette ikke var et krav i oppgavebeskrivelsen, og i tillegg ville tatt mer lagringsplass og gjort det vanskeligere å lage dummy arrangementer for testing. Vi brukte heller ikke ikoner, da vi ikke ville bruke UI-biblioteker grunnet kravet “At du kan lage komponenter uten bruk av UI-biblioteker”. Hamburgermeny og close-cross som er ment for mobilmenyen er hentet fra HTML-symboler.

Statistikk panel (/admin/statistics) - kalt StatistikkPanel.png

På wireframe tar vi hensyn til alle tre kravene, men da oppgaven kun spør om at ett av de tre statistikk-kravene skal være fylt, har vi valgt å redesigne denne siden til å inneholde en tabell med alle events og antallet som er meldt opp og antallet som er på venteliste.

ChatGPT logger:

ChatGPT ble brukt til å få hjelpefunksjonen “showCorrectDatepicker” fra eventUtils.tsx til å fungere sammen med AdminEvents.tsx. Både funksjonen og AdminEvents.tsx har blitt videreutviklet i etterkant.
https://chatgpt.com/share/674e0e5a-ca10-8013-bd70-4e7134e0c647

Link til github-repo:
https://github.com/Webapp-Eksamen-Gruppe10/Webapp-2024-eksamen-Gruppe10

Informasjon til sensor:

Installasjon av pnpm avhengigheter

Vi bruker pnpm for å installere avhengigheter.
For å installere alle avhengigheter må du navigere til

oppgave_1/backend
oppgave_1/frontend
oppgave_2/backend
oppgave_2/frontend

og skrive `pnpm i` i terminal for hver av disse mappene.

Installasjon og kjøring av playwright

Naviger til oppgave_1/frontend og kjør følgende i terminal (etter pnpm i er brukt)

For å installere avhengigheter:

```
pnpm exec playwright install
```

For å kjøre alle tester:

```
pnpm exec playwright test
```

.env fil:

*gjelder både for oppgave 1 og oppgave 2:

Plasser en fil med navn .env i rotmappen til backend. Denne skal inneholde kun denne linjen:

DATABASE_URL="file:./dev.db"

Generere prisma database

For oppgave 1 må du navigere til folder oppgave_1/backend i terminal
For oppgave 2 må du navigere til folder oppgave_2/backend i terminal

Skriv følgende kommandoer i terminalen for å generere prisma databasen:

pnpm prisma generate
pnpm prisma:migrate (NB Om den spør etter navn eller lignende, bare trykk enter)
pnpm prisma:seed