GREENMARK

Group: Lasse & Co.

Magnus, magn552n@stud.kea.dk

Lasse, lass417r@stud.kea.dk

Sider: 41

Anslag: 43345

Links:

Github: https://github.com/magnusdalkvist/exam-festival

Website: https://greenmark.netlify.app/

Screencast (Lasse): https://youtu.be/mQRO6tEh4FY

Screencast (Magnus): https://youtu.be/NArS8gonKA0

INDHOLDSFORTEGNELSE

INTRODUKTION

INDLEDNING

“In fact, according to The Show Must Go On, a report by festival industry steering group Powerful Thinking, the UK music camping festival industry emits 24,261 tons of CO2 per year. That’s equivalent to greenhouse gas emissions from over 60 million miles driven by an average gasoline-powered passenger vehicle. Now imagine what that looks like if you include festivals in the rest of the world too.” (Kilde: GLOBAL CITIZEN - Tess Lowery, 2022)

Greenmark er en festival der i den grad vægter miljøet højt. Festivalen står inde for den grønne omstilling og forsøger at være et positivt pust i en meget miljøbelastende branche. Festivalen forsøger i høj grad at minimere deres negative miljøbidrag både fysisk, men også online! Vi blev bedt om at udvikle to nye digitale webløsninger til deres kommende festival i 2023.

Det ene del af webløsningen skal bestå af et komplet billet booking system, som skal gøre processen ved køb af billetter nem, overskuelig og så miljøbevidst som mulig.

Den anden del af webløsningen skulle bestå af en flot oversigt over de musikalske optrædener på festivalen. Den skulle igen også være nem, overskuelig og let tilgængelig for brugerne.

KONCEPT

Der var forskellige krav for opgaven samt en backend som man skulle gøre brug af. Dataen i backenden lægger meget op til en klassisk rock festival, men da ingen i gruppen har større interesse for genren rock, valgte vi at ændre retning. Baggrunden for vores koncept er en kombination af ønsket om forandring i nutidens online tilstedeværelse for festivaler og for at skille sig ud. Tankerne om at lave en hjemmeside der er let, flot og alsidig, men samtidigt har de klassiske funktioner festival hjemmesider har. Vi har valgt at kombinere de to systemer til en samlet webløsning, med en genkendelig overordnet struktur. Dette har vi valgt, fordi det giver bedre mening for brugervenligheden og strukturen for systemerne. Greenmark vægter genkendelighed, miljøbevidsthed og at have det sjovt.

MÅLGRUPPE

Løsningen er tiltænkt som et moderne og mere miljøbevidst alternativ til festivaler som Roskilde Festival og Smukfest. Festivalen skal være et sjovt, friskt og inkluderende arrangement, som er mangfoldigt. Festivalens primære målgruppe har en gennemsnitsalder på 25 år og er 50-50% mænd og kvinder. Vores primære målgruppe kommer fra omkring 55 forskellige lande verden over. Den sekundære målgruppe ses som en kombination af musikelskende familier og den ældre generation.

Vores koncept appellerer til en bred geografisk målgruppe. Konceptet appellerer i højere grad til personer som søger gode oplevelser, men ikke ønsker at det er på bekostning af miljøet.

CONZOOM

Vi har anvendt Conzoom for at støtte vores målgruppe i Danmark. Conzoom er et demografisk værktøj, der bruger kvantitative data til at segmentere den danske befolkning. Det giver os indsigt i forbrugsvaner, indkomst og geografisk placering.

E - Urban mangfoldighed

Den primære målgruppe er bymennesket (Conzoom: E, Urban mangfoldighed) der dyrker kulturlivet og skaber liv i byen. De er cyklister og brugere af kollektiv transport og har en klimavenlig tilgang til livet. Derudover er de storforbrugere af kulturtilbud i ind- og udland.

H - Unge på vej

Den sekundære målgruppe er unge i byen (Conzoom: H, Unge på vej). Dette omfatter singler under 35 år uden børn. De er meget aktive på sociale medier ligesom bymennesket, men har lav købekraft. Interessen for kvalitet er dog stadig høj hos denne målgruppe. De unge under 35 har de seneste år givet genbrug højere prioritet, når de handler, dels fordi man får et unikt produkt til en god pris, og dels fordi jagten på det unikke produkt er spændende. Denne målgruppe er meget nysgerrig og holder også meget af kulturlivet.

PROJEKTSTYRING

Vi startede projektarbejdet ved at oprette et team canvas (BILAG A) for at skabe en fælles forståelse af, hvordan projektarbejdet ville forløbe. Vi brugte også vores team canvas til at diskutere eventuelle hindringer, som vi kunne støde på undervejs, og hvordan vi ville håndtere dem. Team canvaset blev også brugt til at lave en fælles forventningsafstemning inden projektets påbegyndelse.

Det var vigtigt for os, at der var styr på vores projekt og have en god struktur. Af den grund valgte vi at gøre brug af Trello board, som værktøj til at opdele projektet i mindre delopgaver. Dermed gjorde vi projektarbejdet markant mere overskueligt og gjorde det betydeligt nemmere at danne sig et overblik over den samlede proces. Vi startede og sluttede hver arbejdsdag ved at gennemgå vores Trello board og gøre status.

For at optimere arbejdsprocessen har vi benyttet Google Docs, Figma samt Teams for at samle vores arbejde og for at kunne arbejde på tværs af dokumenter og opgaver samtidig. Under projektet har vi også brugt Github for at samarbejde om udviklingen af applikationerne.

ARBEJDSFORDELING

I et projekt af denne størrelse er det vigtigt at have et godt samarbejde for at opnå den bedste løsning og optimere arbejdsprocessen. Af den grund holdt vi et møde i starten af projektet for at dele vores styrker og svagheder og for at diskutere vores individuelle mål og ønsker for projektet. På denne måde kunne vi bedre forstå hinandens synspunkter og anvende vores forskellige kompetencer til at gavne projektet.

Vi blev begge hurtigt enige om, at vi godt kan lide at arbejde indenfor alle fagområderne, og at vores faglige niveau er tæt på det samme. Vi valgte derfor at arbejde sammen på de store beslutninger, der skulle tages i forhold til design og de lidt svære områder i kodnings delen. Da vores site er delt op i selve festivals siden og booking menuen, valgte vi at splitte det op mellem os. Magnus endte med at stå for bookingdelen hvor han sørgede for funktionalitet og design. Lasse stod for at hente dataen fra vores API og sætte siderne op, så det fik det samme look som vores mockup.

Undervejs i processen var vi gode til at informere og hjælpe hinanden, hvis vi stødte på udfordringer undervejs. Vi brugte dagligt teams til at kommunikere igennem og brugte det aktivt til at søge hjælp hos hinanden.

DESIGN PROCESS

DESK RESEARCH

For at finde inspiration til vores visuelle udtryk og design af hjemmesiden, begyndte vi med at lave desk research. Vi samlede alle vores inspirationskilder og ideer i et fælles dokument og tilføjede korte notater. Herefter så vi på de bedste ideer og inspirationskilder og udarbejdede en fælles forståelse af den grafiske retning, som vi ville arbejde os hen mod. I vores Desk research undersøgte vi også konkurrenter. Vi undersøgter mange festivalers hjemmesider og dannede os en idé om, hvad en festivals hjemmeside burde og ikke burde indeholde.

MOODBOARD

Vores 2 moodboards er meget inspireret af vores bæredygtige tilgang til opgaven. Begge moodboards består af forskellige ideer, teksturer og farver for at visualisere vores fælles tanker, ideer og følelser. Endvidere gør vi brug af vores 2 moodboards til at danne en fælles idé for stilen/udtrykket, hvilket vil give designet af websitet en rød tråd igennem sitet. Efter færdiggørelsen af vores moodboard (BILAG B), igangsatte vi arbejdet med at designe et styletile.

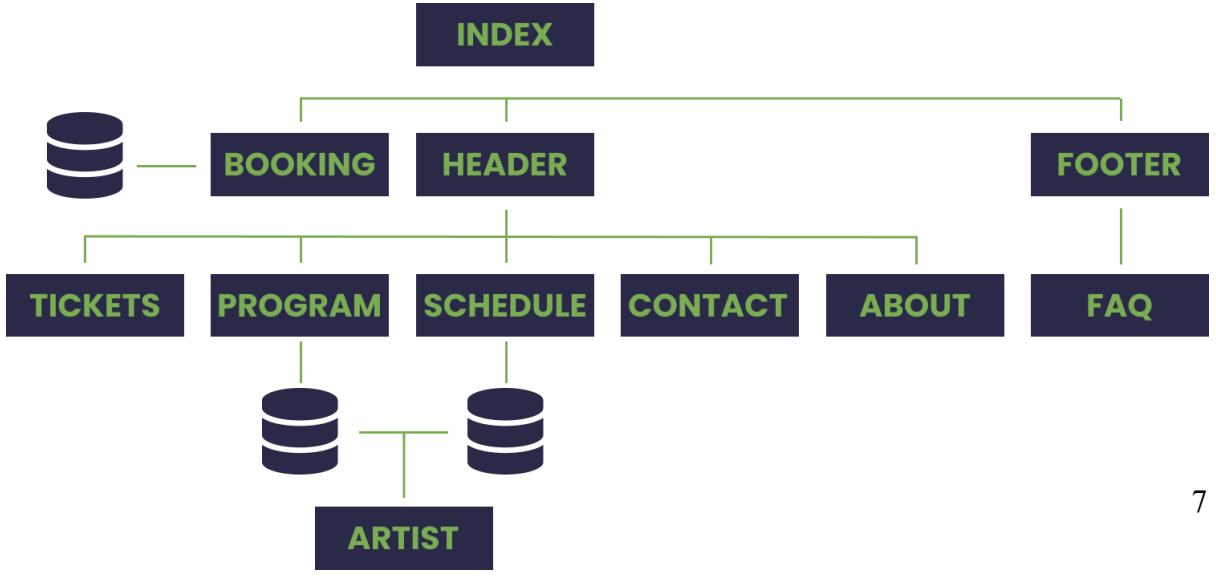
WIREFRAMES

Efter vores desk research udformede vi en forståelse for den gængese festivaldeltagers behov og forventninger til opsætningen af en festivals website. Herefter begyndte vi at kigge på, hvordan vores navigation skulle opbygges. Her er det vigtigt for os, at informationerne på websitet er kommunikeret godt, samt at der er et godt og flydende flow gennem sitet.

Efter det indledende arbejde begyndte vi at skitsere vores wireframes. Vi startede med at komme med hver vores bud på opbygningen af vores index side. Herefter satte vi os sammen og samlede de bedste ideer fra begge wireframes og rentegnede det herefter i Figma (BILAG C). Vi gentog processen igennem alle de primære sider på sitet.

INFORMATIONSARKITEKTUR

Vores tilgang til opbygning af informationsarkitektur på vores hjemmeside var ret simpel. Det var vigtigt for os at hjemmesiden udstrålede troværdighed og dette opnår vi ved en simpel og genkendelig informationsarkitektur. Vi valgte at bruge det IA pattern, der kombinerer forskellige typer patterns i én, også kaldet kombinationer. I vores tilfælde er vores informationsarkitektur opbygget af lineært, hierarki og database. Lineært er aktuelt på sider med statiske informationer om festivalen. Hierarki opstår i det vi dynamisk bygger Slugs til alle bandsene, der spiller på festivalen og databasen bruges på forskellige sider til at indhente indhold dynamisk. Herunder har vi demonstreret det ved et sitemap:



STYLETILE

Meningen med et styletile er at benytte det som en miniature designguide. Vi har aktivt benyttet vores styletile i gruppen til udarbejdelse af et design, som var strømlinet og ensformig gennem hele webløsning. Styletilen har hjulpet med at undgå forvirring i design processen. Vores styletile er bygget op af fonte, logo(navnetræk), UI elementer, farver og teksturer. Det var vigtigt for os, at vores UI elementer var ensartede samt, at de følger genkendelige konventioner. Derudover var det vigtigt for os, at vores call to action knapper var iøjnefaldende og havde tydelige hover effekter for at indikere at elementet er klikbart.

FARVER

I vores valg af farver, har vi valgt nogle farver, som udstråler bæredygtighed, og det har vi gjort for at formidle festivalens budskab og for at differentiere sig fra konkurrenter og hjælpe med at skabe en visuel brand identitet. Ved at lave en specifik farvepalette og lave brand farver forsøger vi at skabe genkendelighed og styrke brandets identiteten.

NATURE GREEN #79AC53

Vi har valgt farven Nature Green som vores primær farve. Dette har vi valgt, fordi farven afspejler festivalens grønne budskab og målsætning. Nature Green er en alsidig og velkendt farve. Verdenover symbolisere farven friskhed, vækst og balance (Kilde: Designs.AI, besøgt d. 16. december 2022). Hos Greenmark symboliserer farven miljø betænksomhed og giver en klar indikation af hvor festivalen bliver afholdt.

MIDNIGHT BLUE #37395f

Vi har valgt farven Midnight blue som sekundær farve. Dette har vi valgt, fordi farven underbygger festivalens budskab og målsætning. Festivalen foregår under åben himmel, hvilket farven understøtter. Ifølge farvepsykologi, udstråler farven pålidelighed og troværdighed. (Kilde: Canva, besøgt d. 16. december 2022). På vores hjemmeside illustrerer farven nattehimlen.

4AM BLUE #2a2a48

Vi har valgt farven 4AM Blue som sekundær farve. På vores website har vi gjort brug af farven til at farvelægge vores footer, forskellige natur silhuetter og som baggrund på vores collaps groups.

KNÆKKET HVID #F2F2F2

Vi har valgt at bruge en knækket hvid. Vi har valgt at benytte knækket hvid til vores copy på sitet. Det har vi gjort for, at det ikke skal være hårdt for brugerens øjne at læse teksterne. Vores knækkede hvid bruger vi til vores copy på de sider, der er informationstunge.

Ved benyttelse af denne farvepalette kan vi bedre kommunikere festivals miljøbevidste håndtering, samt formidle brandets værdier bedre.

FONTE

Vi valgt at bruge “web safe fonts” eller system fonte. Dette betyder, at vi fortæller sitet, at den skal bruge den font, der er installeret på det system, som sitet bliver vist på. Dette gør, at brugeren undgår at skulle downloade en font ved en HTTP request. Dette resulterer i, at vores webløsning indlæses hurtigere, grundet brugeren skal downloade en mindre pakke og er derfor mere miljøbevidst, fordi der bliver brugt mindre energi pr. load (Kilde: MightyBytes, besøgt d. 16 december 2022).

Dog kan system fonts begrænse os i at designe hjemmesiden som ønsket på enkelte systemer, men dette mener vi er ofringen værd. Vi har dog stylet overskrifterne ved at give dem en høj font-weight, gjort deres fill transparent og givet dem en stroke. Til vores brødtekst har givet dem vores knækket hvid og line-height

UI-ELEMENTER

Vores UI elementer er nøje designet efter genkendte konventioner. Vores menu er en ‘Burger’ menu, som er stylet i vores primære farve Nature green. Ved klik på vores burger menu, åbner menuen sig og ikonet ændrer sig til et X hvilket kommunikerer, at man lukker menuen der. Vores knapper er designet som 1 primær og 1 sekundær knap. De er stylet vha. vores styletile og følger Greenmarks farvepalette. Vi har også gjort brug af Collapses, hvilket er en overskuelig oversigt over flere sektioner på siderne, som man kan åbne og lukke efter behov. Det resulterer i, at brugerne nemt har et organiseret overblik over indholdet på siderne og dermed kun behøver at tilgå det indhold, der er relevant for dem. Yderlig kan det også resulterer i, at sitets load bliver nedsat grundet mængden af indhold, der skal loades med det samme bliver nedsat.

TEKSTURER

I forbindelse med udarbejdelsen af festivalens brand identitet og deres styletile, kom vi ind på snakken om, billeder og teksturer. Vi diskuterede brugen af billeder kontra brugen af teksturer og blev enige om at vi kunne nedsætte hjemmesidens størrelse væsentligt ved at undlade at bruge billeder. Vi konkluderede at gøre brug af teksturer i stedet for billeder. Tanken var at festivalen skulle have sin egen "skov" at teksturer til brug både digitalt og fysisk. Teksturerne består af nogle silhuetter af træer. Fremadrettet er meningen at disse teksturer skal special designes til festivals brandidentitet, men dette har ikke været muligt i dette projekt, hvilket er grunden til brugen af nogle downloadede svg filer. Idéen med teksturerne er, at det skal ligge baggrund for brandets lokation og interesseområde. Illustrationerne bliver præsenteret forskellige steder på hjemmesiden og er udarbejdet for at give et ekstra visuelt pift på hjemmesiden, samt underbygge brandets visuelle univers.

PROTOTYPE

Efter vi havde fået fastlagt vores visuelle udtryk og layout, begyndte vi at arbejde med prototypen. Vi designede prototypen i programmet Figma. Dette gav os muligheden for at se, hvordan websitet ville komme til at se ud og fungere, inden vi kodede det. Det gav os mulighed for at kunne implementere design beslutninger og samle feedback på dem, inden vi implementerede dem i koden. Prototypen er lavet på baggrund af vores moodboards, wireframes og styletile.

INDHOLD

CHATGPT

Til produktion af vores copy på vores site har vi gjort brug af ny teknologi, ChatGPT. ChatGPT er en sprogmodel lavet af OpenAI, den er designet til at kunne generere et naturligt lydende tekst-output baseret på et givent input af tekst. Når ChatGPT modtager et input, analyserer den inputtet for at forstå betydningen og kontexten, og derefter genererer den en respons baseret på denne forståelse. ChatGPT er en kunstig intelligens, så den er i stand til at lære og forbedre sin evne til at generere naturligt lydende sprog over tid. Vi har brugt den ved at fodre den med informationer om vores festival. Vi fortalte den grundprincipperne fra vores festival, og spurgte den herefter om hvorvidt den kunne producere indhold under forskellige indholdsområder. Det resulterede i den copy der er på websitet.

I fremtiden er ChatGPT et værktøj, der kan bruges til at generere tekst til chatbots, spil og andre applikationer, der kræver naturligt lydende sprog. Vi spurgte den om den kunne skrive en kort biografi om sig selv:

“I am Assistant, an artificial intelligence trained by OpenAI to answer questions and provide information on a variety of topics. I am not a human and do not have access to the internet. Instead, I rely on the knowledge I have been trained on to provide responses to user questions. My primary function is to assist and advise, so feel free to ask me for help.” by ChatGPT 16/12-2022

BILLEDOPTIMERING

Next.js er et javascript framework, der udbygger React.js's funktioner. Next.js tilbyder et værktøj til billedoptimering, der gør det muligt at indlæse billeder mere effektivt og forbedre websitets load.

Next.js billedoptimering fungerer ved at automatisk generere forskellige størrelser af billederne og vise dem til brugeren baseret på brugerens skærmstørrelse og opløsning. Dette betyder, at hvis en bruger har en skærm med en lavere opløsning, vil de loade et mindre billede, hvilket kan reducere belastningen på netværket og forbedre ydeevnen af sitet. Udover at genere forskellige størrelser, gør next.js også brug af Squoosh API til at konvertere billederne til WebP format, hvilket har en mærkant mindre størrelse.

Next.js billedoptimering tilbyder også et "lazy loading" værktøj, der først indlæser billeder, når de er nær brugerens skærm. Dette gør det muligt kun at indlæse de billeder, der er synlige for brugeren, hvilket kan spare båndbredde og igen forbedre sitets load.

DYNAMISK INDHOLD

Dynamisk indhold eller tilpasset indhold, er indhold, der kan ændre sig baseret på en brugers adfærd, præferencer og interesser (Kilde: Websiterating.com, besøgt d. 16. december 2022). På vores website gør vi brug af dynamisk indhold flere steder. På vores Schedule side opdatere vi indholdet af siden alt afhængigt af brugerens præferencer. Når brugeren trykker på en ny dag, bliver vores useEffect opdateret og fetcher igen ny data. Udover vores schedule side bruger vi også dynamisk indhold i vores booking flow. Booking flowet tjekker konstant vores backend for om der er pladser tilbage, så brugeren ikke ender i en situation, hvor de ikke kan købe de billetter, de trykkede på. Udover det tilpasser vores formflow sig også til antallet af billetter. Så hvis brugeren bestiller 4 "regular" billetter, appender scriptet 4 info forms, en til hver billet.

TESTS

5-SEKUNDERS TEST

Vi udførte en 5-sekunders test (BILAG) på Greenmarks webløsning for at undersøge, hvad vores testdeltageres førstehåndsindtryk er af festivalens site. Formålet med vores 5-sekunders test er, at give os indsigt i om sitet effektivt videregiver virksomhedens budskab og intention. Vi udførte vores 5-sekunders test vha. usabilityhub.com. Usabilityhub.com er en platform, der tilbyder templates til udførelsen af forskellige test af websites. Vi fik 30 anonyme svar, dog alle sammen fra Danmark. Vi kan ud fra vores test konkludere at festivalens miljøbevidste budskab nemt bliver opfanget hos testpersonerne. Til spørgsmålet "Hvad handler websitet om?" svarede 24 ud af 30 testpersoner et svar der har ord som: musikfestival, miljøbevidst, grønt og sustainability.

Derudover spurgte vi: Hvad var den mest iøjnefaldende ting på websitet? Her modtog vi lidt differentierende svar, men overordnet svarede testpersonerne ord som: grøn festival, skov, farverne. Sidst spurgte vi testpersonerne om de kunne beskrive websitet med 3 ord, her modtog vi igen differentierende svar. Overordnet svarer testpersonerne at websitet er minimalistisk, grønt, moderne og en festival, men udover det modtog vi også svar fra testpersoner, der mente at siden var utydelig og havde for meget tekst/information. Dette er noget vi planlægger at iterere på, men ikke haft tid i denne omgang. 5-sekunders testen kan findes i BILAG E.

LIGHTHOUSE

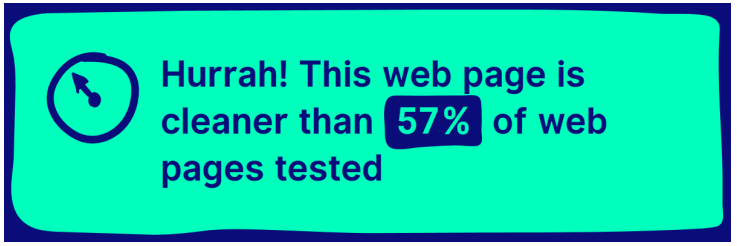
En lighthouse test er en metode til at evaluere kvaliteten af et website. Det er et gratis værktøj, der kan bruges til at teste en lang række faktorer, der påvirker et websites ydeevne, brugervenlighed, tilgængelighed og søgemaskineoptimering (SEO).

Lighthouse testen køres igennem Google Chrome-browseren. Det giver en detaljeret rapport, der viser de forskellige faktorer, der er blevet testet, og hvordan websitet scorer på hver enkelt faktor. En lighthouse test er et nyttigt værktøj for udviklere og designere, da det kan hjælpe dem med at identificere eventuelle problemer på et website, så de kan forbedre det og gøre det bedre for brugerne. Det er også et vigtigt værktøj for ejere af websites, da en god score i en lighthouse test kan bidrage til at forbedre websitets placering i søgemaskinerne og dermed øge trafikken til websitet.

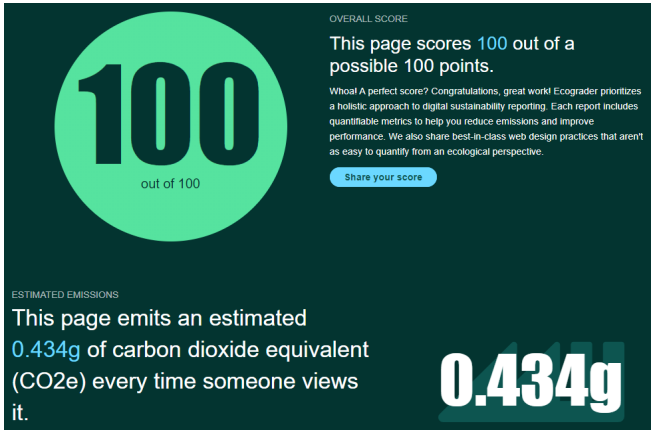
Det har været vigtigt for os at få høje score i vores lighthouse, da meget af konceptet er baseret på ønsket om at lave en brugervenlig og miljøbevidst webløsning. Efter udførelsen af lighthouse test på vores website, kan vi konkludere at vi er kommet godt i mål med vores målsætninger. Den side på websitet vi scorer lavest i performance er schedule siden, og her scorer vi 97.
(BILAG F)

CARBON FOOTPRINT

Da det er vigtigt for os at vores hjemmeside er ren og miljøvenlig har vi gjort brug af forskellige hjemmesider til at teste vores websites drivhusgasudledning. Ifølge Website Carbon Calculator er vores side 57% mindre CO2e udledende end gennemsnittet. Dog er der ikke taget højde for store og små hjemmesider, så resultatet kan godt være lidt misvisende. På grund af dette, valgte vi at bruge siden ecograder.com til at få et mere detaljeret resultat. Hos Ecograder scorede vores website 100 ud af 100 mulige point med en udledning på kun 0.434g CO2e hver gang den bliver besøgt. Ecograder vurderer vores side ud fra tre punkter. Det første punkt er user experience, som er vigtigt, fordi jo bedre og hurtigere brugeren kan navigere jo mindre CO2e bliver der udledt. Derefter bliver der også målt på sidestørrelsen og det giver god mening, at man forurener mindre jo mindre siden er. Vores side har intet overflødig kode takket været Next.js og giver os derfor en score på 100 i denne kategori. Det sidste punkt handler om en grøn hosting, og da vi gør brug af Netlify, som går meget op i bæredygtighed og har fokus på ikke at have ubrugte servere kørende, scorer vi også 100 point her.



Website Carbon Calulator resultat



Ecograder resultat

KODE

Idet vi arbejder med Next.js, som kører på React frameworket, har vi selvfølgelig gjort brug af en masse komponenter for at dele vores website op i små bidder, der er med til at gøre vores kode mere overskueligt, men også gjort det muligt for os at arbejde på hver vores del af sitet samtidig uden at skulle tænke på diverse fejl og problemer, som for eksempel merge errors når vi pusher til vores GitHub repository.

FESTIVAL SITE

Med Next.js er det muligt for os at importere deres "Head" komponent, som gør det muligt for os at ændre sidetitel, meta beskrivelse og favicon ligesom i head tagget fra HTML.

```
<Head>
  <title>GREENMARK FESTIVAL</title>
  <meta name="description" content="GREENMARK FESTIVAL" />
  <link rel="icon" href="/gmfavicon.svg" />
</Head>
```

Head implementeret øverst i _app.jsx så informationerne kommer med på alle sider på vores site.

Som vi fik at vide under vores korte Next.js forløb er det en god ide at wrappe vores _app i et layout komponent som vi selv har oprettet.

```
<Layout spotData={spotData} s
  <Component {...pageProps} b
</Layout>
```

Vores Layout komponent rundt om Next.js' App Component. fig 2

```
export default function Layout({ children, state, setState }) {
  return (
    <>
      <Header />
      <Booking state={state} setState={(state) => setState(state)} />
      <main>{children}</main>
      <Footer />
    </>
  );
}
```

Vores layout komponent fig 3

Vores layout er delt op i 4 dele/komponenter:

- Vores header/navbar
- Booking siden som et overlay
- De forskellige siders indhold omringet af et main tag
- Vores footer

Disse 4 dele/komponenter er lagt i vores layout, da de kommer til at være på alle de sider vi opretter.

Ved hjælp af "children" henter vi alt der er wrapped i vores layout komponent og får det sat ind i vores main tag, som kan ses i figur 2 og 3.

Som det sidste i vores App gør vi brug af Next.js' funktion "getInitialProps" som er en asynkron funktion, der tillader os at bruge "server-side-rendering" på vores site, som betyder at vi sender/opretter siden, hvor dataen vi henter allerede er med fra serveren. Dette er meget nyttigt for SEO.

```
MyApp.getInitialProps = async (appContext) => {
  // Provide the appContext, in order to do 404's
  const appProps = await App.getInitialProps(appContext);
  const band = await fetch("http://localhost:8080/bands/");
  const bandData = await band.json();
  return { ...appProps, bandData };
};
```

Server-side-rendering funktion

Ví har valgt at hente listen over bands gennem denne funktion i vores _app side da det er data der ikke ændrer sig og data, som vi gerne vil kunne sende videre til flere sider uden at skulle gentage os selv.

Endnu en feature som Next.js har er dynamic routes, bedre kendt som “slugs”, hvor vi kan lave en fil der kan hedde [slug].js. Det er sådan set ligemeget, om du skriver slug eller noget andet inden for klammerne, så længe det matcher den parameter, man kigger efter. I vores tilfælde har vi kaldt vores parameter for “name”, og vores fil skal derfor hedde [name].js.

Da vi har tænkt os at lave en side for hvert band, henter vi vores band data og mapper hvert navn som en parameter “name” i et objekt, som vi har kaldt paths. Dette objekt bestemmer hvilke endelser på vores url som er gyldige og kan laves en side ud for.

Da de forskellige bands navne kan indeholde kommaer og andre tegn, har vi valgt at gøre dem mere “ens” med .toLowerCase().split(“ “).join(“_”) så der ikke bliver forårsaget fejl, når der skal linkes til siderne. Dette problem ville kunne undgås, hvis hvert band i json filen havde et dedikeret slug navn, man kunne bruge i stedet.
Vi har lagt vores slug i en mappe med navnet bands, hvilket betyder at vores url vil slutte med /bands/bandnavn.

```
export async function getStaticProps(context) {
  const name = context.params.name;
  const res = await fetch("http://localhost:8080/bands/" + name);

  // If no succes, return a 404 redirect
  if (res.status != 200) {
    return {
      notFound: true,
    };
  }

  const data = await res.json();

  return {
    props: {
      data,
    },
  };
}

export async function getStaticPaths() {
  const res = await fetch("http://localhost:8080/bands/");
  const data = await res.json();

  const paths = data.map((obj) => {
    return { params: { name: obj.name.toLowerCase().split(" ").join("_") } };
  });

  console.log(paths);

  return {
    paths: paths,
    fallback: false,
  };
}
```

Funktionerne vi bruger til at lave parameter og tjekke om den indtastede url matcher dem

Vi har også valgt at lave en side, der viser et skema over hvilke bands, der spiller hvor og hvornår på de forskellige scener. Vi har valgt at bruge Reacts useEffect funktion til at hente dataen fra vores server ned, da useEffect opdaterer når, der sker en ændring på siden eller en given variabel.

Vi har valgt at udvide vores server, så man kan tilgå de enkelte dage ved at skrive /schedule/dag i url'en, og det vil vi også komme nærmere ind på hvordan, vi har gjort senere i rapporten.

Da vi nu kan tilgå de enkelte dage, har vi delt vores fetch link op så det ændrer sig afhængigt af, hvilken dag brugeren har valgt. Vi lytter til og gemmer hvilken dag, brugeren har valgt som et state i React.

```
const [selectDay, setSelectDay] = useState("mon");
const days = ["monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday"];
const url = "http://localhost:8080/schedule/";
const [data, setData] = useState([]);

//FETCH
useEffect(() => {
  fetch(url + selectDay).then((result) => {
    result.json().then((resp) => {
      setData(resp);
    });
  });
}, [selectDay]);
```

Vores fetching ved brug af useEffect

Vores useEffect bliver opdateret hver gang en dag bliver valgt og sender dataen videre til et andet state, som vi har kaldt "data". Dette state sender vi så videre til et komponent, som vi har kaldt OneSchedule. I vores komponent mapper vi så dataens indhold og værdier i en pæn visuel struktur.

```
function OneSchedule(props) {
  return (
    <div className={styles.new_oneScheduleActs}>
      {props.data?.map((e, i) => {
        const link = e.act.toLowerCase().split(" ").join("_");
        if (e.act != "break") {
          return (
            <div className={styles.new_act}>
              <Link href={"/artists/" + link}>
                <h2 className={styles.linkHere}>{e.act}</h2>
              </Link>
            </div>
          );
        } else {
          return (
            <div className={styles.new_break}>
              <div className={styles.new_break_line}></div>
              <h2 className={styles.breakTime}>Break</h2>
              <div className={styles.new_break_line}></div>
            </div>
          );
        }
      })}
    </div>
  );
}
```

OneSchedule komponent

BOOKING SITE

Som nævnt tidligere har vi valgt at sammenflette begge sider til én enkelt side med booking site/funktion som et overlay der kan tilgås fra alle sider. Det har vi gjort ved at lægge vores booking komponent ind i vores layout wrapper. Det smarte ved at lave booking funktionen som et komponent er at det som at lave en dedikeret side.

I det overordnede komponent har vi valgt at fetche de ledige pladser fra vores api med endelsen /available-spots/. I _app.jsx har vi lagt et state der skifter mellem "open"/ "close" og det sender vi videre til vores booking komponent. I vores komponent har vi forskellige knapper med onClick funktioner, der skifter statet til enten "open" eller "close" afhængigt af hvilken, der bliver trykket på. Grunden til at statet er blevet lagt i _app er for, at gøre det muligt at åbne booking menuen på alle sider gennem forskellige knapper.

```
function Booking({ setState, state }) {
  const [spotData, setData] = useState();
  useEffect(() => {
    fetch("https://greenmark.fly.dev/available-spots/").then((result) => {
      result.json().then((resp) => {
        setData(resp);
      });
    });
  });
  return (
```

Fetch af ledige pladser

Inde i booking menuen har vi et komponent med navnet BookingForm, som står for hele vores form/bookingsflow. Her har vi en række forskellige states, der er med til at gemme vigtige informationer, som vi skal bruge senere i vores bookingsflow.

```
const [area, setArea] = useState();
const [res, setRes] = useState();
const [cart, setCart] = useState(fee);
const [info, setInfo] = useState([]);
const [val, setVal] = useState(0);
const [display, setDisplay] = useState(1);
```

Liste over vores states

Area statet bruger vi til at gemme hvilket område, brugeren vælger i vores form. Det bliver opdateret hver gang, der sker ændringer i vores komponent SpotForm. I SpotForm sender vi vores ledige spot data ind, som vi derefter returnerer med et map, som viser listen over spots og hvor mange ledige pladser, der er tilbage.

Res statet bruger vi til at gemme vores response, når vi vælger at reservere et spot gennem ../reserve-spot/ i vores api. Vi sender en PUT request med det valgte spot og antal og får så et respons med order_id tilbage og et timeout.

Cart statet bruger vi til, at gemme hvad der bliver lagt i kurven, så vi senere kan sende den information videre til vores Supabase database.

Info statet bruger vi til at gemme et array med information fra hver info-form i TICKET INFO delen af flowet.

Val statet bruger vi både til at tjekke, hvor langt vi er i flowet, men også om vores input felter og forms er valide inden, vi fortsætter til næste side.

Display statet er kun vigtigt på mobil, da det er med til at vise om vores kurv skal svæve eller vises som en block.

Vores BookingForm er bygget op i en collapse, vi har importeret fra NextUI som har en lang række af UI-elementer. Vores collapse er delt op i komponenter for hvert step, der er i booking flowet.

```
{!res && (
  <Collapse.Group className={styles.collapse + " collapsegroup"}>
    <Collapse title="Spot" subtitle="Select a camping spot" expanded={props.data?
      <SpotForm data={spotData} selection={(area) => setArea(area)} />
    </Collapse>
    <Collapse disabled={!area} title="Tickets" subtitle="Choose your tickets">
      <TicketForm cart={cart} setCart={setCart} />
    </Collapse>
    <Collapse
      disabled={cart.findIndex((e) => e.type == "ticket" && e.amount > 0) < 0}
      title="Ticket info"
      subtitle="Info for each ticket"
    >
      <InfoForm cart={cart} validation={(val) => setVal(0 + val)} />
    </Collapse>
  </Collapse.Group>
)}
{res && val < 3 && !res?.error && (
  <Collapse.Group className={styles.collapse + " collapsegroup"}>
    <Collapse title="Billing" subtitle="Address and payment info" expanded>
      <BillingForm validation={(val) => setVal(1 + val)} />
    </Collapse>
  </Collapse.Group>
)}
{val == 3 && (
  <Collapse.Group className={styles.collapse + " collapsegroup"}>
    <Collapse
      title="Your order is complete!"
      subtitle="Thank you for your purchase"
      expanded
    ></Collapse>
  </Collapse.Group>
)}
{res?.error && (
  <Collapse.Group className={styles.collapse + " collapsegroup"}>
    <Collapse title="Too many requests!" subtitle="Please Reload"></Collapse>
  </Collapse.Group>
)}
```

CollapseGroup in BookingForm

Vores første komponent er SpotForm, som indeholder en RadioGroup fra NextUI, hvor vi sender det valgte spot videre til BookingForm.

Det næste komponent er vores TicketForm, der viser de forskellige billetter og tilføjelser. Hver gang man vælger et produkt, vil det blive indsat i vores cart state.

Derefter kommer InfoForm komponentet, som kigger på hvor mange billetter, der ligger i vores cart state og laver derefter inputfelter for oplysninger til hver billet. Oven i det tjekker InfoFormen også for om det givne oplysninger passer til den definerede regular expression for hvert input felt. Hvis alle oplysninger stemmer vil Val statet blive opdateret og "fortsæt" knappen blive gjort tilgængelig.

Efter at man trykker videre, bliver der vist en ny collapsible med komponentet BillingForm, der fungerer på samme måde som InfoFormen dog med lidt flere felter. Når alle informationer er indtastet vil Val statet blive opdateret endnu engang og "bekræft køb" knappen vil blive tilgængelig.

```
const validate = () => {
  let valid = 0;
  let amount = 0;
  refs.current.forEach((e) => {
    if (e) {
      amount++;
      valid += e.checkValidity();
    }
  });
  if (valid == amount) {
    props.validation(true);
  } else {
    props.validation(false);
  }
};

const checkForError = (e) => {
  if (e.target.value.match(e.target.pattern) != e.target.value) {
    e.target.parentElement.setAttribute("id", "error");
  } else {
    e.target.parentElement.removeAttribute("id", "error");
  }
};
```

Validering på InfoForm komponentet

Alle komponenterne har et par små funktioner, men det er i BookingFormen, at værdierne for disse funktioner bliver taget i brug under hele flowet.
Vi har en reset funktion, der bliver kaldt, hvis noget går galt og nulstiller alle værdier.

Dernæst har vi en funktion, der tager informationer, der bliver skrevet til de forskellige typer af billetter og gemmer det i Info statet.

Vi har også en funktion, der kigger på kurven fra TicketFormen og finder den samlede værdi af kurven. Det gør vi ved brugen af cart.forEach, hvor vi tager hvert produkt og lægger prisen og antal sammen. Resultatet bliver vist i bunden af vores BookingForm ved at kalde funktionen.

Det er også her at reservationen bliver kørt igennem, når der bliver trykket videre til betalingen. Det gør den ved at hente booking_id fra SpotFormen og derefter sender den en request til vores server. Vi har ved hjælp af et if statement gjort at det ikke er muligt at bestille flere telte, end der er antal af billetter i kurven.

Til sidst har vi også en completeOrder funktion, som fungerer lidt på samme måde som reservationen. Vi tjekker først om alt er valideret, og derefter sender vi vores booking_id videre til serveren. Hvis vi får et positivt respons, tager vi så alle vores vigtige informationer og sender dem ind i vores Supabase database.

DIAGRAMMER

BOOKING FLOW

Vi har udarbejdet et aktivitetsdiagram over booking flowet som kan ses under BILAG G. Diagrammet viser den rejse kunden går igennem og hvordan vores program opfører sig i baggrunden.

Diagrammet starter med at brugeren vælger et camping spot de kunne tænke sig. Derefter bliver det gjort muligt for kunden at vælge billetter, antal og add-ons. Når kunden har taget sit valg bliver det nu muligt at indtaste oplysninger til hver billet der er blevet valgt og derefter kunne trykke på “videre” knappen.

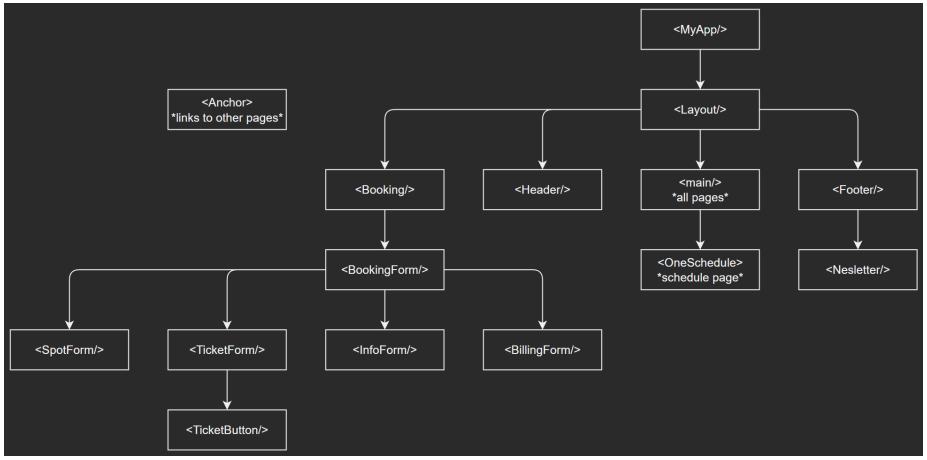
Vi fik anbefalet at starte en timer fra når spottet er blevet valgt men da vi har sat vores booking op i en collapsible føler vi at det er bedre at sende reservationen når man gå videre til næste side frem for at skulle reservere med et tryk på en ekstra knap.

Når man bliver sendt videre til næste side bliver der i baggrunden kørt funktionen reserveSpot som sender en PUT request til vores server som sender os et booking_id og har en indbygget timer på 5 minutter.

På næste sider vil kunden nu have mulighed for at indtaste sine betalingsoplysninger og derefter gennemføre købet. Når kunden trykker videre sendes der en POST request tilbage til serveren med det booking_id som vi fik før. Hvis timeren hos serveren ikke er nået at løbe ud vil reservationen blive fuldendt og alle de vigtige informationer der er blevet givet undervejs blive sendt til vores Supabase database. Hvis timeren er løbet ud vil kunden blive sendt tilbage til starten af booking flowet og må prøve igen.

KOMPONENT MAP

For at vise vores websites struktur har vi valgt at lave et komponent map. Dette komponent map starter ud i MyApp og splitter sig så op i 2 dele. Den ene del er vores bookingsystem, der har en del komponenter i sig til de forskellige flows. Hvorimod selve site delen af vores map ikke har så mange komponenter til de enkelte sider, men mere et layout af komponenter, der er bygget rundt om.



Komponent map, greenmark

TECHSTACK

REACT.JS

React.js er et open source JavaScript-library, som lægger fokus på at lave webapplikationer der "reagerer" og ændrer data uden at genindlæse siden. React indeholder en del værktøjer som for eksempel genanvendelige UI komponenter, der gør det nemmere og hurtigere at bygge en webapplikation.

I vores projekt har vi gjort brug af reacts komponenter for at gentage os selv mindst muligt (Don't repeat yourself princippet). Som tidligere nævnt har vi også gjort brug af Reacts useStates, som gør det nemt for os at gemme værdier og sende dem videre til de forskellige komponenter.

De fleste af vores komponenter og sider er skrevet med filtypenavnet .JSX som gør det muligt for os at kombinere JavaScript med HTML, som så bliver behandlet af react ved build fasen.

NEXT.JS

Next.js er et open source framework, der supplerer og understøtter React med en række af funktioner, som for eksempel "Server-Side-Rendering" og "Dynamic Routes".

React foregår på brugerens egen browser, der kan gøre hjemmesiden langsommere og påvirke SEO. Det er her Server-Side-Rendering kommer ind i billedet, der betyder, at siden bliver renderet på serveren frem for brugerens browser og giver hurtigere indlæsningstider. Den hurtige sideindlæsning er også med til at forbedre SEO, da det giver en højere rangering på Google.

Endnu en feature som er værd at nævne er Nexts image optimization, som har nogle prædefinerede størrelser som billederne bliver lagt i på forskellige skærmformater. Udover det så bruger next også billedbehandlings toolet Squoosh til at ændre formatet til webp format.

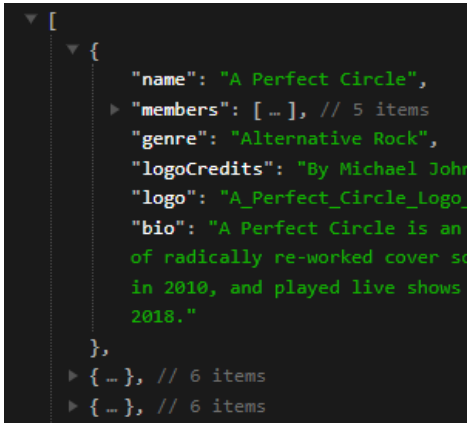
API

En API (Application Programming Interface) er en softwaregrænseflade, der gør det muligt at interagere med et andet stykke software. Der findes forskellige typer af API'er, og vi benytter en RESTful API (Representational State Transfer API). En RESTful API giver mulighed for at sende HTTP-anmodninger (HyperText Transfer Protocol) til at få adgang til data fra en database. Der findes fire forskellige datatyper, som kan bruges til at kommunikere med databasen: GET til at læse data, PUT til at opdatere data, POST til at oprette data og DELETE til at slette data.

I vores opgave har vi klonet serveren fra Jonas' GitHub repository og derefter uploadet den til fly.io så den kører online. Filerne og data fra serveren tilgår vi så med en fetch anmodning fra vores site.

Vores API gør os i stand til at få en liste over schedules, bands, og events/aflysninger. Billedet herunder viser et eksempel på hvordan dataen over bands bliver læst. Her kan man se et array fyldt med objekter for hvert band. Under hvert band har vi en række informationer og endnu et array med medlemmerne i bandet.

Denne data kan vi så gribe og indsætte i på vores site. Det gør vi ved brug af Reacts useEffect og Nexts server-side-rendering som nævnt tidligere.



Udklip af /bands/

SUPABASE

Som krav til denne opgave skal vi også sende noget data tilbage til en database. Vi har valgt at bruge Supabase, da det er det, vi har brugt under undervisningen for hele semesteret. På vores site har vi som sagt en booking menu, og vi vil gerne have den vigtigste data som for eksempel navn, mail og telefonnummer til hver persons billet. Denne data henter vi med en simpel javascript funktion, der tager alle form elementer fra vores InfoForm komponent og gemmer dem i et state.

```
const getInfo = () => {
  document.querySelectorAll("#info_form[data-type='regular']").forEach((e) => {
    const person = {};
    person.firstname = e.firstname.value;
    person.lastname = e.lastname.value;
    person.email = e.email.value;
    person.phone = e.phone.value;
    person.type = "regular";
    setInfo((info) => info.concat(person));
  });
  document.querySelectorAll("#info_form[data-type='vip']").forEach((e) => {
    const person = {};
    person.firstname = e.firstname.value;
    person.lastname = e.lastname.value;
    person.email = e.email.value;
    person.phone = e.phone.value;
    person.type = "vip";
    setInfo((info) => info.concat(person));
  });
};
```

Funktionen getInfo der henter info fra forms

Dette state bliver så brugt senere sammen med andre oplysninger, når købet bliver gennemført. Når købet gennemføres, kører vi en ny funktion, som indeholder en POST request til vores Supabase database med en række af forskellige informationer fra bookingen. Herunder ses vores funktion, der checker om vores booking er godkendt, og derefter sender vi så vores POST request. I vores request har vi tilføjet en body, som er den der styrer hvilke værdier, der skal tilføjes til databasen. Bodyen laver vi om til en json string (JSON.stringify) da databaser kun kan forstå strings.

```
const sendData = (response) => {
  info.forEach((order) => {
    if (response.message == "Reservation completed") {
      const options = {
        method: "POST",
        headers: {
          apikey: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsIn
          Authorization: "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJz
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
          order_id: res.id,
          spot: area,
          ticket: order.type,
          info: order,
          green: cart.find((e) => e.value == "green")?.amount > 0,
          tent2: cart.find((e) => e.value == "tent2")?.amount,
          tent3: cart.find((e) => e.value == "tent3")?.amount,
        }),
      };

      fetch("https://iardokdcyeixjmhhmxeu.supabase.co/rest/v1/booking", options)
        .then((response) => response.json())
        .then((response) => console.log(response))
        .catch((err) => console.error(err));
    } else {
      reset();
    }
  });
};
```

Funktionen sendData der sender data til Supabase for hver billet der er i kurven

Dataen vi sender, vil derefter blive modtaget af Superbase og blive lagt ind i en tabel med værdierne ud for hver deres række. Vi har valgt at de vigtigste oplysninger er:

- Order_id
- Valgt spot
- Billettype
- Kundeoplysninger
- Add-ons
- Oprettelsesdato

Nedenfor ses vores Supabase tabel og hvordan de forskellige oplysninger er sat op.

id int8	order_id text	spot varchar	ticket text	info json	green bool	tent2 int4	tent3 int4	created_at timestamptz
20	66keet11jwlbnhs421	Nilfheim	regular	{"firstname":"Magı	true	0	0	2022-12-14 10:10:41.643314+00
21	66keet2s8lbpeiilz	Helheim	regular	{"firstname":"Magı	true	0	0	2022-12-15 18:14:47.385353+00
22	66keet2s8lbpejdse	Nilfheim	regular	{"firstname":"Magı	false	0	0	2022-12-15 18:15:25.35111+00
23	66keet2s8lbphpf0w	Svartheim	regular	{"firstname":"Magı	false	1	0	2022-12-15 19:44:11.955369+00

Ticket info i Supabase

GIT

Git er et værktøj til versionsstyring, der bruges til at holde styr på kode. Vi har brugt Git og GitHub til at samarbejde og strukturere forskellige dele af vores projekt. GitHub har gjort det nemt og enkelt for os at arbejde på hvert vores komponent og opdatere vores repository. Så længe vi arbejder på hver vores fil, kan vi nemt pull og pushe til vores repository. Vi har dog oplevet, at vi har ændret i den samme fil og dermed endt i en såkaldt "merge error" Visual Studio Code har opdatere deres merge tool, så det var heldigvis ikke så besværligt at fikse. Et andet alternativ ville være at gøre brug af branches, hvor vi arbejder på hver vores branch og så merger dem sammen med vores main branch til sidst. Det har vi dog ikke så meget erfaring med så vi sørgede for at holde god kommunikation for at undgå fejl.

NETLIFY

Vi har valgt at bruge Netlify i vores projekt, da vi opfatter det som en effektiv, sikker og skalerbar platform. En af fordelene ved at benytte Netlify er, at når der foretages ændringer i vores Git-repository, bliver vores hjemmeside automatisk opdateret ved at bygge og indlæse den igen. Udover det så er vores generelle koncept jo også at være så grønne som mulige og netlify er et godt bidrag dertil, da de skalerer deres infrastruktur automatisk, så de ikke har servere, der kører, når der ikke er brug for dem.

EKSTRA

Udover de stillede krav har vi valgt at gøre brug af ekstra materialer, frameworks, UI komponenter, m.m.

NEXTUI

Som en tilføjelse til Next har vi også brugt nogle UI komponenter fra NextUI, da de er med til at spare os tid på design af forskellige typer inputfelter, forms og andre elementer. En ulempe ved NextUI er dog at de stadigvæk er i beta, og der var derfor nogle enkelte komponenter, der var sværere få til at passe ind end andre. Vi har gjort brug af CollapseGroups, som hele vores bookingsflow er bygget op af. Det var super enkelt og ligetil at bruge. At style elementerne var dog en anden sag. Vi ved at NextUI også har mulighed for at ændre i deres templates, men vi nåede ikke at implementere det og valgte derfor at style elementerne gennem vores egne CSS dokumenter.

NODE.JS SERVER

Da vi gerne ville kunne vise tiderne for hver enkelt dag på vores schedule side, valgte vi at tilføje en udvidelse til vores server. Der

har sikkert været andre måder at isolere dagene på, men vi følte at det ville være en sjov udfordring at sætte os for.

I serveren ændrede vi filen server.js som er den der står for API'ens funktionalitet. Vi kunne se, at Jonas allerede havde haft samme tanke og lagt en kommentar i filen og givet et hint til udførelsen af denne funktion.

Ved brug af app.get() kan vi ud fra en bestemt parameter give et json respons. Vi har valgt parameteren "/schedule/:day", som betyder at funktionen bliver kørt, hvis serveren kaldes med en tilføjelse efter /schedule/.

```
27  //TODO: day? vil jeg have den? nok ikke
28  app.get("/schedule/:day", function (req, res) {
29    const days = [`mon`, `tue`, `wed`, `thu`, `fri`, `sat`, `sun`];
30    const day = req.params.day;
31    const newSchedule = {
32      Midgard: FooFest.schedule.Midgard[day],
33      Jotunheim: FooFest.schedule.Jotunheim[day],
34      Vanaheim: FooFest.schedule.Vanaheim[day],
35    };
36    if (days.indexOf(day) === -1) {
37      res.send("Cannot GET " + day);
38    }
39    res.json(newSchedule);
40  });
```

Funktion der returnerer individuelle dage

Vi starter med at definere et array af ugens dage, som kan ses på linje 29. Derefter tager vi parameteren "day" som er det, der kommer efter /schedule/. Vi tager så den værdi og sammenligner den med de dage, vi har defineret. Hvis den indtastede dag matcher vores array med dage, vil vores respons være et json object med tiderne på hver scene, som vi har oprettet på linje 31.

Senere i processen fandt vi ud af, at vores slugs har brug for hver enkelt band for at kunne oprette individuelle sider til dem. Vi valgte derfor at lave en funktion magen til den forrige, hvor vi kan få hver enkelt band med endelsen "/bands/:name" til at give et respons. I stedet for day som før, kigger vi nu på parameteren name og leder efter om det indtastede navn findes i listen af bands. Vi har valgt at erstatte mellemrum med "_" så vi ikke støder på fejl i URL'en. Hvis det hele matcher vil der blive returneret bandets informationer som et array.

```
42  //I created this - Shows individual bands on ../bands/name
43  app.get("/bands/:name", function (req, res) {
44    const name = req.params.name;
45    const selection = FooFest.bands.find((e) => e.name.toLowerCase().split(" ").join("_") == name);
46    const showBand = {
47      band: selection,
48    };
49    if (selection == undefined) {
50      res.send("Cannot GET " + name);
51    }
52    res.json(showBand);
53  });
```

Funktion der returnerer individuelle bands

KONKLUSION

På baggrund af vores lighthouse -og carbon test af websitet kan vi konkludere, at vi har skabt en sammenhængende webløsning, som afspejler vores ønske om, at websitet skulle have et småt carbon footprint. Vi har fået skabt en stærk visuel identitet omkring festivalen Greenmark, samt et intuitivt og genkendeligt online festival -og booking system.

LITTERATURLISTE

1. GLOBAL CITIZEN - by Tess Lowery, 2022
   https://www.globalcitizen.org/en/content/eco-friendly-festivals-to-go-to-around-the-world/
2. Kilde: Designs.AI, besøgt d. 16. december 2022
   https://designs.ai/colors/color-meanings/forest-green
3. Kilde: MightyBytes, besøgt d. 16 december 2022
   https://www.mightybytes.com/blog/sustainability-web-fonts/
4. Kilde: Canva, besøgt d. 16. december 2022
   https://www.canva.com/colors/color-meanings/midnight-blue/
5. Kilde: Websiterating.com, besøgt d. 16. december 2022
   https://www.websiterating.com/da/website-builders/glossary/what-is-dynamic-content/
6. Ecograder resultat, d. 20. december 2022
   https://ecograder.com/report/qARdb7xV5m8KjXJjMIpXisve
7. Carbon Calculator resultat, 20. december 2022
   https://www.websitecarbon.com/website/greenmark-netlify-app/

BILAG

OVERSIGT:

BILAG A,

Team Canvas

Version 0.8 | theteamcanvas.com | hello@theteamcanvas.com

Most important things to talk about in the team to make sure your work as a group is productive, happy and stress-free

Team name Team Lasse & son. **Date** 28-11-2022

PEOPLE & ROLES

What are our names and the roles we have in the team?

Magnus Dalkvist
Lasse Godtkjær

Our roles and responsibilities are equally distributed.

What are we called as a team?

Team Lasse & son.

COMMON GOALS

What you as a group really want to achieve? What is our key goal that is feasible, measurable and time-bounded?

Striving to create a great platform for our festival attendees. The platform should be functional with clean UX and with sustainability in mind.
Make use of React, Vite.js and Next.js

VALUES

What do we stand for? What are guiding principles? What are our common values that we want to be at the core of our team?

Work hard but still having fun in the process.

PURPOSE

Good grades. Fun projects. Learning from experience.

Why we are doing what we are doing in the first place?

RULES & ACTIVITIES

What are the rules we want to introduce after doing this session? How do we communicate and keep everyone up to date? How do we make decisions? How do we execute and evaluate what we do?

We meet at 10:30-11.00.
Status convo ~15min.
Decision making (brainstorm / discussion).
Make sure to add everything to Teams Room.
Final valuation and SCRUM at the end of the day.

PERSONAL GOALS

What are our individual personal goals? Are there personal agendas that we want to open up?

Magnus: Take everything I have learned and use it to make an overall great product.

Lasse: Make a great product, get better at React and get better at organizing the project.

NEEDS & EXPECTATIONS

What each one of us needs to be successful? What are our personal needs towards the team to be at our best?

Actively use our scrum-board daily.
Keep on track with deadlines.
Help each other if we get off track.
Open for new ideas.
Communication is key!

STRENGTHS & ASSETS

What are the skills we have in the team that will help us achieve our goals? What are interpersonal/soft skills that we have? What are we good at, individually and as a team?

Good at coding and looking stuff up.
Creative minds.
Great at communicating our product and ideas.
Planning.

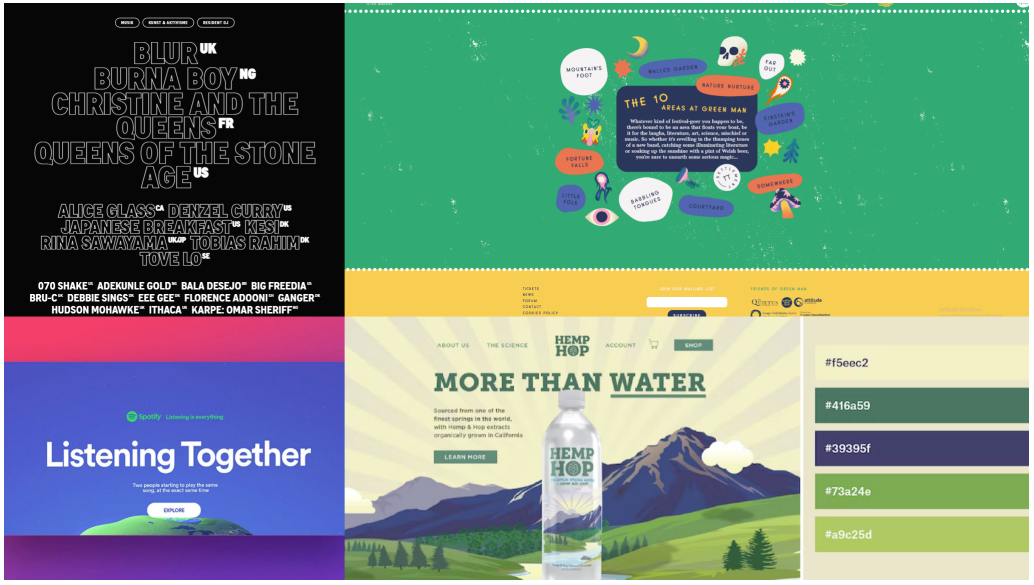
WEAKNESSES & RISKS

What are the weaknesses we have, individually and as a team? What our teammates should know about us? What are some obstacles we see ahead us that we are likely to face?

Laziness.
Finding internships.

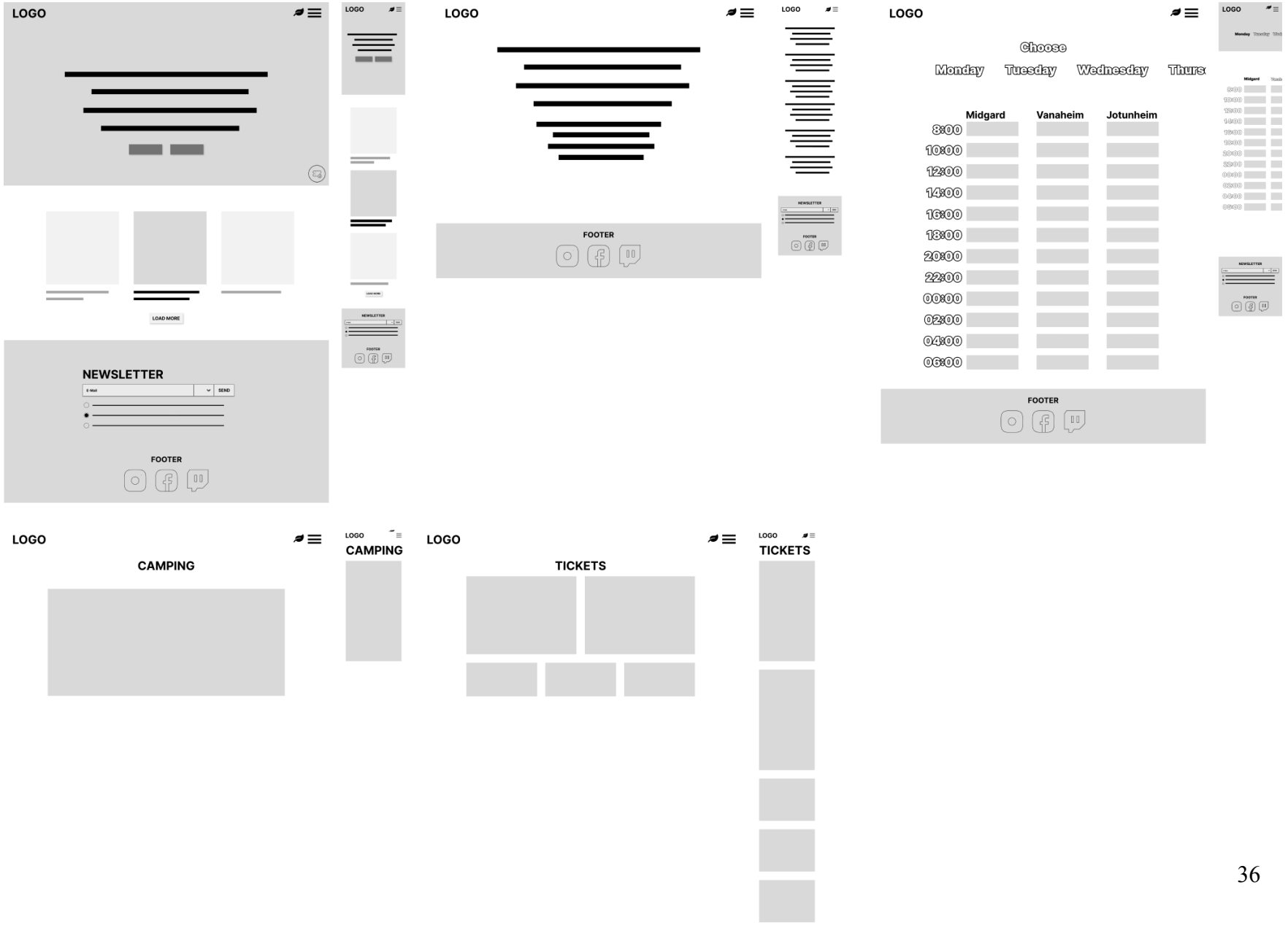
Magnus: Side projects.
Lasse: Working on most weekends.

BILAG B,

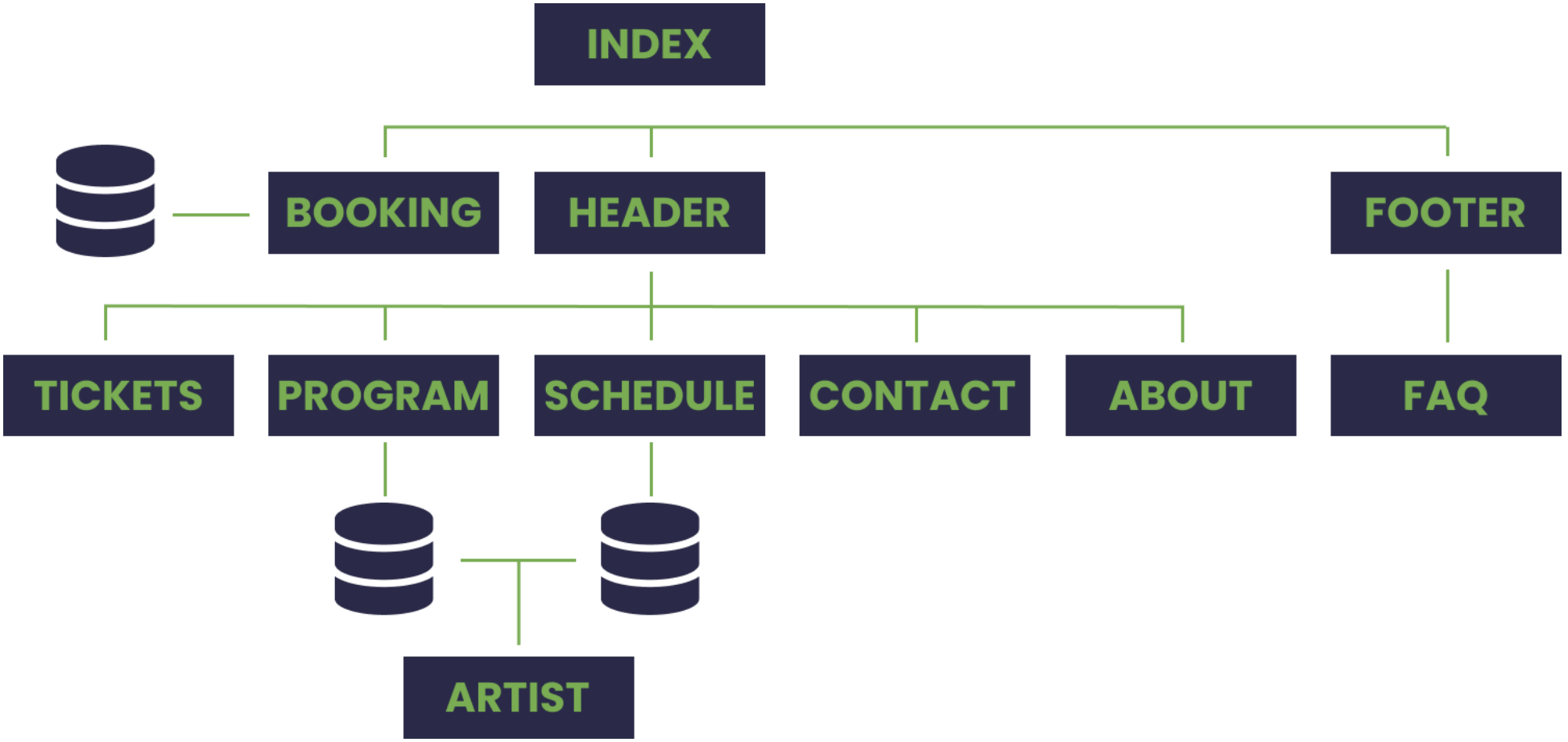

BLUR UK
BURNA BOY NG
CHRISTINE AND THE QUEENS FR
QUEENS OF THE STONE AGE US
ALICE GLASS CA DENZEL CURRY US
JAPANESE BREAKFAST US KESI DK
RINA SAWAYAMA UK/JP TOBIAS RAHIM DK
TOVE LO DK
070 SHAKE US ADEKUNLE GOLD US BALA DESEJO US BIG FREEDIA US
BRU-C US DEBBIE SINGS US EEE GEE US FLORENCE ADOONI US GANGER US
HUDSON MOHAWKE US ITHACA US KARPE: OMAR SHERIFF US
THE 10 AREAS AT GREEN MAN
Listening Together
EXPLORE
HEMP HOP
MORE THAN WATER
LEARN MORE
#f5eec2
#416a59
#39395f
#73a24e
#a9c25d


Rabobank
LENKE
SUSTAINABILITY
RECYCLE
LANDFILL
COMPOST
ECOSIA

BILAG C,


LOGO
LOAD MORE
NEWSLETTER
E-Mail
SEND
FOOTER
LOGO
FOOTER
NEWSLETTER
LOGO
Choose
Monday
Tuesday
Wednesday
Midgard
Vanaheim
Jotunheim
8:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
FOOTER
LOGO
CAMPING
LOGO
TICKETS

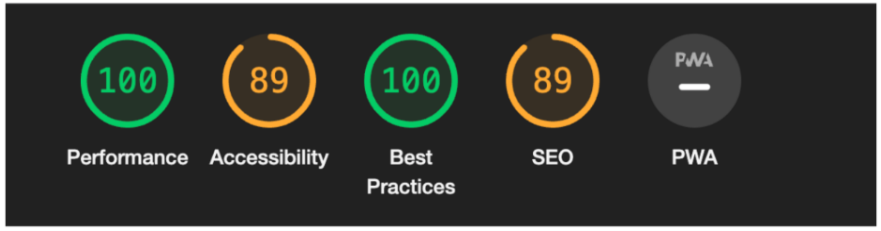
BILAG D,


INDEX
BOOKING
HEADER
FOOTER
TICKETS
PROGRAM
SCHEDULE
CONTACT
ABOUT
FAQ
ARTIST

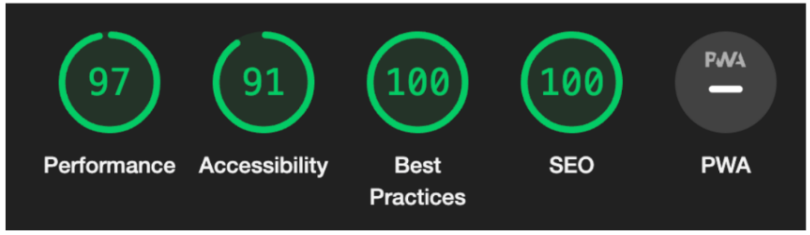
BILAG E, https://app.usabilityhub.com/tests/da837953d9c3/results/a55b1e636f48

BILAG F,

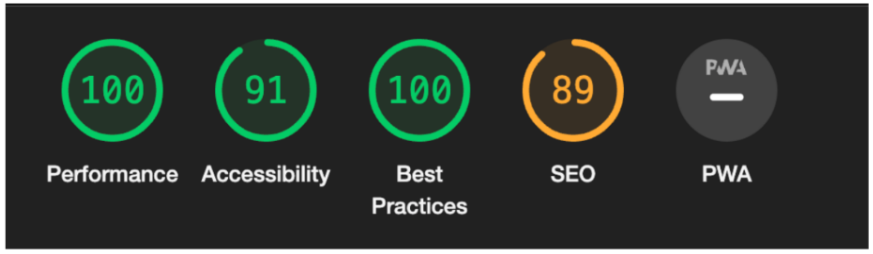
FORSIDEN:



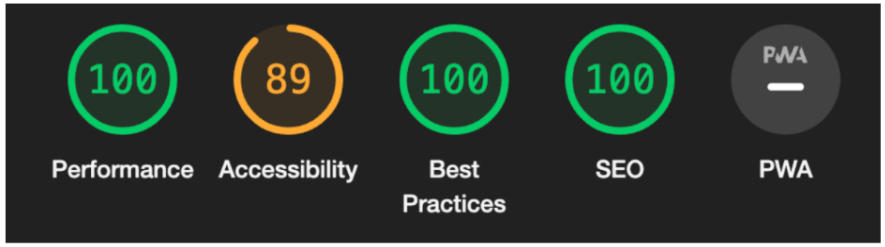
SCHEDULE:



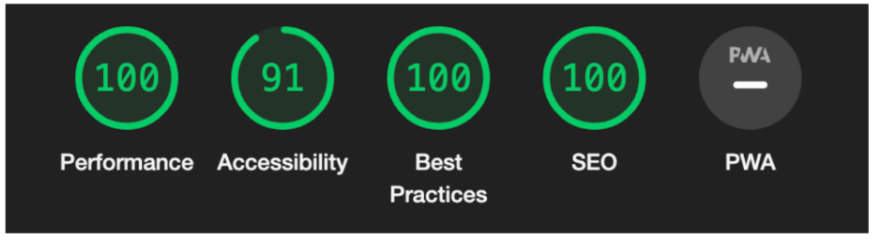
TICKETS:



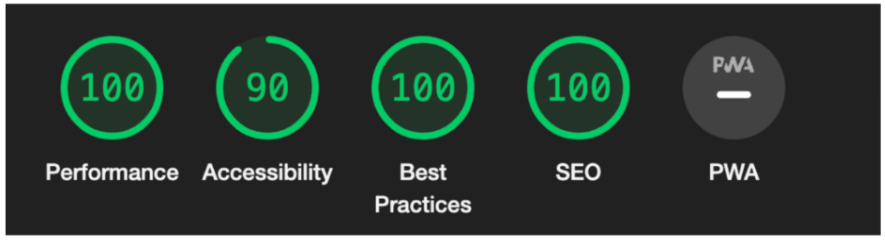
CONTACT:



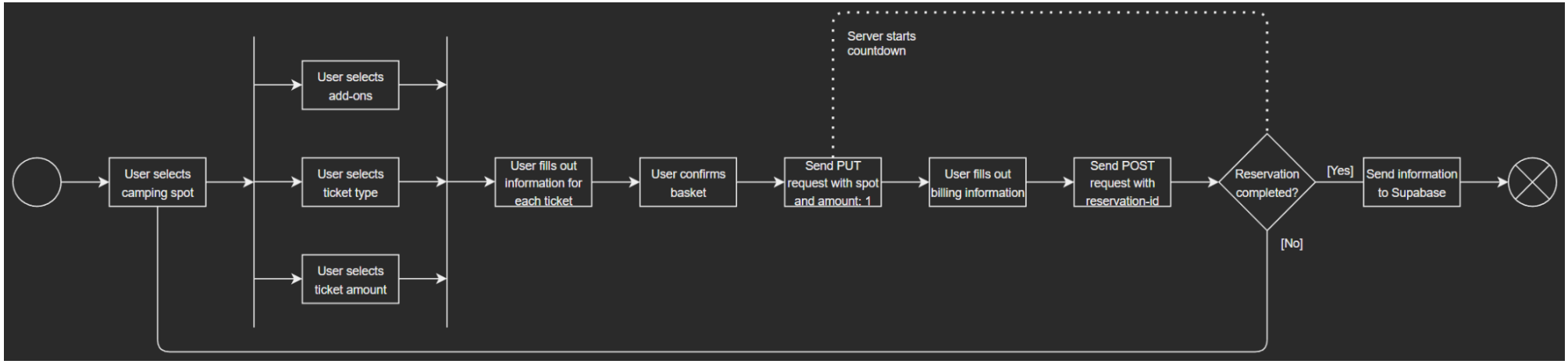
PROGRAM:



ABOUT:



BILAG G,


Server starts countdown
User selects add-ons
User selects camping spot
User selects ticket type
User selects ticket amount
User fills out information for each ticket
User confirms basket
Send PUT request with spot and amount: 1
User fills out billing information
Send POST request with reservation-id
Reservation completed?
[Yes]
[No]
Send information to Supabase