Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Magnar Dybwad Dæhli
Tore Andre Haugstad Orheim

Design og utvikling av nettside, med fokus på søkemotoroptimalisering

Bacheloroppgave i Informasjonsbehandling
Veileder: Atle Olsø
August 2023

NTNU
Kunnskap for en bedre verden

Magnar Dybwad Dæhli
Tore Andre Haugstad Orheim

Design og utvikling av nettside, med fokus på søkemotoroptimalisering

Bacheloroppgave i Informasjonsbehandling
Veileder: Atle Olsø
August 2023

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU

Kunnskap for en bedre verden

Gruppe 069: Magnar Dybwad Dæhli, Tore Andre Orheim

Jarle Hjelen AS

Hovedrapport

Forord

Denne bacheloroppgaven er skrevet som en avsluttende oppgave for studieretningen informasjonsbehandling på instituttet for datateknologi og informatikk ved Norges teknisk-naturvitenskapelige universitet. Oppgaven er utført med utgangspunkt i oppgave 069, "Utvikling/design av nettside produsert av Jarle Hjelen AS". Gjennom denne rapporten vil en få en oversikt over oppgaven, endringer av oppgaven, svar på oppgaven i form av systemutvikling, i tillegg vil problemstillingen bli besvart og tilknyttet teori.

For studentgruppen, var denne oppgaven interessant av flere grunner, da den i hovedsak åpnet for muligheter til å vise kompetanse innen systemutvikling, samt for å lære ved å grave dypere inn i ukjente territorier.

Vi ønsker å gi en spesiell takk til følgende for deres veiledning og støtte gjennom bacheloroppgaven:

- Atle Olsø, universitetslektor på instituttet for datateknologi og informatikk ved NTNU for sin rolle som veileder og all tid brukt på våre møter sammen og raske svar på spørsmål.
- Jarle Hjelen og Wilhelm Dall, for deres rolle som kunde og veiledere i prosjektet.

Signert:

22/05/2023	Stadsing Dahls Gate 16A	Magnar Dybwad Dæhli
22/05/2023	Anne Hogstads Veg 18	Tore Andre Orheim

Oppgavetekst

Oppdragsgiver Jarle Hjelen AS er utvikler, produsent og leverandør av EBSD (Electron Backscatter Diffraction)-detektorer for elektrondiffraksjon i SEM (Scanning Electron Microscopes). For å fremme kontakt med nye kunder er det et behov for å opprette en nettside som både lar Jarle Hjelen AS nå, og opprette en enklere kontaktmulighet for kunder angående produkter og tjenester som Jarle Hjelen AS tilbyr.

Nettsiden som Jarle Hjelen AS er ute etter en brukervennlig nettside som kan sees både på datamaskin og på mobile enheter, den skal også være søkemotoroptimalisert og generere "leads" for å fremme kontakt med kunder. Siden skal kunne markedsføre produkter og tjenester som oppdragsgiver leverer og skal informere godt om disse. Selv skal oppdragsgiver kunne gjøre oppdateringer på nettstedet uten å måtte endre kildekode.

Endringer av oppgavens innhold

Oppgaven er utredet av Jarle Hjelen AS, med revidering gjennom diskusjon mellom oppdragsgiver og studentgruppen. Oppgaven utføres som bacheloroppgave (fagkode INFT-2900) ved NTNU sin studieretning Informasjonsbehandling, ITBAINFO

Sammendrag

Jarle Hjelen AS er et firma som forhandler EBSD-detektorer, og tilbyr tjenester innenfor denne bransjen. De er på utkikk etter et system der de kan markedsføre sine produkter og tjenester på nettet for å nå ut, og fremme kontakt med nye og eksisterende kunder. Dette krever at systemet er godt søkemotoroptimalisert slik at kunder finner fram til nettstedet på en organisk måte, samtidig er det viktig at systemet er brukervennlig, og at informasjonen kommer fram på rett måte. Sammen med disse behovene fra Jarle Hjelen AS er det også nødvendig at endringer på nettsiden kan utføres uten å gå inn i kildekoden.

For å løse disse utfordringene er det utviklet et system som kombinerer Next.js med Sanity, for å bygge en godt søkemotoroptimalisert og dynamisk nettside. Next.js er bygd for å skape et utviklingsmiljø som gjør det lettere å utvikle en dynamisk nettside som er rask, tilgjengelig på de fleste enheter, og sist men ikke minst, gjør det enklere å implementere god og effektiv søkemotoroptimalisering.

For å få implementert og utnyttet god søkemotoroptimalisering bruker systemet flere teknikker: Sitemaps, nøkkelord, element-hierarki og metadata er noen av disse. Dette fører til at domenet: https://nordif.com, er synlig og lett tilgjengelig på Google.

Innholdsfortegnelse

Figurer

Akronymer

SEO, Search Engine Optimization.
HTML, Hyper Text Markup Language.
DOM, Document Object Model.
CSS, Cascading Style Sheets.
JS, JavaScript.
TS, TypeScript.
WCAG, Web Content Accessibility Guidelines.
OWASP, Open Worldwide Application Security Project.
HTTP, Hypertext Transfer Protocol.
SERP, Search Engine Result Page.
API, Application Programming Interface.
UX, User Experience.
SSR, Server-Side Rendering.
SSG, Static Site Generation,
ISR, Incremental Static Regeneration
EBSD, Electron Backscatter Diffraction.
SEM, Scanning Electron Microscope.
CI:CD, Continuous Integration : Continuous Delivery/Development.
CSM, Content Management System.
NPM, Node Package Manager.

Ordliste

Rammeverk: Et sett med ressurser og verktøy for å bygge/utvikle/vedlikeholde applikasjoner, tjenester og nettsteder.

SCRUM: Et smidig prosess rammeverk for produktutvikling.

Pipeline: En automatisert prosess som utfører definerte oppgaver, ofte innenfor CI:CD, der oppgavene kan være: bygging, testing, og integrering/utrulling av kode.

Revisjonshistorikk

Dato	Versjon	Beskrivelse	Forfatter
16.02.2023	0.1	Oppsett av kapitler, Oppgavetekst	Tore Orheim
20.05.2023	0.2	Innfylling av flere felter	Magnar Dybwad Dæhli, Tore Orheim
22.05.2023	1.0	Klart for innlevering	Magnar Dybwad Dæhli, Tore Orheim

Tabell 1: Revisjonshistorikk

1 Introduksjon og relevans

1.1 Bakgrunn for oppgaven

Jarle Hjelen AS er et firma som utvikler, produserer og selger EBSD-detektorer, og tilbyr tjenester innenfor denne bransjen. For å kunne nå ut til nye kunder har de et behov for en nettside, der de kan fremme kontakt, og markedsføre sine produkter og tjenester. Firmaet har hatt en nettside tidligere, men denne har ikke vært på nettet på en god tid. Derfor ønsker firmaet at det blir utviklet et system der de kan markedsføre produktene sine til kunder, samtidig skal det være lett å utføre endringer på systemet uten å gå inn i kildekode.

1.2 Problemstilling

De trenger et system der de kan markedsføre sine produkter og tjenester på nettet for å nå ut til nye og eksisterende kunder. Dette krever at systemet er godt søkemotoroptimalisert. En av de største utfordringene vil være å kunne få bedre rangering på SERP enn konkurrenter som allerede ligger på nett. Derfor oppstår det en klar problemstilling: Hvordan kan vi implementere, og i tillegg utnytte SEO får å få en gunstig plassering i forhold til konkurrenter på en ny nettside?

1.3 Struktur av hovedrapport

Rapporten er delt opp i 8 kapitler.

Kapittel 1 - Introduksjon går inn på oppgaven på overordnet grad, her er oppgaveteksten formulert med endringer som er utført i løpet av arbeidet. I tillegg til dette er problemstillingen for rapporten introdusert.

Kapittel 2 - Teori

Kapittel 3 - Valg av teknologi og metode

Kapittel 4 - Resultater formulerer resultatene knyttet til det vitenskapelige, ingeniørfaglige og administrative.

Kapittel 5 - Diskusjon utgir drøfting og refleksjon rundt resultatene definert i kapittel 4.

Kapittel 6 - Konklusjon og videre arbeid svarer på problemstillingen(e) og går inn på videre arbeid knyttet til utvikling av oppgaven.

Kapittel 7 - Referanser gir oversikt over kildene som er utnyttet.

Kapittel 8 - Vedlegg presenterer vedleggene til rapporten.

2 Teori

Hensikten med dette kapittelet er å gi en forklaring på det teoretiske arbeidet som er nødvendig for å løse problemstillingen fra kapittel 1.2, samt for å utvikle systemet etter avtalte krav.

2.1 Systemutvikling

2.1.1 Rendering

Rendering er den prosessen der kode, som har blitt skrevet i språk som blant annet React, blir konvertert til HTML, som igjen blir vist på nettsiden. Det finnes flere ulike metoder å utføre denne prosessen, der hovedskillet går mellom rendering på serveren (Pre-Rendering) og klienten (Client-Side Rendering). I tillegg skilles det mellom om denne prosessen ved "build time" eller "runtime". Build time vil si det tidspunktet man laster opp koden til serverene, der Next.js optimaliserer og konverterer koden om til det formatet som er lesbart av servere, som blant annet HTML, CSS og Javascript. Runtime (også kalt request time) henviser til det tidsintervallet en bruker er inne på nettsiden, og koden fra build time blir gjenbrukt. Next.js har, ved tidspunktet denne besvarelsen ble skrevet, tre ulike metoder for renderingprosessen, og disse er: Client-Side, Server-Side, samt Static Site Generation.

2.1.1.1 Client-Side Rendering (CSR)

Dette er den metoden som blir benyttet av en standard React-applikasjon. Den går ut på at nettleseren mottar et tomt HTML skall, sammen med instruksjoner i form av Javascript, fra serveren. Dette kan ta relativt lang tid, og fører til at brukeren må vente på at nettleseren konstruerer nettsiden på klientsiden for hver gang den blir åpnet. Heldigvis finnes det en bedre måte å utføre dette på.

2.1.1.2 Server-Side Rendering (SSR)

I likhet med rendering på klientsiden er siden nødt til å bli konstruert ved hver forespørsel (request time), men i motsetning gjøres denne prosessen her på serversiden. Dette vil si at serveren først konverterer den tilsendte dataen om til statisk HTML og sender den til brukeren, da uten Javascript. Deretter vil React ta i bruk den tilsendte dataen i form av JSON, samt Javascript for å gjøre nettsiden interaktiv, i en prosess som blir kalt for hydration. Brukeren vil altså oppleve denne metoden som raskere enn CSR, da det tar kortere tid før det visuelle til nettsiden blir vist. Med Next.js kan denne metoden benyttes dersom man tar i bruk den innebygde funksjonen "getServerSideProps".

2.1.1.3 Static Site Generation (SSG)

HTML'en blir her, i likhet med SSR generert på serveren, men i motsetning er det ved bruk av denne metoden ingen server aktiv ved kjøretid. Innholdet blir da generert en gang, hver gang nettsiden blir konstruert, altså når nettsiden blir deployert. Dataene blir lagret i en såkalt CDN (Content Delivery Network), som igjen blir brukt ved hver forespørsel. Dette

medfører at brukeren i praksis vil oppleve nettsiden som en statisk nettside, selv om den er konstruert på serveren. Dersom nettsiden bruker dynamisk data, som for eksempel henter data fra en database, eller gjennom et CMS, vet ikke nettsiden automatisk all dataen som skal benyttes. Gjennom den nyeste oppdateringen til Next, 13.4, kan man enkelt løse dette problemet ved å benytte seg av den innebygde funksjonen "generateStaticParams". Dette gjør at man på forhånd kan la serveren vite alle de mulige sidene som brukerene kan navigere seg til på nettsiden. Grunnen til at dette er gunstig er at disse sidene vil da bli konstruert ved build time og ikke ved hver forespørsel, slik som SSR. Static Site Generation er dermed på mange måter en god kombinasjon av flere ulike

2.1.1.4 Incremental Static Regeneration (ISR)

Dette er en strategi som gjør det mulig å lage eller oppdatere statiske sider etter siden er konstruert. (Next, 2023). Strategien går ut på at man kan utnytte statisk generering av sidene, på hver enkelt side, uten å måtte konstruere hele nettsiden på nytt. I praksis fungerer det ved at nettsiden benytter data som blir lagret i cache, som kan valideres på nytt, ved gitte intervaller. I de nyeste versjonene av Next.js kan denne funksjonaliteten implementeres ved hjelp av det innebygde "fetch" API'et. Dersom det ikke er gunstig, eller mulig å benytte seg av dette for å hente dataene, kan man isteden eksportere en konstant med navnet "revalidate", som angir det tidsintervallet at ny data skal hentes fra databasen, i form av sekunder.

2.1.2 CI:CD

En CI:CD pipeline er en prosess som sjekker koden en utvikler ved å automatisk utføre prosesser ved integrering og/eller utrulling. I systemet som er utviklet er det tatt i bruk to CI:CD prosesser, en systemspesifikk prosess gjennom GitLab, og Vercel sin prosess som kjører automatisk ved hver integrering.

GitLab sin CI:CD prosess er diktert av utviklere med en gitlab-ci.yml fil, i denne filen blir det definert hvilke prosesser og hvordan disse prosessene skal utføres, samt i hvilket miljø prosessene skal utføres. For en dypere gjennomgang i prosessen for dette systemet se vedlegg C, Systemdokumentasjon kap. 10.1.

Vercel sin CI:CD prosess er integrert inn i GitLab og gjør det mulig å systematisk rulle ut produksjonsmiljøet, dette skjer ved at GitLab legger merke til at en forespørsel forsøker å kombinere en utviklings-gren inn i produksjons-grenen. Når en slik forespørsel blir oppdaget kjører Vercel sin CLI for å bygge et produksjonsmiljø, etter dette laster CI:CD-prosessen produksjonsmiljøet til Vercel og danner en såkalt "preview deployment". Når kombinasjons-forespørselen er godtatt og utført, vil produksjonsmiljøet bli rullet ut til Vercel og oppdatere nettsiden med ny kildekode. Dette gir også mulighet for rask tilbakekalling ved feil i produksjonsmiljøet, da vi kan raskt gå tilbake til en tidligere versjon ved bruk av versjonskontrollen til GitLab, mer om versjonskontroll i Kap. 3.4.

2.2 Sikkerhet

Sikkerheten til applikasjoner som ligger på nettet er alltid viktig å være bevisst på, av denne grunnen tar systemet et utgangspunkt i OWASP sin sikkerhetsstandard, slik at systemet er utviklet robust.

2.2.1 OWASP

OWASP, eller Open Worldwide Application Security Project, er et ideelt firma, som kontinuerlig jobber for å fremme sikkerheten til programvare. For å fremme denne sikkerheten, jobber OWASP sammen med medlemmer, og resten av nettet på flere open-source kampanjer. Resultatet av slike kampanjer er en kontinuerlig vedlikehold av flere standarder, en av disse er OWASP Top 10, en sikkerhetsstandard som utviklere av nett-applikasjoner kan ta i bruk, denne standarden omhandler en overordnet rangering av de største sikkerhetshullene for nettapplikasjoner.

2.3 Brukervennlighet

2.3.2 WCAG

WCAG er et sett med retningslinjer for tilgjengelighet av webinnhold. Den nyeste utgaven (WCAG 2.1) består av fire prinsipper, understøttet av 13 retningslinjer og 78 testbare suksesskriterier. De fire prinsippene er: mulig å oppfatte, mulig å betjene, forståelig, samt robust. Eksempler på disse retningslinjene er "innhold skal kunne tilpasses", "siden skal være navigerbar", samt "innholdet skal være leselig og forståelig". (Uutilsynet, 2023)

2.4 Utviklingsmetodikk

2.4.1 Smidig utvikling

Smidig utvikling er et begrep som brukes om en programvareutviklingsstrategi, som omhandler rask og fleksibel utvikling for å oppnå ønsket resultat. Strategien er en såkalt iterativ programvareutviklingsprosess som tar i bruk metoder som blant annet Scrum og Kanban. Filosofien går i all hovedsak ut på at man skal stykke opp utviklingen og levere verdi med korte intervaller (Cognizant, n.d). Dette forsikrer at man jevnlig kan realisere verdi, samt få tilbakemeldinger på det man leverer, for å eventuelt gjøre justeringer i arbeidet.

2.4.2 Scrum

Scrum er en variant av smidig utvikling som går ut på at man deler opp arbeidet i intervaller på om lag 2-4 dager, såkalte sprinter. Denne metodikken åpner dermed for jevnlig tilbakemelding, samt at man får prioritert og jobbet med de viktigste tingene først, slik at man ikke bruker tid og ressurser på det som ikke er verdt investeringen (Computas, n.d). I et større team har man også gjerne ulike roller, som Scrum master og Agile coach, som har som oppgave å lede og bistå i arbeidet med Scrum.

2.5 Søkemotoroptimalisering

2.5.1 Hva er SEO?

Søkemotoroptimalisering er en praksis der en øker organisk trafikk fra SERP (søkemotor resultatside), dette får en til ved å komme høyere på rangeringen til søkemotorer, som Google. For å komme høyere på rangeringen, er det i hovedsak fire punkter som søkemotorer bruker for å gi en rangering av nettsider: Crawling, Indexing, Rangering, og UX. Dersom en ikke kommer godt ut på SERP, kan det være grunnet: konkurrenter har bedre innhold på sidene sine, dårlig eller feil bruk av nøkkelord, dårlig konstruerte lenker, sidene dine laster tregt, eller at brukeropplevelsen er dårlig.

2.5.2 Hvordan utnytte SEO?

For å kunne utnytte SEO på en god måte er det viktig å holde orden i både on-page, og off-page SEO teknikker. Off-page SEO teknikker er nevnt men kommer ikke til å være brukt i løsningen av oppgaven, da dette inkluderer tiltak som er ut av kontroll for utviklingsprosessen, samt tiltak som krever betaling.

Da en del av søkemotoroptimalisering består av god brukeropplevelse, er det viktig å ha en oversikt over markedet. I 2022 stod internasjonale søk fra mobile enheter for 59% av organisk trafikk (StatCounter, n.d-a). Søk fra mobile enheter i Norge derimot, står bare for 34% av organiske søk i 2022 (StatCounter, n.d-b). Dette betyr at det er viktig å passe på at systemet er godt, og intuitivt å bruke på mobile enheter.

2.5.2.1 On-page SEO teknikker

Som nevnt er det crawling og indexing viktige deler av rangering på SERP, disse delene kan en påvirke ved bruk av on-page SEO. On-page SEO teknikker er teknikker som kan implementeres på on-site nivået er: bruk av nøkkelord, tagger, html-hierarki, url-formatering, lenkebygging, sitemaps. En av de viktigste on-page teknikkene er nøkkelord, for å kunne øke rangeringen sin på SERP er det viktig å bruke nøkkelord som er relevante for siden din. Nøkkelordet som velges bør ha et godt søkevolum da om nøkkelordet ditt ikke søkes til vil det bli vanskelig å skaffe trafikk gjennom dette søkeordet. Vi tar i bruk Google ADsense for å gjøre nøkkelordundersøkelse, i tillegg til dette kan Ahrefs eller Semrush også tas i bruk, men dette er betalte løsninger.

2.5.2.2 Off-page SEO teknikker

Off-page SEO teknikker er teknikker som tar effekt utenfor den direkte kontrollen som utviklere har på en gitt nettside, altså det er noen utenfor (derav off-page), som bidrar til å øke/senke rangeringen til nettsiden. Teknikken som bidrar mest med off-page rangering er baklenker (backlinks). Baklenker dannes når artikler utenfor ditt nettsted linker til din nettside. Kvaliteten og mengden av sider som linker til nettstedet ditt bidrar til rangeringen din på SERP. I tillegg til baklenker er også markedsføring på sosiale medier med på å bidra på rangeringen din.

3 Valg av teknologi og metode

3.1 Frontend | Tore og Magnar

3.1.1 Next.js

Da et av ønskene til oppdragsgiveren var at nettsiden skulle gi en god plassering i søkemotorer, var valget av front-end rammeverk ganske enkelt, og det falt på Next.js. Dette rammeverket er meget godt egnet til utvikling av nettsider som for eksempel må være raske og responsive. Dette får Next.js til på en god måte gjennom en strategi som har fått navnet SSG. SSG betyr at innholdet til nettsiden blir generert på serveren når nettsiden konstrueres, istedenfor på klienten. Dette fører igjen til svært kort ventetid for brukeren, siden klienten ikke er nødt til å vente på generering av sidene som blir navigert til i systemet.

3.1.2 TypeScript

JavaScript er i utgangspunktet et dynamisk programmeringsspråk, som betyr at en ikke trenger å sette datatyper til variabler som blir tatt i bruk. Dette gjør JS til et "løst" programmeringsspråk. En kan overkomme problemer som kan oppstå ved dette, ved å ta i bruk TS, siden det er en utvidelse av JS som endrer programmeringsspråket til å være "strengt". Det vil si at vi nå er nødt til å forutse hvilken datatype variablene vi oppretter kommer til å være, samt at vi ikke kan endre en variabel fra en datatype til en annen. Dette gjør utvikling med TS lettere i den forstand at en kan oppdage eventuelle problemer før/eller under kompilering, før en kjører koden. TS er også selvdokumenterende, som betyr at det er mindre nødvendigheter til kommentarer i koden.

3.1.5 Tailwind CSS

Tailwind CSS er et CSS-rammeverk som lar utviklere skrive CSS-regler i klassenavn til DOM elementer, og automatisk skriver disse reglene til en egen .CSS fil, samt fjerner unødige CSS-regler. Dette er gunstig for å effektivisere utviklingen av det estetiske av nettsiden. I motsetning til rammeverk, som Bootstrap, som utnytter ferdiglagde "designmaler" for elementene på nettsiden, har Tailwind ingen slike forhåndslagde komponenter. Fordelene er nettopp at det er så modulært, at man tilnærmet kan utvikle produktet fullstendig slik man ønsker det, samtidig som syntaksen for å skrive klassene er svært enkel. Tailwind har også en åpen kildekode, noe som gjør det ekstra attraktivt for utviklere å ta det i bruk, siden man har et mye bedre innblikk i teknologien man faktisk bruker.

3.2 Tester

3.2.1 Cypress

Cypress er et open-source test-rammeverk som lar utviklere skrive komponent- og E2E-tester. Under utvikling, og i en CI:CD-pipeline kan test-utføringen tas opp slik at en kan gå gjennom tester som har feilet og analysere disse. I utviklingsprosessen av dette systemet, har vi tatt i bruk den lokale applikasjonen, og Cypress Cloud for å hente opptak av testene.

3.3 Backend

3.3.1 Sanity.io

Sanity er et Content Management System (CMS) som vil si at det er et system for å enkelt opprette, redigere og slette innhold som kan brukes på blant annet nettsider. Systemet kan både være helt separert fra produktet der dataen blir brukt, såkalt headless, eller integrert sammen med produktet, såkalt traditional, eller monolithic. Det er flere grunner til at vi valgte nettopp Sanity for å håndtere dataen til prosjektet, blant annet fordi selskapet bak har et tett samarbeid med Vercel, som igjen vedlikeholder Next.js-rammeverket. Dette er et svært viktig punkt, både fordi det gjør utviklingsprosessen mer smidig, samt at det sikrer det langsiktige aspektet med både vedlikehold og videreutvikling. I tillegg visste vi at abonnementet som var gratis inkluderte en høy månedlig kvote for å hente data, noe som naturligvis er svært ettertraktet for slike tjenester.

3.4 Versjonskontroll

3.4.1 GitLab

Systemet tar i bruk GitLab som versjonskontroll, da det er dette vi er blitt mest kjent med gjennom studiet vårt. Mange av GitLabs tjenester er gunstige for utvikling og vedlikehold av systemet, blant annet en robust CI:CD-pipeline, samt en god integrering med Vercel som vi bruker som host av domenet og for selve nettsiden.

3.5 Hosting | Magnar

3.5.1 Vercel

Vercel er en såkalt Platform as a Service (PAAS) som er bygget rundt JAMstack-arkitekturen som er en stack som inneholder både Javascript, API og Markup. At de er en PAAS vil si at det er de som har ansvaret for både skytjenestene, programvaren og operativsystemet, samt datasenteret. Dette er da svært gunstig, spesielt for en relativt liten bedrift, da man selv ikke trenger å ha ansvaret for noe hardware for tjenesten. Grunnen til at vi har valgt å bruke Vercel er at det er de som vedlikeholder Next.js-rammeverket, som vi også tar i bruk for utviklingen av systemet. Vercel har også et lett forståelig dashboard der en enkelt kan hente ut ulik informasjon om ulike funksjonaliteter, blant annet brukernes geografiske

fordeling, hvilke sider på nettsiden de navigerer seg til, etc. I tillegg til dette har de også nylig blitt gode samarbeidspartnere med Sanity, som vi bruker som vårt innholdshåndteringssystem. Vercel har også tre ulike abonnementer, der den billigste er gratis, dersom man ikke overskrider den månedlige kvoten man får tildelt. Alle disse punktene, med at denne løsningen både var et samlingspunkt for majoriteten av teknologiene vi valgte å bruke til prosjektet, samt at det var svært lav kostnad, gjorde valget enkelt. Det at Vercel og Sanity nylig inngikk et samarbeid, forsikrer også det langsiktige aspektet med både vedlikehold og mulig utvidelse av nettsiden.

4 Resultater

4.1 Vitenskapelige resultat

Søkemotoroptimalisering er tidkrevende, ikke implementeringen, men ventetiden for å se resultater av det som er blitt implementert. Det kan ta flere uker før Google sine roboter går gjennom domenet. I tillegg til dette er det vanskelig å utføre en vurdering av hvor effektiv søkemotoroptimaliseringen har vært.

4.1.1 Hvordan måle SEO

Det kan være vanskelig å måle hvor godt en har fått utnyttet SEO, da en ikke får vite hvordan en er rangert mot sine konkurrenter. Uansett finnes det KPI-er som kan utforskes for å finne ut til hvilken grad SEO-kampanjen har fungert, disse KPI-ene er: plassering på SERP, Besøkende på nettstedet, tid per bruker på siden, hvilke sider bruker besøker, og til sist, drar bruker etter første landing.

Google Analytics kan brukes for å få et overblikk over organisk trafikk til nettstedet, målet er å få en økning i organisk besøkende etter implementering av SEO-kampanjen. I tillegg til å måle hvor trafikken kommer fra, er det også viktig å se på hvordan nettsiden blir brukt av brukerne. Viktige punkter her er hvor mange sider som brukerne besøker, hvor lang tid de bruker på disse, og om det er mange som kommer inn på siden og drar umiddelbart.

Gjennom Google search console kan det utforskes hvordan nøkkelordene som er i bruk har utviklet seg, her er det viktig å sjekke om nøkkelordet fortsatt er relevant eller om vi burde gjøre endringer. Samt kan en også få en oversikt over sidene som har/eller ikke har blitt crawled eller indexet.

For å sjekke hvor gode baklenker en har, kan en ta i bruk ahrefs eller Moz link explorer, disse verktøyene kan gi informasjon om hvor baklenker kommer fra, samt kan det også utforskes om noen baklenker kommer fra sider som ikke er trofaste, eller ukjente sider. Det er viktig å være oppdatert om hvem som lenker til domenet for å sikre at det ikke faller på rangeringen basert på tredjepartsoppførsel.

4.1.2 SEO teknikk-tabell og implementerte teknikker

Tabell 4.1 viser SEO teknikker som systemet har implementert, denne tabellen må gjerne brukes som veileder for hva som kan utforskes videre for å komme bedre ut på SERP.

On-page	I bruk	Off-page	I bruk
Sitemaps	Ja	Baklenker: artikler/journaler som linker til nettstedet.	Nei
Godt, relevant innhold, som samsvarer med nøkkelord	Ja	Markedsføring (sosiale medier, m.m)	Nei
HTML-element hierarki som deler opp relevant innhold på en logisk metode (h1,h2,h3, p, osv..).	Ja	Google ad-plasseringer/kjøpte plasseringer på SERP	Nei
Metadata: meta-tagger, header-tagger, alt-tagger.	Ja		
Responsivt design	ja		

Tabell 4.1: SEO teknikker

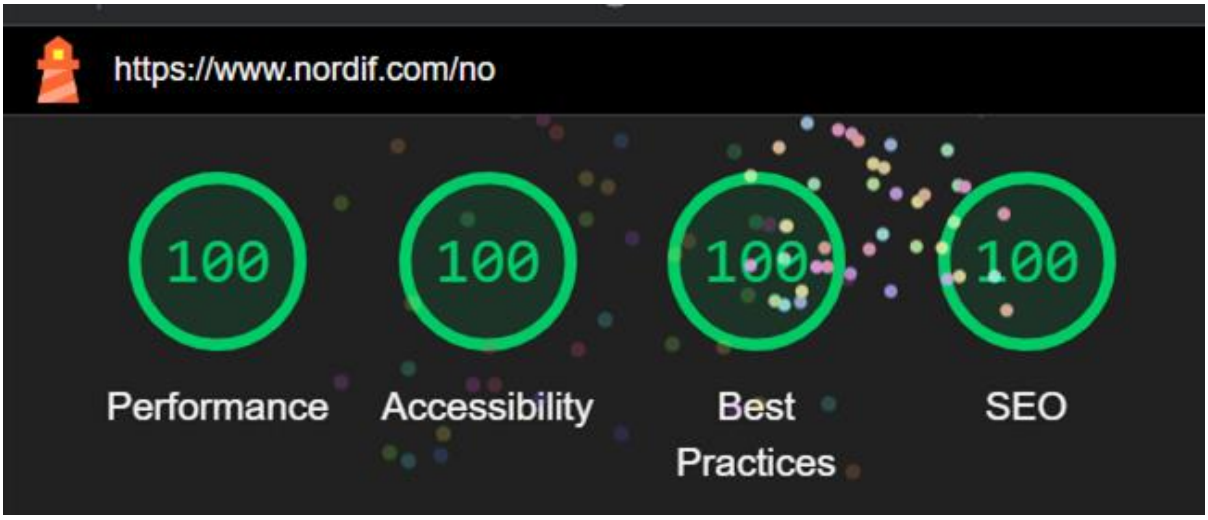
4.2 Ingeniørfaglige resultat

4.2.1 Funksjonelle krav

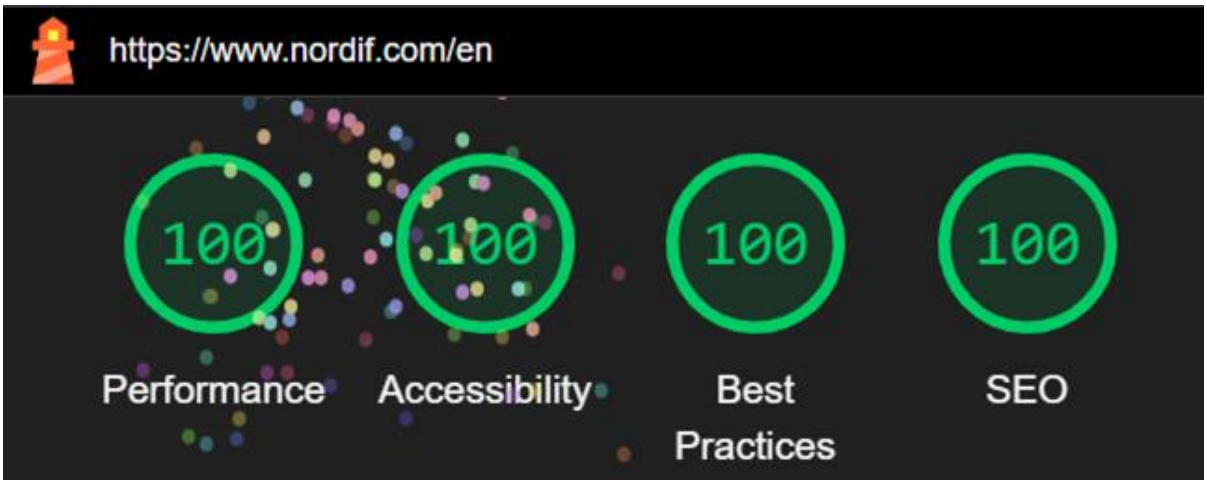
De funksjonelle kravene er satt opp i vedlegg A Visjonsdokument, kap 5. Disse kravene er dekket i systemet og er videre definert fremover.

Hjemmesiden

Det finnes flere ulike måter man kan måle kvaliteten på en nettside og dermed hvor godt den dekker de kravene som skal dekkes. En måte å teste dette på er ved å bruke Google sitt innebygde verktøy - Google Lighthouse. Dette verktøyet gir en oversiktlig og grundig vurdering av hvor godt nettsiden gjør det innenfor fire ulike hovedkategorier: **Performance**, **Accessibility**, **Best Practices**, samt **SEO**. Resultatet kommer i form av en poengsum fra 0 til 100, for hver av de fire kategoriene, basert på hvor godt nettsiden gjør det innenfor de ulike underkategoriene i hver av de. Ettersom de to hjemmesidene er det første brukerne vil se, har vi brukt Google Lighthouse til å teste disse. På grunn av at vi har implementert mange av Google sine krav for en god nettside, har vi klart å få den høyeste mulige poengsummen i alle de fire kategoriene. For dette prosjektet spesifikt, har det vært spesielt viktig å få en god score på de to punktene Performance og SEO, da disse to var svært sentrale i oppgavebeskrivelsen. Det at vi i tillegg valgte å ta for oss SEO som problemstilling, gjorde at dette var et naturlig punkt å tilstrebe best mulig poengsum på. Eksempler på funksjonalitet som må være implementert for å få en høy poengsum er blant annet at de ulike sidene har en god metabeskrivelse, at lenkene på siden er crawlable, samt at bildene på siden har et definert alt-attributt.

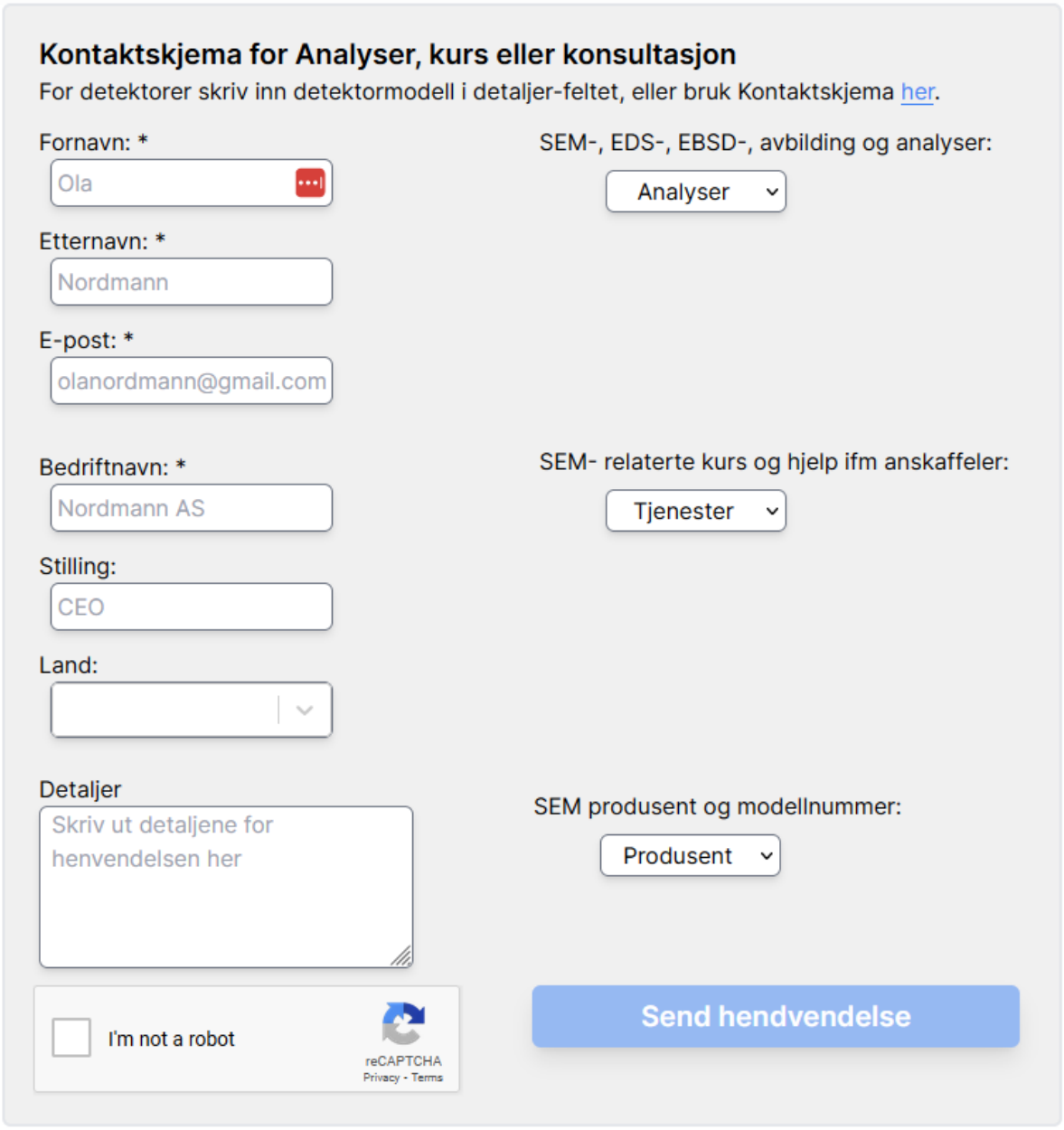


Figur 4.1 Resultat fra Google Lighthouse for the nasjonale hjemmesiden



Figur 4.2 Resultat fra Google Lighthouse for den internasjonale hjemmesiden

Kontaktskjema

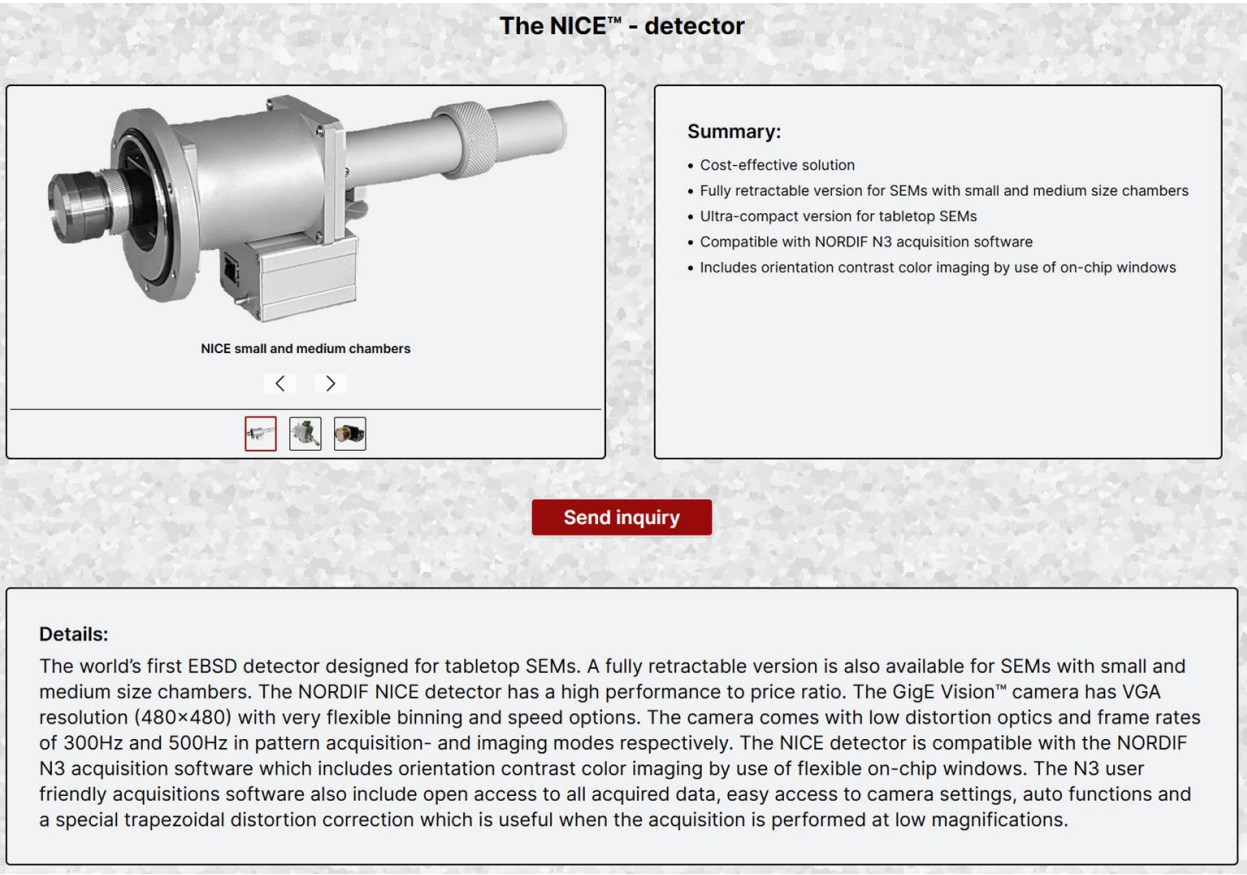


Figur 4.3: Kontaktskjema

Kontaktskjemaet som er enkelt å navigere til på detaljer siden til detektorer, analyser, tjenester og services, ligger også på egen side både på nasjonalt- og internasjonal-side. Lar brukeren om å skrive inn kontaktinformasjon, sammen med detaljene rundt henvendelsen. I tillegg til dette kan brukeren også velge hvilken analyse og/eller tjeneste henvendelsen omhandler. Dersom brukeren velger valget ANNET/OTHER kommer det fram et tekstfelt der brukeren kan skrive inn videre detaljer rundt deres valg. En lignende funksjonalitet kommer også frem ved valg av Produsent/Manufacturer, her vil en ved å gjøre et valg få fram en tekstboks for modellnummeret til SEM brukeren henvender om. Det er bare valgene

Fornavn, Etternavn, e-post og bedriftnavn som er krevde felter, i henhold til oppdragsgiverens ønsker.

Produktinformasjon



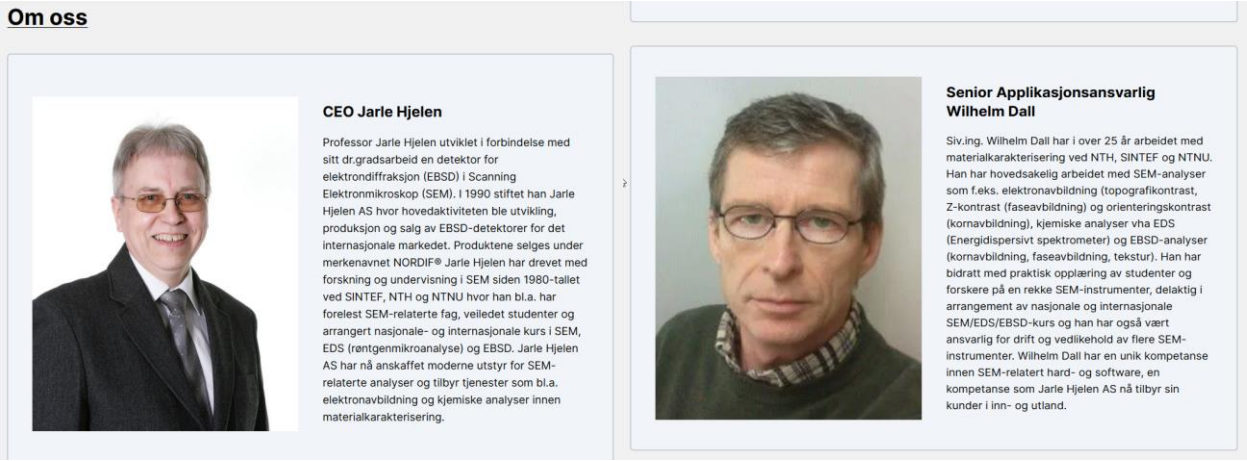
Figur 4.4: Side for produktinformasjon

Produktinformasjonssiden gir brukere en introduksjon til et produkt på en enkel, og oversiktlig måte. Her er brukerne først gitt en bildekarusell med bilder som relaterer til produktet, ved siden av har produktet et sammendrag som oppsummerer med korte punkter, hva produktet er. Under disse feltene finner en knappen for å dra til kontaktskjemaet slik at en kan sende en henvendelse til Jarle Hjelen AS. Sist på produktinformasjonssiden har den en større oversikt over spesifikasjoner, og hva produktet er i stand til.

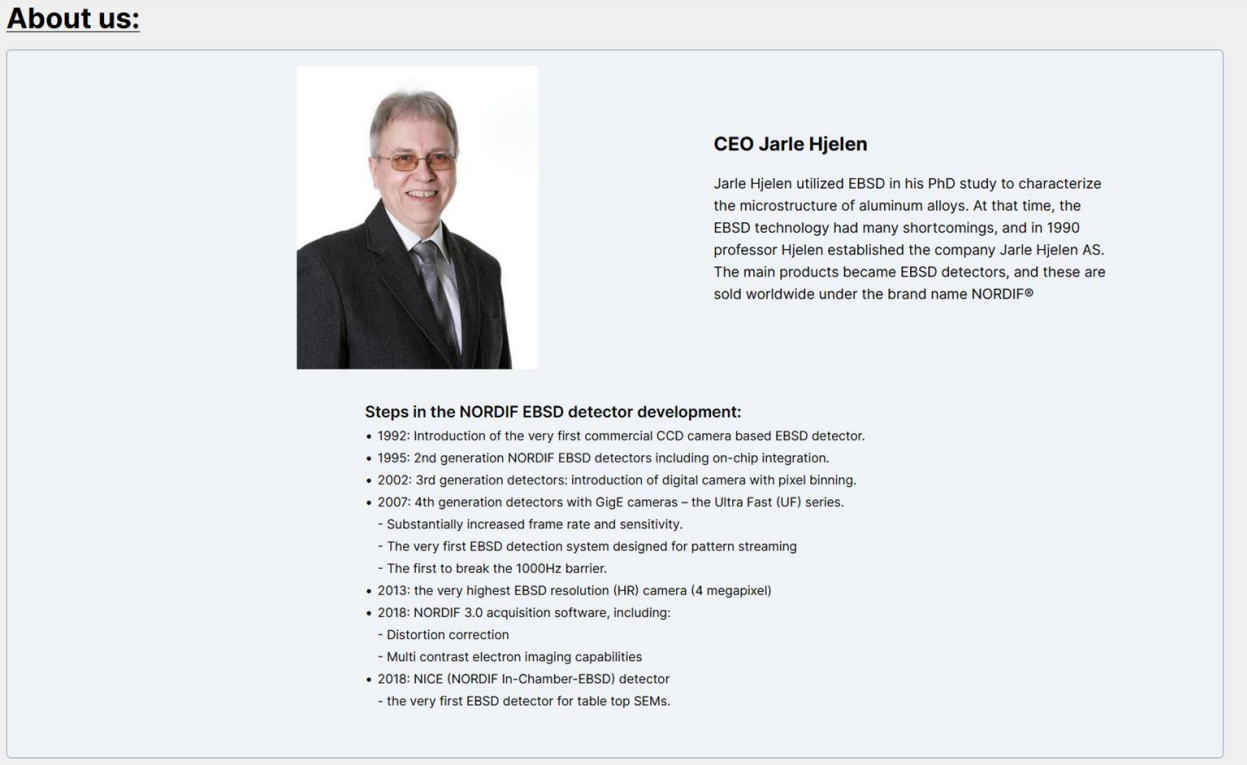
Dette oppsettet er delt på følgende sider: analyser, tjenester, services, og applications. Med bildekarusell, sammendrag, detaljer og knapp for enkel tilgang til kontaktskjemaet.

Om oss

Om oss siden sin funksjonalitet er for å gi informasjon og kunnskap til bedriftens historie og de ansatte sin kompetanse, for å bygge økt tillit og nærhet til firmaet. På den nasjonale siden er det utviklet "kort" (se figur 4.3) som inneholder hver ansatt, disse kortene inneholder stillingstittel og navn, i tillegg til et portrett og en personlig tekst knyttet til ansattes kunnskap i bransjen og deres historie og relasjon til Jarle Hjelen AS. På det internasjonale markedet er siden "About" som inneholder et portrett, stillingstittelen og navnet til Jarle Hjelen. Under dette er det en historikk for Jarle Hjelen AS sin utvikling.



Figur 4.5: Den nasjonale Om oss siden



Figur 4.6: Den internasjonale Om oss siden

4.2.2 Ikke-funksjonelle krav

4.2.2.1 Tilgjengelighet

Som satt i vedlegg A Visjonsdokument, kap 6, dekker systemet mange av tilgjengelighetskravene til WCAG 2.1, både på nivå AA og AAA.

1.1 Systemet dekker alle krav for alternativ tekst.
1.2 Systemet tar ikke i bruk tidsbegrenset media.
1.3 Systemet dekker alle krav for å være tilpasningsdyktig
1.4 Systemet dekker alle krav for kontraster mellom elementer.
2.1 Systemet er fullt brukbart med bare tastatur, inneholder ikke tastatur-fallgruver og bruker ikke hurtigtaster.
2.2 Siden er ikke tidsbegrenset, så brukere har god tid til å lese og bruke funksjonaliteter.
2.3 Systemet er utformet uten overflødige animasjoner, og animasjonene som er brukt er vurdert essensielle for funksjonalitet og presentasjon av informasjon.
3.2 Systemet er konsistent og forutsigbart ved å ikke utføre endringer i kontekst uten å gi god beskjed til brukeren.
3.3 Systemet presenterer feilmeldinger til brukere og instrukser blir anbefalt til brukere. Punkt 3.3.5 Help, og 3.3.6 Error Prevention er ikke blitt implementert.
4.1 Systemet er kompatibelt med nåværende og fremtidige bruker-assisterende teknologier.

4.2.2.2 Sikkerhet

Sikkerheten på systemet er godt ivaretatt, og utført med utgangspunkt i OWASP Top Ten 2021. For mer informasjon rundt dette temaet, se Vedlegg C Systemdokumentasjon kapittel 7.

4.2.2.3 Testing

Alle komponenter har tilhørende tester, samt er det tatt i bruk E2E tester gjennom Cypress for å teste fullstendig funksjonalitet til nettsiden. Mer om dette er definert i Vedlegg C Systemdokumentasjon kapittel 10.

4.2.2.4 Dokumentasjon

I kildekoden til systemet er det skrevet kommentarer som tillater utviklere å utnytte hjelpetekst ved videre utvikling, disse kommentarene er også kompatible med Typedoc, slik at en kan generere en html-side som inneholder dokumentasjon for kildekoden. Hvordan genereringen av denne dokumentasjonen er utført er informert om i vedlegg C Systemdokumentasjon kapittel 9.

4.2.2.5 Vedlikehold

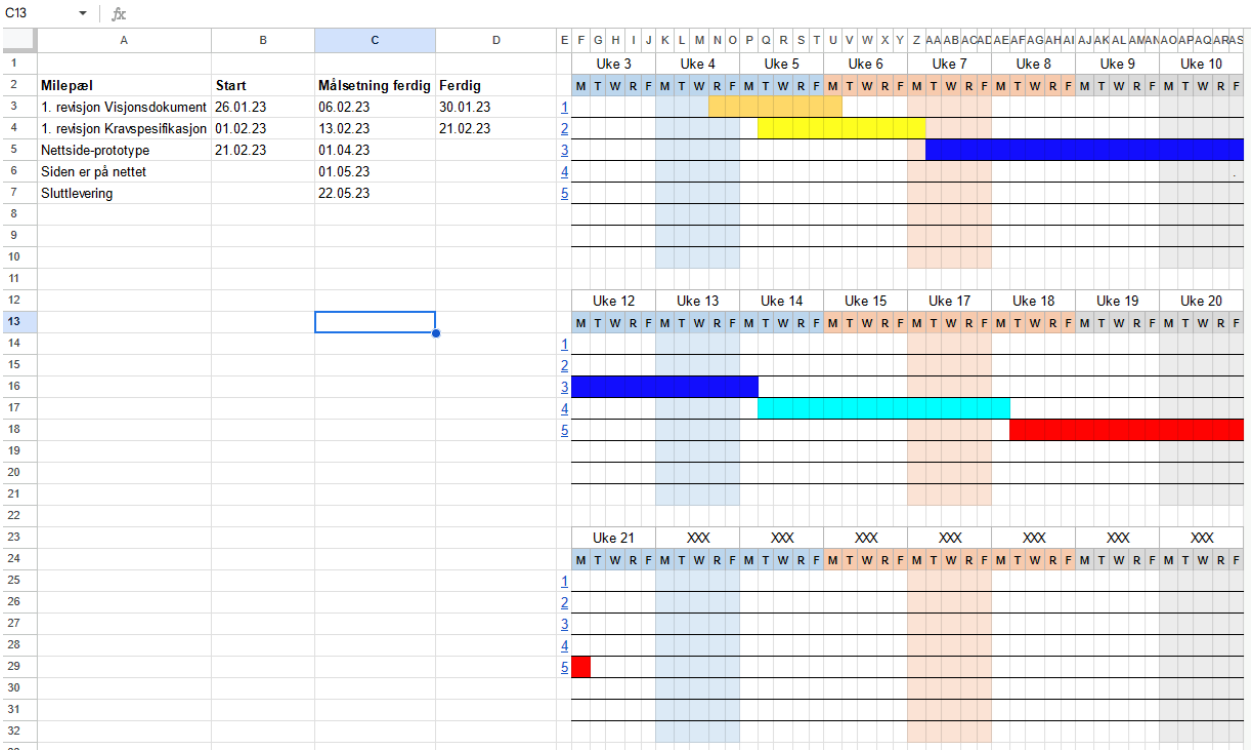
Systemet er utviklet med NextJS sin AppDir prosjektstruktur, dette gjør det lett å holde oversikt over koden, alle gjenbrukbare komponenter ligger under app/components, og endepunktene ligger i egne mapper under app/, f.eks app/About. Mer informasjon rundt prosjektstruktur, se vedlegg C Systemdokumentasjon kapittel 3.

4.3 Administrative resultat

4.3.1 Fremdriftsplan

Under oppstarten av prosjektet ble det utviklet et gantt diagram som skulle gi en klar oversikt over når og hvor fokus skal være på overordnet ukentlig nivå, dette er diagrammet

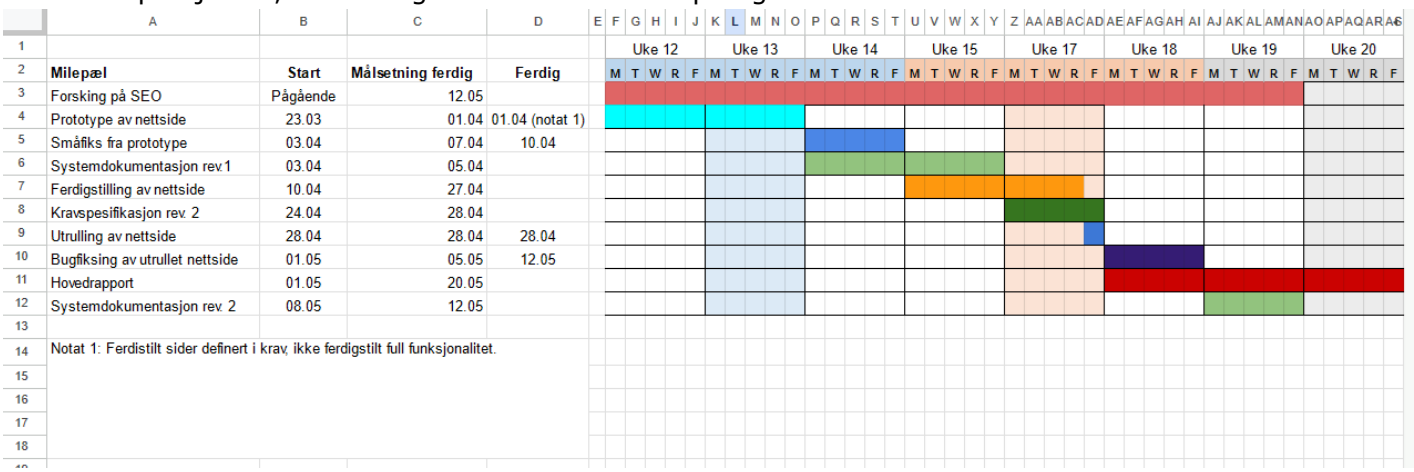
som er vist på figur 4.5.



Milepæl	Start	Målsetning ferdig	Ferdig
1. revisjon Visjonsdokument	26.01.23	06.02.23	30.01.23
1. revisjon Kravspesifikasjon	01.02.23	13.02.23	21.02.23
Nettside-prototype	21.02.23	01.04.23	
Siden er på nettet		01.05.23	
Sluttlevering		22.05.23	

Figur 4.5: Fremdriftsplan versjon 1

Dette diagrammet var ikke godt utarbeidet, og ble reiterert fra og med uke 12 og utover resten av prosjektet, dette diagrammet er definert på figur 4.6.



Milepæl	Start	Målsetning ferdig	Ferdig
Forsking på SEO	Pågående	12.05	
Prototype av nettside	23.03	01.04	01.04 (notat 1)
Småfiks fra prototype	03.04	07.04	10.04
Systemdokumentasjon rev.1	03.04	05.04	
Ferdigstilling av nettside	10.04	27.04	
Kravspesifikasjon rev. 2	24.04	28.04	
Utrulling av nettside	28.04	28.04	28.04
Bugfiksing av utrullet nettside	01.05	05.05	12.05
Hovedrapport	01.05	20.05	
Systemdokumentasjon rev. 2	08.05	12.05	

Notat 1: Ferdistilt sider definert i krav, ikke ferdigstilt full funksjonalitet.

Figur 4.6: Fremdriftsplan versjon 2

4.3.2 Timeregnskap

Under er en tabell som viser antall timer brukt på prosjektet, utvidet timeforbruk er lagt ved i vedlegg D Prosjekthåndbok, kap 2.

4.3.3 Utviklingsprosess

Utviklingsprosessen er utdypet i vedlegg D Prosjekthåndbok kapittel 3, dette kapittelet inneholder timeregistrering, aktiviteter for uken, start- og slutt-skjermbilde av Trello tavlene for hver ukentlige periode, samt også et retrospektiv pr. uke for hva som har hendt den perioden.

5 Diskusjon

5.1 Vitenskapelige resultater

5.1.1 Tidsmessige utfordringer

En av de største problemene vi møtte på som omhandlet søkemotoroptimalisering, var hvor mye ventetid som kreves for å kunne utføre vurderinger og endringer på teknikker. Pr. google sin dokumentasjon om SEO kan det ta flere uker før Google først utfører en "crawl" på nettsiden. Dette førte til forutsatt tid der ingen vurdering av implementerte teknikker kunne utføres (Google, 2023), av den grunn er det ikke lagt ved resultater av hvor godt det utviklede systemet har kommet ut av implementeringen av SEO. Som følge av dette har rapporten gått gjennom hvordan systemet har blitt rangert, og hvordan systemet kan videreutvikles for å kunne komme bedre ut på valg av nøkkelord, og andre teknikker.

5.1.2 Implementering

Som nevnt i kapittel 5.1.1, er det største problemet med tanke på implementering av SEO, tiden som prosessen tar. Men det betyr ikke at vi ikke har støtt på utfordringer underveis. Et tidlig problem var hvilke nøkkelord som vi skal prøve å sikte oss inn på, i tillegg til hvordan vi skal finne ut av om nøkkelordet har trafikk eller ikke. Tidligere i rapporten nevnes Ahrefs og Semrush, som originalt var det vi fant under forskningen på SEO, uheldigvis var dette betalte løsninger, noe som vi ville unngå, det var litt etter dette vi fant Google Adsense gjennom en Google bedriftskonto. Med Google Adsense kunne vi se hvilke relevante nøkkelord som kunne brukes for å bedre SEO-rangeringen til systemet, resultatet av undersøkelsen vår var at alle nøkkelordene som var relevante for systemet er lavtrafikk-nøkkelord, altså det er ikke mange som søker på disse, noe som vil føre til en lav gevinst for implementering. Ergo er det mulig at bruk av nøkkelord til systemet vil ha et lavt resultat for vårt spesifikt dette systemet. I motsetning er vi heldige for at det utviklede systemet er assosiert med Jarle Hjelen AS og deres NORDIF® detektorer, nøkkelordet Nordif i seg selv ligger på lik, og ofte større søketrafikk enn de andre nøkkelordene.

5.2 Ingeniørfaglige resultater

5.2.1 Funksjonelle krav

5.2.1.1 Design

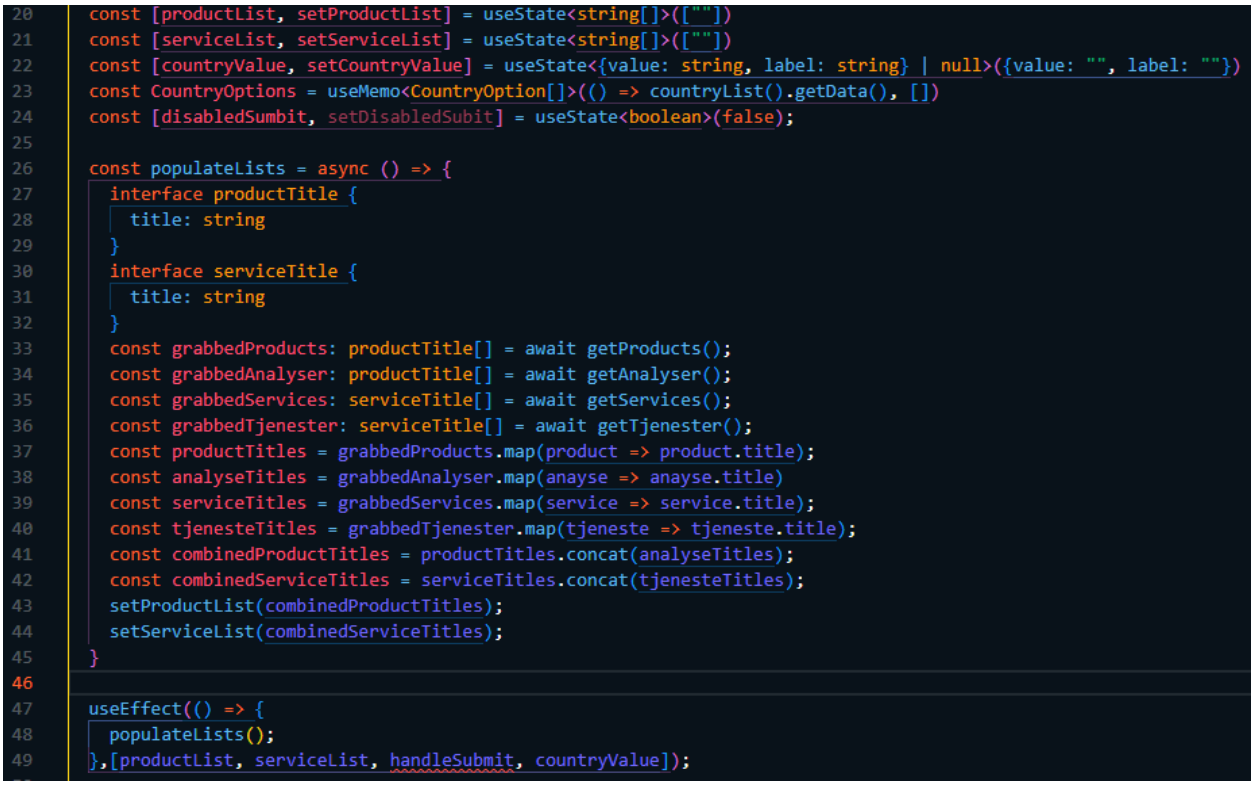
Designet av nettsiden var et viktig punkt i oppgavebeskrivelsen fra kunden. I oppstartsfasen av prosjektet produserte vi grove skisser over hvordan en eventuell nettside kunne se ut, noe kunden også mente passet bra med det de ønsket. Vi ga også et eksempel i form av en eksisterende nettside, ettersom vi ikke helt følte at den opprinnelige utformingen lot oss demonstrere våre faktiske ferdigheter innenfor produktutvikling. Eksempelet inneholdt blant annet en del animasjoner, noe som kunden senere ga uttrykk for at fjernet det profesjonelle inntrykket til nettsiden. Dette førte til at vi i hovedsak gikk tilbake til å bruke det originale designet vi hadde produsert, som kunden dog var fornøyd med.

5.2.1.2 Sanity

Ved implementering av Sanity brukes det api-kall gjennom en service.ts fil, etter å ha satt systemet tilgjengelig på nettet at vi at antallet api-kall ble skyhøye, se figur 5.1.

Figur 5.1: Sanity cache requests april

Med vår makskvote på 1 million API CDN forespørsler i måneden, måtte vi finne ut hva problemet er. Etter en del undersøking av kildekoden, viste det seg at det var en uendelig løkke som kjørte på kontaktskjema-siden. Løkka kan sees på figur 5.2, denne løkka er typisk for nyere utviklere i React-rammeverk, og fungerer slik: useEffect blir kalt automatisk når siden åpnes, og har en liste over variabler som ved endring skal kjøre en refresh på nettsiden. Funksjonen populateLists blir kjørt ved sidens åpning, denne funksjonen kjører api-kall, og setter state til serviceList og productList. Disse to statene er i listen over variabler som skal føre til en refresh på siden, som da fører til at useEffect blir kjørt på nytt.

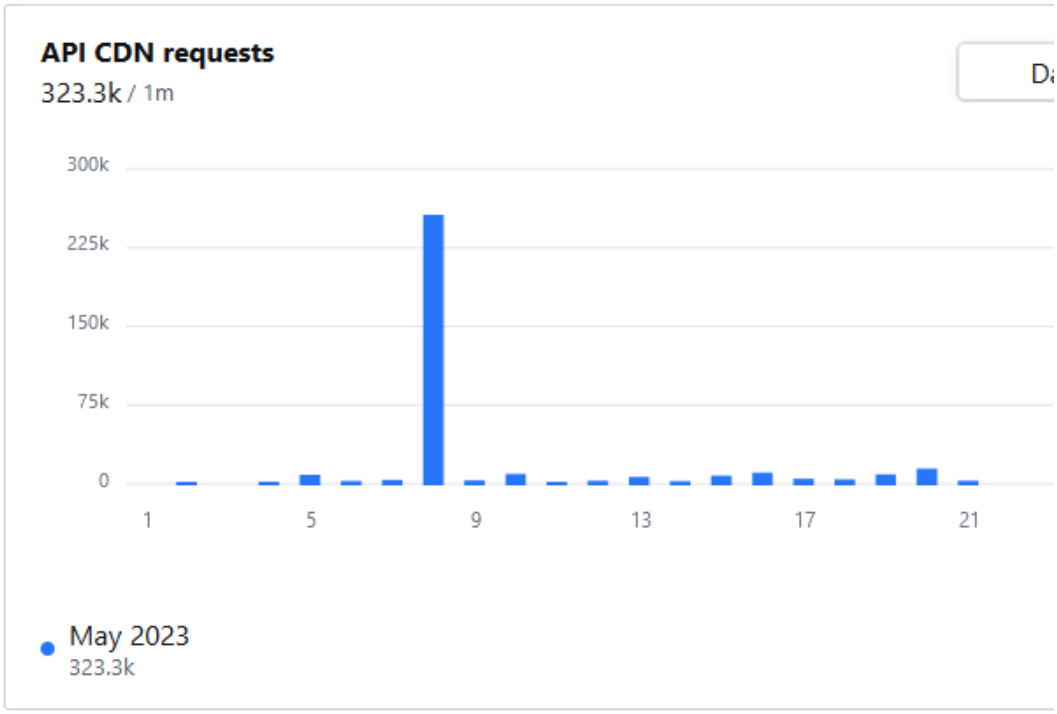


```
const [productList, setProductList] = useState<string[]>([""])
const [serviceList, setServiceList] = useState<string[]>([""])
const [countryValue, setCountryValue] = useState<{value: string, label: string} | null>({value: "", label: ""})
const CountryOptions = useMemo<CountryOption[]>(() => countryList().getData(), [])
const [disabledSumbit, setDisabledSubit] = useState<boolean>(false);

const populateLists = async () => {
  interface productTitle {
    title: string
  }
  interface serviceTitle {
    title: string
  }
  const grabbedProducts: productTitle[] = await getProducts();
  const grabbedAnalyser: productTitle[] = await getAnalyser();
  const grabbedServices: serviceTitle[] = await getServices();
  const grabbedTjenester: serviceTitle[] = await getTjenester();
  const productTitles = grabbedProducts.map(product => product.title);
  const analyseTitles = grabbedAnalyser.map(anayse => anayse.title)
  const serviceTitles = grabbedServices.map(service => service.title);
  const tjenesteTitles = grabbedTjenester.map(tjeneste => tjeneste.title);
  const combinedProductTitles = productTitles.concat(analyseTitles);
  const combinedServiceTitles = serviceTitles.concat(tjenesteTitles);
  setProductList(combinedProductTitles);
  setServiceList(combinedServiceTitles);
}

useEffect(() => {
  populateLists();
},[productList, serviceList, handleSubmit, countryValue]);
```

Figur 5.2: Kildekode med infinite loop

Når denne løkken ble oppdaget endret vi umiddelbart koden, etter dette har vi bare hatt ett til problem med mengder på api-kall, dette skjedde mellom den 8-11. Mai når en variabel med et uhell ble satt til false, istedenfor true. Under utvikling av nettsiden ser vi fra statistikken (figur 5.3) at vi ikke kommer over ti tusen api-kall per dag, og det er antatt at grensen på en million kall i måneden vil være mer enn nok til å holde siden oppdatert i god

stund fremover.



Figur 5.3: Sanity cache requests mai

5.2.2 Ikke funksjonelle krav

5.2.2.1 Tilgjengelighet

WCAG 2.1

Ved å følge WCAG sine retningslinjer så godt som mulig, kunne vi sikre at systemet er tilgjengelig for så mange som mulig. I tillegg gjorde det at vi lettere kunne skrive tester for systemet, ved å bruke arzia-knagger. Selv om vi ikke fikk implementert alle punktene innenfor WCAG 2.1 er vi fornøyde med hvor langt vi kom, og vi har fått en god forståelse over hva som kreves for å utvikle gode, brukervennlige systemer.

5.2.2.2 Sikkerhet

Vi har forsøkt å følge OWASP sin retningslinje for å sikre nettbaserte applikasjoner så godt som mulig. På likt nivå som WCAG 2.1, tilfredsstiller vi mange, men ikke alle punktene på retningslinjen.

5.2.2.3 Testing

Som nevnt tidligere, og i vedlegg C Systemdokumentasjon kapittel 10.2, har vi utviklet tester for systemet gjennom Cypress. Noe som vi oppdaget gjennom utviklingen av systemet, var at Cypress sine komponenttester ikke kan dekke komponenter som bruker

funksjoner som henter ut URL som komponenten ligger i. Dette betyr at for å få testet komponenten, er vi nødt til å ta i bruk E2E tester.

Når vi startet med prosjektet var planen å utføre brukertester på kunder av Jarle Hjelen AS, dette endte vi opp med å ikke utføre da etter lufting av ideen på et møte med Jarle Hjelen AS ble dette ikke tenkt på igjen før utrulling av systemet startet i overgangen mellom mars og april måned. Etter dette temaet kom frem burde vi ha opprettet en plan for utførelse, men dette var noe som aldri tok sted.

5.2.2.4 Dokumentasjon

Slik som nevnt i kapittel 3.1.2, er Typescript selvdokumenterende, dette gjør at vi alltid har hatt fokus på å skrive god og forståelig kode. Uansett kan det ofte være vanskelig å se for seg hvordan noen andre forstår koden som en selv skriver, så vi har låg ved kommentarer underveis. Mot slutten av prosjektet oppdaget vi et rammeverk for dokumentasjon av kildekode, TsDoc, som vi da umiddelbart tok i bruk for å forbedre dokumentasjonen vår. Dette rammeverket lar oss skrive kommentarer som blir tilgjengelige i kontekstmenyer i mange koderedigeringsapplikasjoner. I tillegg til dette tillater rammeverket oss å bruke Typedoc for å generere en dokumentasjons-nettside som inneholder kommentarene som er i gyldig TsDoc-format fra alle filene som vi definerer. Dette er veldig bra for videre utvikling av systemet og vil være til god hjelp for å forstå koden som er skrevet. Hvordan denne dokumentasjons-siden opprettes er skrevet i vedlegg C Systemdokumentasjon kapittel 9.

5.2.2.5 Vedlikehold

Når vi valgte å ta i bruk Nextjs som frontend-rammeverk bestemte vi oss å ta i bruk en nyere variant av Nextjs sin prosjektstruktur, altså appDir-struktur, istedenfor den tradisjonelle. pages-strukturen. Dette gjorde at vi var på fronten med ny dokumentasjon og lite oversikt fra andre utviklere, en spennende utfordring og en god måte for å utforske våre ferdigheter. Vi gjorde dette valget for å sikre et system med kode rettet mot fremtiden, slik at en kan unngå at deler av systemet ikke lenger er støttet.

5.2.3 Refleksjon

5.2.3.1 Styrker

For å gi en god evaluering av styrkene til prosjektet er det naturlig å gå gjennom de ønskede kriteriene som kundene presenterte i oppgavebeskrivelsen. Disse kriteriene var:

- Er brukervennlig
- Fungerer på alle enheter
- Gir gode plasseringer i søk
- Markedsfører produktet og tjenester
- Genererer "leads"
- Fremmer kontakt med kunder
- Informerer om spesifikke produkter og tjenester
- Vi skal enkelt kunne gjøre oppdateringer på nettsiden selv

De to første punktene går litt inn i hverandre, og ble løst etter beste evne, mye takket være Tailwind som gjorde det relativt enkelt å implementere en god universell utforming. Vi implementerte alle de aktuelle teknikkene tilgjengelige for best mulig SEO, gitt de forutsetningene vi hadde, noe som skal gi gode plasseringer i søk. Vi tok også flittig i bruk Google sitt innebygde analyseverktøy, Lighthouse, for å finne ut hva som var bra og hva som kunne forbedres med nettsiden.

Vi har også på en gunstig metode fått presentert produkter og tjenester, både for det nasjonale og det internasjonale markedet. I tillegg har vi implementert et fleksibelt kontaktskjema, slik at kunder enkelt kan kontakte de ansatte dersom de har noen henvendelser.

At de selv skulle kunne oppdateringer på nettsiden var også et svært viktig punkt for kundene. Dette mener vi ble løst på en svært godt måte, ved at vi tok i bruk et CMS, som gjør det svært enkelt for dem å gjøre endringer, samt for å utvide dersom det skulle bli behov for det i fremtiden. Å lage et slikt system fra bunnen av selv kunne blitt gjort, men dette hadde vært svært tidkrevende, noe som var grunnen til at vi heller valgte en allerede eksisterende tjeneste.

5.2.3.2 Svakheter

Som nevnt under kapittel 5.2.1.1 ble ikke designet på nettsiden fullstendig slik vi ønsket den, eller hadde tenkt på forhånd. Bacheloroppgaven skal være en mulighet for oss å vise vår kunnskap innenfor produktutvikling, noe vi ikke nødvendigvis føler at vi har fått gjort gjennom det designet vi endte med. På den andre siden så gjennomføres en slik oppgave som et oppdrag fra en kunde, der man skal levere et produkt som kunden selv ønsker, noe vi har fått respons på at vi har klart godt.

5.2.3.3 Valg av teknologi

Valget vårt av teknologi mener vi har vært veldig bra, vi har ikke opplevd gjennom utviklingen, at vi har blitt hindret grunnet valget av teknologi. I motsetning har vi oppdaget flere goder av valget gjennom utviklingsprosessen.

Sanity førte til en enkel opprettelse av databasesystemet, som var både intuitivt og skalerbart. Kombinasjon med GitLab, sammen med å hoste domenet på Vercel, førte til en utrolig enkel utrulling av systemet.

5.3 Administrative resultater

5.3.1 Fremdriftsplan

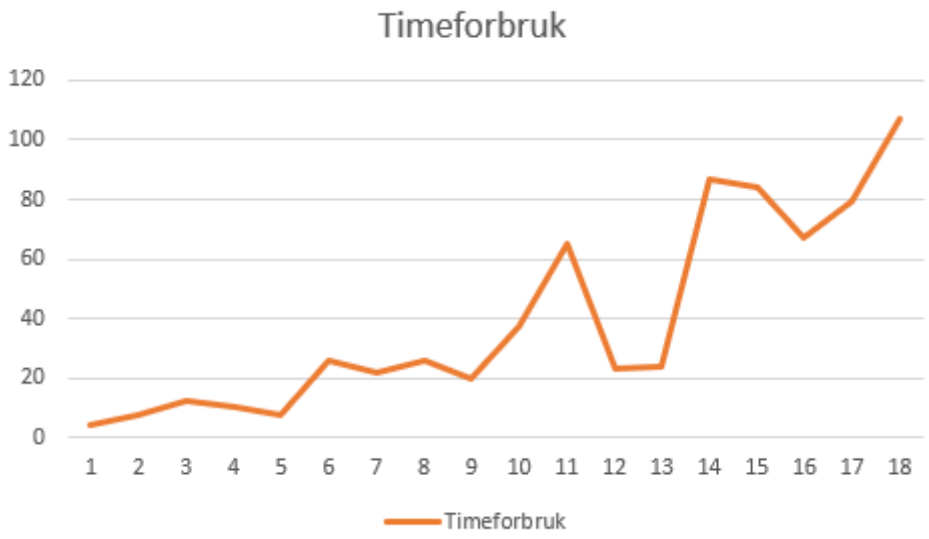
Ved oppstart av oppgaven utviklet vi en plan for hvordan vi skulle fordele arbeidet frem til innlevering, denne overordnede planen inkluderte milepæler med datoer for ferdiggjørelse. Vår originale fremdriftsplan var ok til oppstarten av prosjektet, men manglet gode aktivitetsmål på midtpunktet og videre mot innlevering. Av den grunn utviklet vi en ny

fremdriftsplan fra 20. Mars og fremover, denne planen gav et bedre og mer definert syn på hvilke aktiviteter som skal være klare til hvilken tid.

Milepælene i den første planen frem til overtakelsen av ny plan ble fullført til planlagte tidspunkter, men aktivitetene som kom under disse tidlige milepælene burde vi ha vært strengere med, dette kommer det mer av under kapittel 5.3.2. Milepælene i den nye fremdriftsplanen som omhandler det tekniske ble i hovedsak ferdige til riktige tidspunkt, med noen småting som måtte ordnes til overs.

5.3.2 Timeforbruk

Gjennom flere oppgaver i studien vår, har vi opplevd at timeforbruket ofte starter konservativt, så følger det en opptrapping videre gjennom utførelsen av oppgaven. Uheldigvis for oss, så er dette også noe vi ser på denne oppgaven. Dette har ført til at vi muligens fikk et mer kaotisk resultat på midtpunktet av oppgaven, og det er godt mulig vi ikke ville ha forårsaket samme forvirringer som vi har opplevd gjennom oppgaven.



Figur 5.4: Timeforbruk

Uansett føler vi at forbruket av timer på de ulike aktivitetene er innenfor våre forutsetninger.

5.3.3 Utviklingsmetode

Utviklingen av prosjektet skjedde ikke på den planlagte måten, i starten satt vi opp tre Trello-tavler for å holde oversikt over sprintene/arbeidet vårt. Planen var at den første tavlen inkluderte: en sprint backlog, hva som ble jobbet med, og hva som er ferdig. Etter fullført 7-dagers sprint, ble alt som lå i ferdig-raden automatisk flyttet til trello tavle nr 2, som er for laget for å vurdere arbeidet i sprinten. Ved vurdering skulle vi utføre samme prosess som i SCRUM, altså at vi diskuterer hva som gikk bra, hva som må endres, spørsmål til arbeid og hva som skal utføres på neste sprint. Dagen etter gjennomgangen av

sprinten ville listen som ble flyttet inn i tavle 2, flyttet til tavle 3, som er for arkivering av sprintene. Som sett i vedlegg D Prosjekthåndbok kapittel 2 ble denne prosessen bare gjennomført som planlagt i uke 7, 8, og 16. Dersom vi hadde vært mer disiplinerte, og faktisk klarte å gjennomføre denne prosessen jevnlig, ville vi ha hatt en mer detaljert oversikt over hvor vi ligger i prosjektet. I motsetning tok vi på oss en litt mer kaotisk variant, der vi diskuterte hva vi kom til å jobbe med videre uke for uke sammen. Dette endte opp med å fungere godt for oss og vi er begge fornøyde med samarbeidet oss imellom, som vi kommer til under kapittel 5.3.5.

5.3.5 Gruppearbeid

Samarbeidet i gruppen har gått særdeles bra gjennom hele prosjektet, mye takket være den gode dynamikken vi har med hverandre. Det at vi bare var to studenter på gruppen, samt at vi tidligere har jobbet sammen på ulike gruppeprosjekter gjennom studieløpet, har også forsterket vårt samhold. Vi hadde allerede en god stund før vi skulle velge oppgave, vært enige om at vi ønsket å jobbe sammen, i tillegg til at vi var enige om at det var denne oppgaven som var den best egnede oppgaven for oss. Dette førte til at vi startet diskusjon rundt teknologivalget i god tid før oppgavearbeidet skulle påbegynnes.

6 Konklusjon og videre arbeid

6.1 Konklusjon

Svaret på problemstillingen vi satte i kapittel 1.2, er todelt. Overordnet handler den første delen om det som utviklere har mulighet til å påvirke, dette kalles: on-page. Dette vil være å implementere og utnytte SEO på en god måte, og bruke relevante nøkkelord i innholdet på siden sin. Nettsiden må også være brukervennlig, ha god navigasjon og oppsettet til siden må være logisk og hierarkisk, slik at det er lett for robotene til søkemotorene å finne fram til innholdet på siden. I tillegg til dette er det en del som ikke er like lett, det handler om å holde oversikt over hvordan andre sider oppfører seg rundt sitt eget domene, dette kalles off-page. En teknikk for å sikre god off-page SEO, er å betale søkemotorene for å markedsføre domenet, etter dette blir det litt verre. Resten av off-page SEO går i hovedsak ut på å ha oversikt over hvilke nettsteder som lenker til domenet, dersom nettsiden som lenker er trofast med søkemotoren, ergo at nettsiden har vært solid plassert over lengre tid, vil dette føre til et positivt resultat. Men dersom nettstedet som lenker til domenet ikke er trofast, enten om det er helt nytt, eller at det har mistenksom aktivitet, kan dette føre til negativ innvirkning på SEO-rangeringen til domenet vårt.

6.2 Videre arbeid

6.2.1 WCAG 2.1

Gjennom vår implementasjon av WCAG 2.1 sine retningslinjer, fikk vi ikke implementert alle punktene tilfredsstillende nok til å kunne vurderes som dekket. De følgende punktene vil vi anbefale at tas opp ved videre utvikling av systemet.

3.3.5 Help

Dette punktet går ut på at brukeren skal ha mulighet til å få hjelp, for å motvirke brukerfeil. En teknikk som kan brukes for å dekke dette kravet vil være å tilby en lenke til en hjelpeside på hver nettside der brukeren er nødt til å skrive inn informasjon. I tillegg er det også anbefalt av WCAG (Help (Level AAA), n.d-a) å legge til et tittel-element for å kunne tilføye hjelp avhengig av kontekst.

3.3.6 Error prevention

WCAG sitt punkt nr 3.3.6 Error prevention handler om at før data blir sendt, skal bruker kunne få en oversikt over dataen som blir levert, før den sendes. Vi anbefaler at ved videre arbeid dette implementeres på kontaktskjemaet, på en slik måte at når bruker klikker på send på skjemaet, så blir dataen som er innskrevet visualisert lik e-posten som blir levert gjennom EmailJS. Ved denne visualiseringen blir brukeren spurt om å akseptere at dataen fra skjemaet blir levert slik den er presentert.

6.2.2 OWASP

A04 Insecure design

OWASP sitt punkt A04, fikk vi etter vår vurdering ikke satt oss godt nok inn i, og vi vurderer derfor dette som ett av punktene vi ikke tilfredsstiller. Vi vil anbefale å sette seg inn i dette ved videre utvikling.

A08 Software and Data Integrity Failures

A08 vurderer vi som ikke tilfredsstilt, siden vi har forsøkt å være sikkerhetsmessig kritiske til alle pakkene som vi utnytter, men vi kan også se at vi selv ikke har nok kompetanse til å få utført en tilfredsstillende sikkerhetsanalyse. I tillegg vil vi anbefale å styrke CI:CD prosessen ved å sette opp aksesskontroll.

A09 Security Logging and Monitoring Failures

Systemet er ikke utviklet for å generere logger eller holde oversikt over trafikk, dette fører til at sikkerhetshull kan utnyttes uten at noe data blir ivaretatt angående besøket. Metadata rundt sending av kontaktskjema og hvilke sider som blir navigert kan være smart å ta vare på.

A10 SSRF

Ved utvikling av et system som tar i bruk SSR, burde vi ha tenkt gjennom mulige sikkerhetshull. Ett av disse sikkerhetshullene er Server Side Request Forgery. SSRF kan oppstå dersom et system som bruker SSR henter data uten å sjekke domenet dataen blir hentet fra. Som nevnt er dette et relativt stort sikkerhetshull som burde dekkes, vi har ikke fått satt oss godt inn i det tekniske på hvordan dette bør dekkes, så det anbefales å følge OWASP (OWASP, n.d-b) sine anbefalinger.

7 Referanser

Google. (2022, August 8). *ReCAPTCHA v3 | google developers*. Google. Hentet April 3, 2023, fra https://developers.google.com/recaptcha/docs/v3

StatCounter. (n.d-a). *Desktop vs Mobile Market Share Worldwide.* Hentet 10. April, 2023, fra https://gs.statcounter.com/platform-market-share/desktop-mobile/worldwide#yearly-2011-2022

StatCounter. (n.d-b). *Desktop vs Mobile Market Share Norway*. Hentet 10. April, 2023, fra https://gs.statcounter.com/platform-market-share/desktop-mobile/norway#yearly-2011-2022

Google. (2023, Februar 21). *Get your content on Google*. Hentet 19. Mai, 2023, fra https://developers.google.com/search/docs/fundamentals/get-on-google

Uutilsynet. (n.d). *Oppbygging av WCAG 2.1*. Hentet 20. Mai, 2023, fra https://www.uutilsynet.no/wcag-standarden/oppbygging-av-wcag-21/139

Cognizant. (n.d). *Agil utvikling*. Hentet 20. Mai, 2023, fra https://www.cognizant.com/no/nb/glossary/agile-development

Next.js. (n.d-a). *What is Rendering? - How Next.js works.* Hentet 21. mai, 2023, fra https://nextjs.org/learn/foundations/how-nextjs-works/rendering

Next.js (n.d-b). *Build Time and Runtime - How Next.js works.* Hentet 21.05.2023, fra https://nextjs.org/learn/foundations/how-nextjs-works/buildtime-and-runtime

Next.js (n.d.c). *Incremental Static Regeneration.* Hentet 21.05.2023, fra https://nextjs.org/docs/pages/building-your-application/data-fetching/incremental-static-regeneration

Computas. (n.d). *Agile og lean*, Hentet 21.05.2023, fra https://computas.com/artikkel/agile-og-lean/?gclid=CjwKCAjwgqejBhBAEiwAuWHioMN_FrioHrJe6D7dUL3N_7dX2ZyAeq34A5Vop0PK4ptEZbwK1z9aNhoCLAsQAvD_BwE

WCAG. (n.d-a). *Help (AAA),* hentet 21.05.2023, fra https://www.w3.org/WAI/WCAG21/Understanding/help.html

WCAG. (n.d-b). *Error Prevention (All) (Level AAA),* hentet 21.05.2023, fra https://www.w3.org/WAI/WCAG21/Understanding/error-prevention-all.html

OWASP. (n.d). A10:2021 – Server-Side Request Forgery (SSRF), hentet 21.05.2023, fra https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/

8 Vedlegg

Vedlegg A - Visjonsdokument

Vedlegg B - Kravspesifikasjon

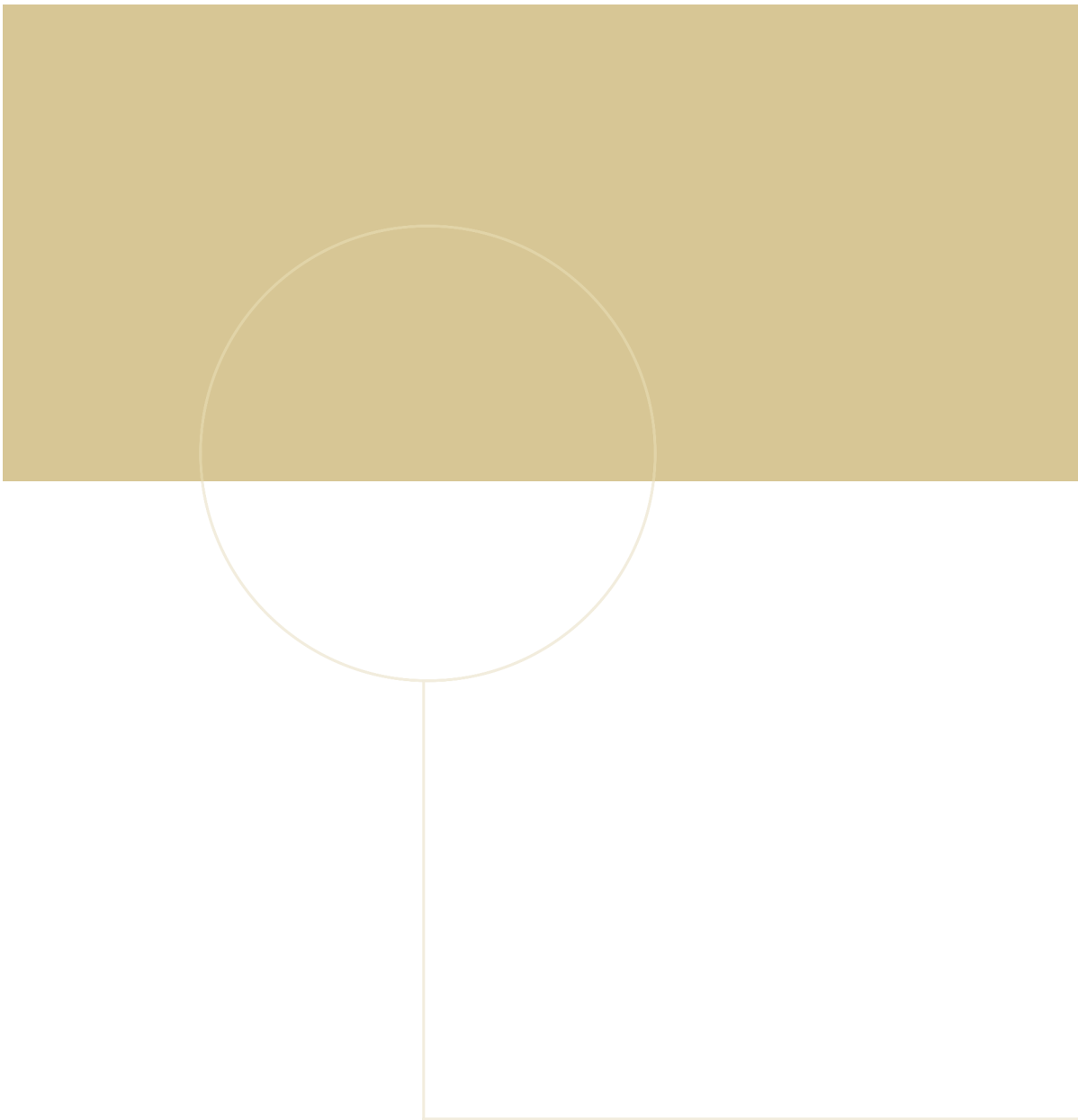
Vedlegg C - Systemdokumentasjon

Vedlegg C1 - Tjenestedokumentasjon Vercel

Vedlegg C2 - Tjenestedokumentasjon Sanity

Vedlegg C3 - Tjenestedokumentasjon EmailJS

Vedlegg D - Prosjekthåndbok



NTNU

Kunnskap for en bedre verden