Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for IKT og realfag

Sander Ringmo Fylkesnes
Mahmod Ahmed Mohamed
Stian Larsen Rørvik

Utvikling av den nye nettsiden til bandet "Frostbite"

Inkluderer blant annet hovedrapport, forprosjektplan, kravdokument og systemdokumentasjon

Bacheloroppgave i BIDATA
Veileder: Kjell Tomren
Mai 2022

NTNU
Kunnskap for en bedre verden

Sander Ringmo Fylkesnes
Mahmod Ahmed Mohamed
Stian Larsen Rørvik

Utvikling av den nye nettsiden til bandet "Frostbite"

Inkluderer blant annet hovedrapport, forprosjektplan, kravdokument og systemdokumentasjon

Bacheloroppgave i BIDATA
Veileder: Kjell Tomren
Mai 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for IKT og realfag

NTNU

Kunnskap for en bedre verden

Sammendrag

Dette bachelorprosjektet dekker utviklingen av den nye nettsiden til bandet kjent som «Frostbite». Gruppen har oppnådd følgende resultater ved å kombinere informasjonen vi har lært fra våre tidligere emner, samt resultatet av forskning på ny teknologi og optimale løsninger for å lage en nettside som passer godt både for Frostbite og NTNUs systemkrav.

I løpet av våre tidligere emner har vi lært nyttig informasjon som blant annet inkluderer hvordan vi kan utvikle programmer, applikasjoner, nettsider og databaser, samt implementere slik kode med høyt fokus på lav kobling, høy kohesjon, gjenbrukbarhet, stabilitet, dokumentasjon og design. Vi har også utforsket et bredt spekter av nettsideteknologier og lært hvordan man kobler sammen forskjellige tjenester gjennom avanserte og sikre dataoverføringsprotokoller. Databasesystem var også en viktig del av utdanningsprogrammet vårt, der vi lærte oss hvordan vi lagrer forskjellige typer data på en sikker måte gjennom en lang rekke med forskjellige databasebehandlingssystemer.

Ved å kombinere informasjonen vi lærte under programmering 1&2, samt systemutvikling, databaseapplikasjoner, nettverk og netteknologi, har vi utviklet en minimalistisk og elegant nettside som enkelt kan bli redigert direkte fra selve nettsiden. Dette betyr at medlemmene av Frostbite ikke trenger å kontakte noen eksterne utviklere dersom de ønsker å gjøre endringer i layout, farger eller innhold på nettsiden, de kan enkelt redigere dette gjennom selve nettsiden. Annen teknologi vi har forsket på og implementert inkluderer rike funksjoner som blant annet vårt flerlagsdesign, dette designet lar oss bruke en effekt kjent som «parallakseeffekten». En slik effekt gir nettstedet et spektakulært utseende av dybde og kompleksitet, samtidig som det forblir minimalistisk med tanke på design, ønske og oppsett.

Innhold som farger, bilder, sanger, billetter, singler og album kan enkelt redigeres gjennom vår egen skreddersydde sikre administrasjonsmeny. Denne menyen er kun tilgjengelig for autoriserte administratorer, i vårt tilfelle er dette Frostbite-medlemmene selv. Nettsiden er koblet til en ekstern back-end server kjent som Firebase, en tjeneste levert av Google LLC. Hver gang en autorisert administrator gjør endringer på siden, blir endringene lastet opp og lagret på serveren. Når en besøkende besøker nettsiden, hentes verdiene fra den eksterne databasen og brukes umiddelbart til å gi det siste designet, oppsettet og innholdet nettsiden har å tilby.

Abstract

This bachelor project covers the development of the new webpage to the band known as "Frostbite". The group has achieved the following results by combining the information we have learned from our earlier courses, as well as researching new technology and potential solutions to make a webpage that fits both Frostbites and NTNUs requirements.

During our earlier courses we have learned to develop programs, applications, websites, and databases, as well as implementing code with high focus on low coupling, high cohesion, reusability, stability, documentation, and design. We have also explored a wide range of website technologies and learned how to connect different services through advanced and secure data transfer protocols. Databases was also an important part of our education program, it taught us how to store different types of data remotely and securely in a wide variety of database management systems.

By combining the information we learned during programming 1&2, as well as system development, database applications, networking, and web technology, we have developed a minimalistic and sleek webpage that can easily be edited directly from the page itself. This means that the members of Frostbite will not have to contact any external developer if they wish to make changes to the layout, colours, or contents of the webpage, they can easily edit this through the page themselves. Other technology we have researched and implemented includes rich features such our multi-layered design which allows us to use an effect known as the "parallax effect" to give the website a spectacular look of depth and complexity while still remaining minimalistic in terms of design and layout.

Contents such as colours, images, songs, tickets, singles, and albums can easily be edited through our own custom-made secure administration menu. This menu is only accessible by authorized administrators, in our case this is the Frostbite members themselves. The webpage is connected to a remote back-end server known as Firebase, a service provided by Google LLC. Whenever an authorized administrator make changes to the page, the changes are uploaded and stored on the server. When a visitor visits the webpage, the values are fetched securely from the remote database and are instantly applied to give the latest design, layout, and contents the website has to offer.

Forord

Gruppen til denne bacheloroppgaven består hovedsakelig av dataingeniør-studentene Stian Rørvik, Sander Fylkesnes og Mahmod Mohamed fra NTNU i Ålesund, tidsrom for utførelse av bachelor oppgaven er [10.01.22] til [20.05.22].

Når det gjelder valg av oppgave og problemstilling, valgte gruppen oppgaven "utvikling av en ny nettside for bandet Frostbite» som førstevalg fordi det hørtes ut som et spennende og lærerikt prosjekt. Merk at denne bacheloroppgaven var en av mange alternative og diverse oppgaver som ble listet opp av instituttet, men basert på hva vi har lært ble gruppen enig om at det var denne oppgaven som passet best til våre bakgrunnskunnskaper, læringsmetoder og kompetansemål.

Hovedgrunnen til at gruppen har valgt akkurat denne oppgaven er fordi den kombinerer en god mengde med tidligere emner fra studieprogrammet, disse emnene inkluderer blant annet INGA1001 (innføringsemne), IDATA1001 (programmering 1), IDATA1002 (systemutvikling), IDATA2001 (programmering 2), IDATA2303 (databaseapplikasjoner), IDATA2304 (nettverksprogrammering), IDATA2301 (webteknologi) og IDATA3306 (applikasjonsutvikling). Oppgaven lar oss derfor uttrykke en god og vid forståelse innenfor disse emnene, noe som tilsvarer god kunnskap innenfor studieretningen med tanke på hvordan de forskjellige emnene henger sammen. Denne oppgaven var derfor veldig relevant for en bachelor, noe som igjen førte til at den ble et soleklart førstevalg.

En stor takk går ut til vår veileder Kjell Inge Tomren, samtidig vil vi også takke vår kontaktperson Reinhardt Oma fra Frostbite for å opprettholde god kontakt og et godt samarbeid gjennom hele utviklingsfasen til prosjektet. Som dataingeniør-studenter setter vi spesielt stor pris på de detaljerte tilbakemeldingene som formet nettsiden til det den er i dag.

Takk for kollaborasjonen!

Oppgavetekst

Oppgaven til gruppen var å lage en ny nettside til bandet Frostbite. Bandet ønsket en minimalistisk nettside der de kunne promotere bandet sitt ved å legge ut lenker til konsertsider, promotere deres sosiale medier plattformer, og vise fram sin nyeste musikk.

Kravene for oppgaven var å lage en nettside som inneholdt:

- Info (om bandet)
- YouTube direkteavspilling
- Linker til sosiale medier
- Integrert Spotify direkteavspilling
- Bilder fra tidligere konserter (i form av et galleri/bildekarusell)
- Konsert info med linkfunksjon til eksterne billettnettsider

Under NTNU kravene var det også en «ønsket funksjon» som ikke var obligatorisk, dette var en «merchandise shop» med «on demand printing» på selve nettsiden. Dette ble tatt opp med Frostbite (arbeidsgiver) tidlig i planleggingsfasen og de konkluderte med at en slik funksjon ikke var ønsket, denne funksjonen ble derfor henlagt.

Se også på kravdokumentet i vedleggene for mer informasjon om kravene til systemet vårt.

Innhold

Sammendrag 1
Abstract 2
Forord 3
Oppgavetekst 4
Innhold 5
Figur og Tabell liste 6
Kapittel 1: introduksjon og relevans 7
Kapittel 1.1: Problemstillinger 7
Kapittel 1.2: Struktur 8
Kapittel 1.3: Akronym og forkortinger 9
Kapittel 2: Teori og materialer 11
2.1 Utforming og oppbygning 11
2.2 Administrasjon og redigering 12
2.3 Autentisering og kryptering 13
2.4 Database modell og variabler 14
2.5 Analysering av data 15
Kapittel 3: Metode 16
Kapittel 3.1 Forskningsmetode 16
Kapittel 3.2 Valg av teknologi og utviklingsmetode 16
Kapittel 3.2.1 Utviklingsprosess og metode 16
Kapittel 3.2.2 Struktur og sammenheng 16
Kapittel 3.2.3 Front-end 17
Kapittel 3.2.4 Back-end 17
Kapittel 3.2.5 Arbeids- og rollefordeling 17
Kapittel 4: Resultater 19
Kapittel 4.1 Vitenskapelige Resultater 19
Kapittel 4.2 Ingeniørfaglige Resultater 20
Kapittel 4.2.1 Status for problemstilling 1 20
Kapittel 4.2.2 Status for problemstilling 2 20
Kapittel 4.2.3 Status for problemstilling 3 21
Kapittel 4.2.4 Status for problemstilling 4 21
Kapittel 4.3 Administrative Resultater 21
Kapittel 5: Diskusjon 22
Kapittel 5.1 Diskusjon Vitenskapelige Resultater 22
Kapittel 5.1.1 Diskusjon plassering av innhold 22
Kapittel 5.1.2 Diskusjon Fargeblindhet 22
Kapittel 5.2 Diskusjon Ingeniørfaglige Resultater 22

Kapittel 5.2.1 Diskusjon av resultat problemstilling 1 22
Kapittel 5.2.2 Diskusjon av resultat problemstilling 2 23
Kapittel 5.2.3 Diskusjon av resultat problemstilling 3 24
Kapittel 5.2.4 Diskusjon av resultat problemstilling 4 24
Kapittel 5.3 Diskusjon Administrative Resultater 25
Kapittel 6: Konklusjon og videre arbeid 26
Videre Arbeid 27
Samfunnspåvirking 28
Økonomi 28
Miljøpåvirkning 28
Diskutering av profesjonsetiske problemstillinger 29
Drøfting: Sander Ringmo Fylkesnes 29
Drøfting: Mahmod Ahmed Mohammed 29
Drøfting: Stian Larsen Rørvik 29
Referanser 30
Vedlegg 32

Figur og Tabell liste

Figur 1: Sett av en person med vanlig syn 11
Figur 2: Sett av en person med protanomali 11
Figur 3: Sett av en person med akromatopsi 11
Figur 4: Nettsiden slik den ser ut I dag 12
Figur 5: Eksempel på forandringer oppnådd gjennom administrator menyen 12
Figur 6: Innloggingsvinduet som vist på nettsiden 13
Figur 7: Strukturen og innholdet til NoSQL databasen 14
Figur 8: Ekstern skylagring av bilder, logoer og ikon 14
Figur 9: vinduet nederst på siden som blir vist til folk som besøker nettsiden for første gang. 15
Figur 10: «Privacy Policy» vinduet som lar besøkende lese om datainnsamlingen og eventuelt oppdatere valget sitt 15
Figur 11: Rollefordeling i Jira 17
Figur 12: Basert på flere studier vil besøkende først se innholdet i den hvite boksen, deretter innholdet i den røde. 19

Kapittel 1: introduksjon og relevans

Kapittel 1.1: Problemstillinger

Vi startet først med å avdekke behovet for en nettside. Frostbite ville ha en hovedportal som de kunne legge ut media og lenker til billett bestillinger. Nettsiden gjør det også lettere for de å vise frem porteføljen deres til folk som vil se mer om bandet. Vi vil si at en nettside er en god måte å tilfredsstille det behovet på.

Der er flere problemstillinger som vi måtte utforske, mange som dukket opp under utviklingen av web løsningen:

1. Den største problemstillingen var hvordan vi skulle implementere muligheten for tilpassing av websiden uten å forandre koden. Dette er for at bandmedlemmene vil ha administrator rolle, men har lite erfaring med HTML/CSS/JS.

2. En annen problemstilling er hva hosting løsning som passer best med dette prosjektet. Det er nødvendig siden den skal være offentlig tilgjengelig.

3. Hva teknologier å anvende er en problemstilling som er relevant. Dette vil si blant annet om det er nødvendig å bruke rammeverk. Skal vi lage egen back-end, eller koble opp mot en eksisterende service?

4. En problemstilling ganske sentral i webdesign er hvordan UI/UX skal både passe estetisk til hvordan kunden vil ha det, men også passe på at grensesnittet er intuitivt for individ som besøker den.

5. Når det kommer til selve utviklingen av oppgaven, er fordeling av ansvar ganske viktig for å ha god fremgang med prosjektet og jevnt arbeid.

Kapittel 1.2: Struktur

Kapittel 1: Dette er introduksjonen der vi gjør rede for det faktiske behovet til kunden og om nettsiden er en relevant løsning. Her beskriver en og forskjellige relevante problemstillinger som vi må løse. Det er her rapport struktur samt forklaring av akronym og andre forklaringer blir beskrevet.

Kapittel 2: Her står det om relevant teori og annet materialet som er nødvendig for forståelse av prosjektet. Det vil si korte utredninger av anvendte teknologier og konsepter.

Kapittel 3: Her er metodene (i ettertid valg av metode) presentert. Dette kapittelet er delt opp i to under-kapittel. Det første under kapittelet tar for seg forskningsmetoden, mens det andre under kapittelet tar for seg utviklingsmetode som igjen er delt opp i mindre små kapittel (utviklingsprosess og metode, struktur og sammenheng, Front-end, Back-end, Arbeid og rollefordeling)

Kapittel 4: Her er resultatene presentert. De er delt opp i tre underkapittel: Vitenskapelige resultater, Ingeniørfaglige resultater og Administrative Resultater. De ingeniørfaglige resultatene er videre delt opp etter hver problemstilling.

Kapittel 5: Her er resultatene som var presentert diskutert. Dette kapittelet har en tilsvarende kapitteloppdeling til kapittel 4. Diskusjonen i dette kapittelet tar for seg fordeler og ulemper av løsningene, samt hvor fornøyd oppgavegiveren er med den endelige løsningen.

Kapittel 6: Her er konklusjon og videre arbeid, dette er det siste hovedkapittel og fungerer som en avslutning på oppgaven.

Samfunnspåvirking: Dette kapittelet blir om positive og potensielle negative effekter på samfunn og miljø.

Kapittel 1.3: Akronym og forkortinger

HTML: HTML står for «HyperText Markup Language». HTML er ansvarlig for selve strukturen av en webside og består av flere typer «markups» som merker av hva type element innhold på en nettside er. HTML er derfor grunnsten til nettsider. Du kan lese mer om HTML her: (Mozilla, 2022)

CSS: CSS står for «Cascading Style Sheet». CSS er ansvarlig for utforming og utseende av en nettside. Med CSS kan du bestemme hvordan nettleseren skal presentere HTML elementene. Det vil si tekst font, farger som nettsiden skal bruke, om bilde skal ha avrundet eller firkantet hjørne ... etc. Utforming delen har med hva «display type» en vil HTML element skal ha. En ofte brukt displaytype er «grid» der du kan bestemme plasseringen av underordnede element i et overordnet element som en til vanlig referer som en konteiner (samt underordnede element i forhold til hverandre). De fleste moderne sider bruker i dag CSS. Du kan lære mer om CSS her: (Mozilla , 2022).

JS: JS står for «JavaScript». JS er et programmeringsspråk hovedsakelig brukt for webutvikling. En kan programmere i JS for å legge til funksjon på en nettside. Et eksempel er når du trykker på en knapp på en nettside så kan en ha en funksjon som venter på klikk gjennom en «onClick» funksjon og kjører den funksjonen som åpner et pop-up vindu. Du kan lære mer om JS her: (Mozilla, 2022)

SQL: SQL står for «structured Query Language». er språket en bruker for å kommunisere med en relasjonsdatabase. Det lar deg hente/legge til/sammenligne data fra det som heter «tables». Du kan lære mer om SQL her: (Loshin, 2022)

API: API star for "Application Programming Interface" ("Applikasjon programmering grensesnitt" på norsk). Dette er en generell betegnelse for et grensesnitt som en programmerer brukar for å kommunisere med et program. Dette gjøres gjennom abstrakte metodekall der utvikleren ikke trenger å vite hvordan det funker internt, bare hva du får tilbake/sender inn. Dette gjør det mye lettere og kjappere å lage nye program med gjenbruk av annen kode. Du kan lære mer om API her: (RedHat, 2017)

DB: DB står for «Database» og er en måte å lagre informasjon på en strukturert måte. Der finnes flere typer databaser, men den mest vanlige er relasjonsdatabaser hvor data blir lagret i tabeller, der attributter blir kolonnene, mens radene er en «record» altså enheter som har sine verdier av disse attributtene. En bruker SQL for å hente og sammenligne data. Du kan lære mer om relasjonsdatabaser her: (Oracle, 2022)

SCRUM: SCRUM er et rammeverk for en arbeidsprosess brukt i systemutvikling. Problemstillinger blir delt opp i mindre oppgaver som havner i en backlog (oppgavekø) som da blir brukt i en sprint. Resultat blir inspisert etter hver sprint. Du kan lære mer om SCRUM her: (SCRUM, 2022)

Nginx: Nginx er en http-server (nettstedserver), men kan gjøre andre ting som å hoste E-post server og. Du kan lære mer om Nginx her: (Nginx, 2022)

GUI: GUI står for «Graphical User Interface». På norsk blir dette kalt for et grafisk bruker grensesnitt. Med GUI, so kan en bruker interagere med programvare med å bl.a. klikke på ikon, skrive i tekst felt, få pop-ups osv... GUI er mye mer brukervennlig for de fleste brukere og øker bredden på folk som kan bruke programvaren på grunn av økt intuitiv. Du kan lære mer om GUI her: (Tutorialsport, 2022)

Kapittel 2: Teori og materialer

2.1 Utforming og oppbygning

Hovedsakelig er nettsiden bygd opp av fundamental HTML (Hypertext Markup Language), CSS (Cascading Style Sheet), og JS (JavaScript). Oppbygningen består av et simpelt og lettvint design som lar alle besøkende enkelt navigere rundt på nettsiden uten problemer. Med tanke på universell utforming har nettsiden blitt designet til å være akkurat like brukbar og oversiktlig for alle typer mennesker som besøker nettsiden uavhengig av syn, kunnskap og konsentrasjonsnivå. Samtidig har nettsiden også blitt designet til å tolerere feil i inndatafelt, dette gjelder da spesielt administrasjons menyen. Skriver man inn en feil verdi innenfor nettsiden sin logikk blir det returnert en relevant feilmelding til sluttbrukeren. Dette kan for eksempel hende når inndatafeltet står tomt når en trykker på «oppdater», eller at inndataen ikke er i riktig format basert på hvilken administrasjons-knapp som blir trykket på.

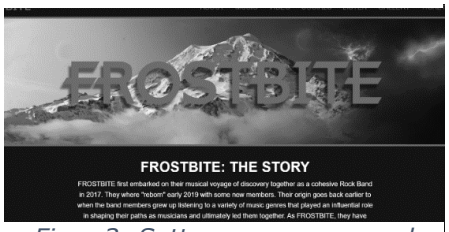
Selve designet og rekkefølgen til innhold på nettsiden ble som sagt bestemt av Frostbite selv, der de ønsket en enkel og lettvint side der man hadde «alt på ett sted». I stedet for å lage flere sub-sider ble innholdet derfor listet opp under forskjellige kategorier på en og samme side. Dette innholdet kan bli nådd ved å bla nedover siden, en annen måte å navigere rundt på nettsiden består av å bruke snarveiene som etter hvert utformet knappene i navbaren. Hver snarvei blar brukeren automatisk ned til den relevante kategorien på nettsiden.

Figur 1: Sett av en person med vanlig syn

Her kan man se noen eksempler på hvordan den sterke kontrasten i fargeforskjellen til nettsiden utgjør et synlig og klart layout selv til de som har varierende fargeblindhet. I figur 1 er nettsiden representert gjennom øynene til en person med vanlig syn, altså ingen øyesykdom, synsforstyrrelser eller andre synsplager.

Figur 2: Sett av en person med protanomali

Figur 2 representerer nettsiden sett gjennom øynene til en person med fargeblindheten protanomali. Som sett på bilde er nettsiden tydelig og lesbar selv om fargen rød ikke er helt til stedet. Dette er fordi nettsiden hovedsakelig bruker et «svart/hvit» tema, noe som igjen resulterer i at det lesbare innholdet på nettsiden ikke blir veldig påvirket av fargeblindhet.



Figur 3: Sett av en person med akromatopsi

I figur 3 ser vi nettsiden gjennom øynene til en person med fargeblindheten akromatopsi. Igjen ser man at teksten kommer klart og tydelig frem med hensyn på denne øyesykdommen. Den svarte bakgrunnen funker også som en lyddemper, slik at de som er spesielt sensitiv til skarpt lys ikke blir blendet om de sitter i et mørkt rom.

2.2 Administrasjon og redigering

En autorisert administrator har muligheten til å redigere nettsiden med stor variasjon i henhold til design og innhold. Gjenstandene som kan bli redigert inneholder blant annet det grunnleggende oppsette på nettsiden, dette inkluderer fargen på tekster og knapper, bakgrunnsfargen, og til sist disposisjonsfargen. Fargeverdiene blir manipulert via HEX fargekoder, fargekodene blir satt direkte i et inndatavindu som blir vist ved å trykke på de relevante knappene i administrator menyen. Funksjonene som er koblet til de relevante fargeknappene gjør til slutt et back-end-anrop som da lagrer HEX fargekoden i databasen til serveren og henter den når nettsiden er lastet. Dette lar administratorer enkelt og greit redigere designet og innholdet på nettsiden uten å ha noen slags forkunnskaper om koding, administratorer trenger altså ikke å åpne en eneste HTML, CSS eller JS fil for å redigere og vedlikeholde nettsiden.

Det er også mulig å redigere forsidebildet til nettsiden etter behov, igjen kan dette kun gjøres av en autorisert administrator gjennom administrator menyen. Nettsiden takler også GIF bildeformatet, noe som fører til at nettsiden enkelt kan bli oppdatert gjennom administrator menyen til å ha et animert forsidebilde i stedet for et statisk forsidebilde, om Frostbite ønsker noe slikt, så klart. Det er også mulig å skru «FROSTBITE» teksten som ligger over forsidebildet av og på, dette er en nyttig funksjon å ha i tilfelle Frostbite ønsker å laste opp et forsidebilde der teksten blokkerer hovedfokuset til bildet.

Andre administrator funksjonaliteter inkluderer muligheten til å legge til en ny single eller et nytt album, igjen gjør man dette gjennom administrator menyen. Først trykker en autorisert administrator på «add single/album» knappen i menyen, deretter laster man opp et ikon til single/albumet, og til slutt legger administratoren inn en link til Spotify eller YouTube basert på behov. Den nye informasjonen er sendt og lagret i back-end serveren. Disse single/albumene blir igjen hentet hver gang man besøker nettsiden eller redigerer nettsidens innhold gjennom administrator menyen. Man sletter en single/et album ved å lime inn linken til det som man ønsker å slette i det relevante inndatafeltet som blir vist når man trykker på «delete single/album» knappen i administrator menyen.

Hovedinnholdet på nettsiden er den informative delen som henviser de besøkende til relevant informasjon om bandet. Dette inkluderer innhold som blant annet live avspilling av YouTube videoer, hyperlenker til andre sosiale medier kontoer, avspillings-plattformer, live avspilling av Spotify låter, animert bildekarussel med relevante bilder, og billettknapper som sender de besøkende til bestillingssidene for konsertbilletter. Alt dette kan som sagt bli redigert og oppdatert etter behov av en autorisert administrator gjennom administrator menyen. Nedenfor ser man et eksempel på forandringer som enkelt kan oppnås ved bruk admin menyen.

Figur 4: Nettsiden slik den ser ut I dag

Figur 5: Eksempel på forandringer oppnådd gjennom administrator menyen

2.3 Autentisering og kryptering

Om en person ønsker å autorisere seg som administrator på nettsiden for å få tilgang til redigerings og bildemanipulerings-funksjonene i administrator menyen, må man først logge seg inn på en administrator bruker med korrekt brukernavn og passord. Dette gjøres gjennom et innloggingsvindu som blir åpnet når man trykker på «administrator login» knappen helt nederst på siden. Denne knappen er «gjømt vekk» i footeren fordi de eneste brukerne som finnes på nettsiden er administrator-brukere. Det er som sagt ingen grunn til å ha vanlige brukere på denne nettsiden. Informasjonen som blir sendt og redigert i den eksterne databasen er kun layout variabler og eksterne bilder som blir vist til alle som besøker siden. Derfor kan alle også se denne databasen, informasjonen som er lagret i denne databasen er altså ment til å bli sett, men merk at det kun er autoriserte administratorer som har tillatelse til å redigere den. Informasjon i forhold til autentisering (blant annet passord og annen sensitiv informasjon) er ikke lagret i denne databasen. Den sensitive informasjonen er kryptert og lagret i et eksternt lukket system utgitt av back-end leverandøren. Krypteringen av sensitiv informasjon består av flere interne modifiserte hashing algoritmer.

Autoriseringen er sikret gjennom en rekke med diverse sikkerhetsregler, disse innebærer blant annet å generere en unik bruker ID for administrator kontoer som manuelt må bli godkjent på server-siden for å få tilgang til å redigere den offentlige databasen (som står for innhold på nettsiden). Per i dag finnes det bare en felles administrator konto som er direkte koblet opp mot back-end serveren. Siden Frostbite bare trenger en bruker, er det derfor ingen grunn til å lage nye brukere gjennom nettsiden. For hver sesjon blir også en unik autentiserings-token generert og opprettholdt til en logger ut. Det er altså ingen måte å «jukse» systemet på med mindre man hacker seg direkte inn i den eksterne serveren, noe som ikke er mulig å gjøre gjennom nettsiden. Potensielle «brute force» angrep er også svært usannsynlig med tanke på sikkerheten til den populære back-end leverandøren vi har tatt i bruk for nettsiden. Autoriseringstrafikken på nettsiden er altså begrenset og beskyttet mot misbruk. All trafikk mellom back-end serveren og nettsiden blir kryptert gjennom HTTPS (Hypertext Transfer Protocol Secure).

Figur 6: Innloggingsvinduet som vist på nettsiden

Til venstre ser man innloggingsvinduet til nettsiden, man kan autorisere seg som en administrator med å oppgi den riktige email adressen og det riktige passordet som er koblet opp til administrator kontoen. Denne kontoen gjør det mulig å forandre på designet og innholdet til nettsiden uten å redigere selve kildekoden. Variablene blir sendt og lagret i en ekstern database og hentet når man laster inn nettsiden.

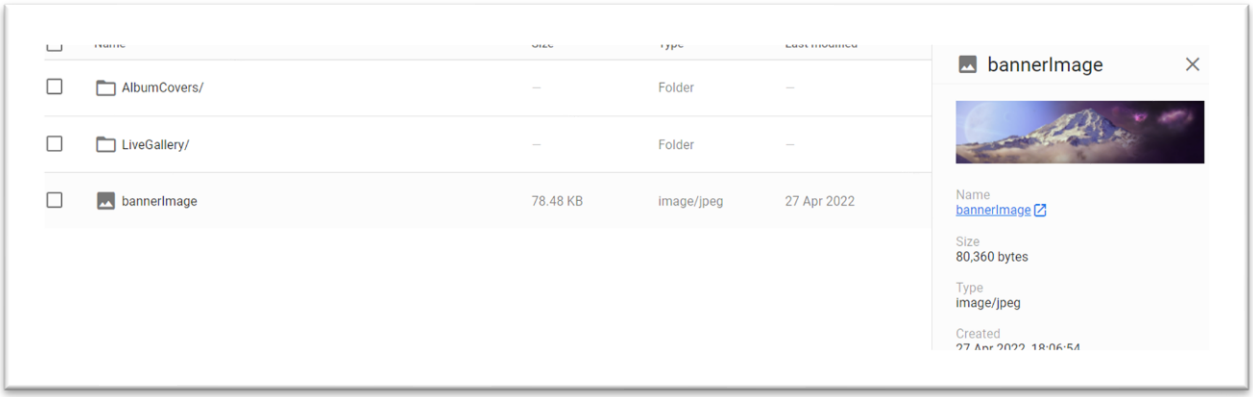
2.4 Database modell og variabler

Back-end serveren bruker en realtime database som er oppbygd av en NoSQL struktur, denne databasen inneholder hovedsakelig variabler som definerer utseende og innholdet på nettsiden. Innholdet er strukturert i databasen etter hvor det hører til på nettsiden, så «tickets» for eksempel har sin egen sub-folder. Andre verdier inkluderer blant annet fargekodene som nevnt tidligere, samtidig som YouTube og Spotify URL-er til diverse Frostbite album. Filene til nettsiden blir også lagret i denne skyen. Disse filene inkluderer blant annet ikon, forsidebilder og albumbilder.

Figur 7: Strukturen og innholdet til NoSQL databasen

Til høyre ser du et eksempel på hva databasen inneholder og hvordan den lagrer de relevante dataene. Disse variablene består som sagt av CSS konfigurasjoner samtidig som innhold på nettsiden. Hovedgrunnen til at disse variablene og dette innholdet blir lagret eksternt i en database er fordi det lar Frostbite medlemmene redigere og manipulere designet og utseende til nettsiden uten å gå inn i selve kildekoden, alle forandringene blir altså gjort gjennom administratormenyen på selve nettsiden.

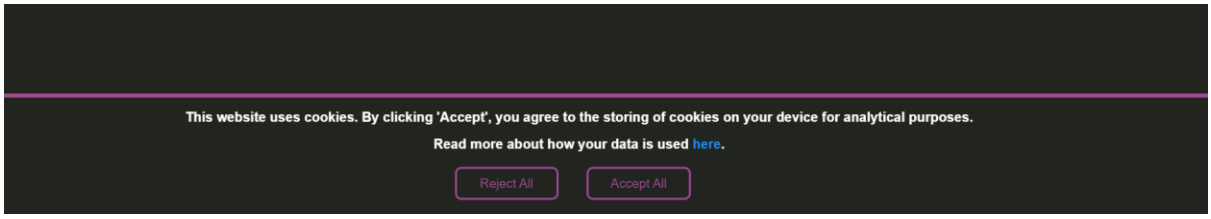
Dette var svært viktig å oppnå siden det hadde blitt tungvint for Frostbite å kontakte utviklere hver gang de ønsket å for eksempel oppdatere litt på fargen eller legge til et nytt bilde. Oppsettet vi har laget for Frostbite gjør det altså enkelt og lettvint å holde siden oppdatert selv, uten noen behov for kodekunnskaper eller ekstern hjelp fra utviklere.



Figur 8: Ekstern skylagring av bilder, logoer og ikon

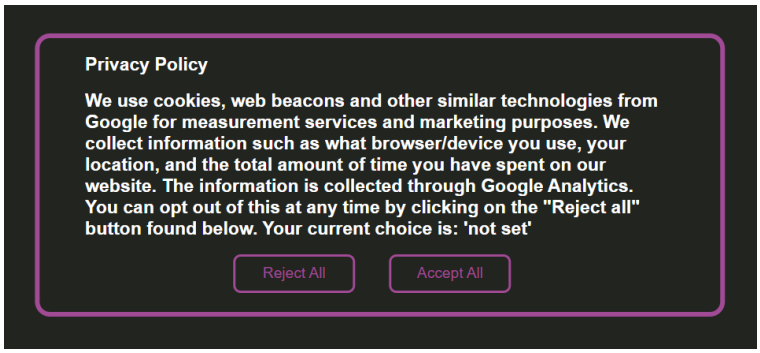
2.5 Analysering av data

Nettsiden bruker et importert analyseringsverktøy som kan måle en rekke med forskjellige informative detaljer om de som besøker siden. Dette innebærer blant annet hvor lang tid noen har brukt på nettsiden, enheten som ble brukt, dato og klokkeslett for besøket og lokasjonen til den besøkende. Dette er informasjon som er svært ønsket av Frostbite, noe som førte til at det ble en viktig del av utviklingsprosessen. Som sagt kan man ikke bare samle inn slik data helt uten videre, så hver eneste person som besøker nettsiden får valget om å godta eller ikke godta dette. Et såkalt «pop-up» vindu blir vist for besøkende som besøker nettsiden for første gang, etter at en besøkende har valgt en av mulighetene vil svaret bli lagret i localStorage, noe som fører til at de kun trenger å se dette «pop-up» vinduet en gang med mindre de tømmer hurtigminnet sitt. Selv om «pop-up» vinduet forsvinner kan besøkende selvfølgelig alltid oppdatere valget sitt uendelig mange ganger gjennom «cookie settings» vinduet om de ombestemmer seg. Dette vinduet viser blant annet også personvernerklæringen som informerer den besøkende om hva slags informasjon som blir lagret, hvilket verktøy og teknologier som blir brukt, og hvordan man enkelt kan oppdatere valget sitt.



Figur 9: vinduet nederst på siden som blir vist til folk som besøker nettsiden for første gang.

Om de velger «reject» eller «accept» blir valget lagret i hurtigminnet og vinduet forblir skjult.



Figur 10: «Privacy Policy» vinduet som lar besøkende lese om datainnsamlingen og eventuelt oppdatere valget sitt

«Privacy Policy» vinduet er som sagt et innstillingsvindu som lar de besøkende på nettsiden oppdatere valget sitt angående datainnsamlingen, samtidig informerer det om hvordan, hvorfor og hvor informasjonen blir samlet inn og håndtert.

Om de velger «reject all» vil datainnsamlingsfunksjonene på nettsiden aldri bli initialisert, om de velger «accept all» vil datainnsamlingsfunksjonene kjøre hver gang de besøker nettsiden til de eventuelt oppdaterer valget sitt på nytt gjennom «privacy policy» vinduet. Dette vinduet er lett tilgjengelig helt nederst på siden. Den innsamlede informasjonen kan for eksempel bli brukt til å finne ut hvor Frostbite burde arrangere sin neste konsert (basert på deres popularitet i forskjellige land og byer). Dataen var også en viktig del av brukertestingen, samt tilbakemeldingene vi fikk fra Frostbite.

Kapittel 3: Metode

Kapittel 3.1 Forskningsmetode

For dette konkrete prosjektet defineres den vitenskapelige metoden som hypoteseskriving, trådskissering, implementering, deploying og observering. Deretter ny hypoteseskriving med tanke på forbedringspotensialet til det som ble testet. Dette ble oppnådd gjennom analysering av observasjoner samt tilbakemeldinger fra testerne. I vårt tilfelle var testerne forskjellige medlemmer og fans fra Frostbite. Vi mottok slike tilbakemeldinger annenhver uke ved å bruke et agilt scrum rammeverk for utviklingsprosessen vår.

Enkelt forklart vil det si at vi gradvis utviklet nettsiden, og annen hver uke testet Frostbite den ut for seg selv og kom med innspill i hva som burde bli lagt til, fjernet, forbedret og forandret. Nettsiden ble hovedsakelig utformet basert på disse testene og tilbakemeldingene.

Kapittel 3.2 Valg av teknologi og utviklingsmetode

Kapittel 3.2.1 Utviklingsprosess og metode

Utviklingsprosessen som ble brukt for dette prosjektet besto av utvikling gjennom et agilt scrum rammeverk. Dette var hovedsakelig bygd opp av ukentlige sprinter, disse sprintene inkluderte både issue boards, videre utvikling, sprint reviews og til slutt sprint tester med arbeidsgiver annenhver uke.

Versjonsstyring er gjort med GIT og ligg som et GitHub prosjekt på GitHub. For å pushe/pulle kode, so brukte vi i hovedsak den interne GIT funksjonen til Webstorm IDE. Der var ingen begrensinger på hvem som kunne pushe kode til remote repository. Merge konflikter ble løst for hånd hvis store forandringer ble gjort samtidig som andre skrev kode.

For kommunikasjon var det brukt Discord som plattform. Den var gunstig siden en kan sette opp privatserver og ha flere underkanaler for forskjellig tema (f.eks nyttige linker/ressurser, kommunikasjonstråder osv..). Støtte for video strømming og voice chat er også nyttig.

Kapittel 3.2.2 Struktur og sammenheng

Nettsiden består hovedsakelig av en «front-end» og en «back-end». Front-end har hovedansvar for domene, hosting og lagring av kildekoden. Back-end har hovedansvar for variabel- og datalagring, samt innlogging, autentisering og sikkerhet. Disse kommuniserer med hverandre hele tiden for å alltid opprettholde det nyeste innholdet og designet til nettsiden. Dette fører til at de besøkende ikke engang trenger å laste inn siden på nytt for å se det nyeste innholdet, det blir automatisk lagt til i nettleseren deres selv om nettsiden blir oppdatert fra en annen datamaskin gjennom administrator menyen. Dette gjelder for mesteparten av administrator funksjonalitetene.

Kapittel 3.2.3 Front-end

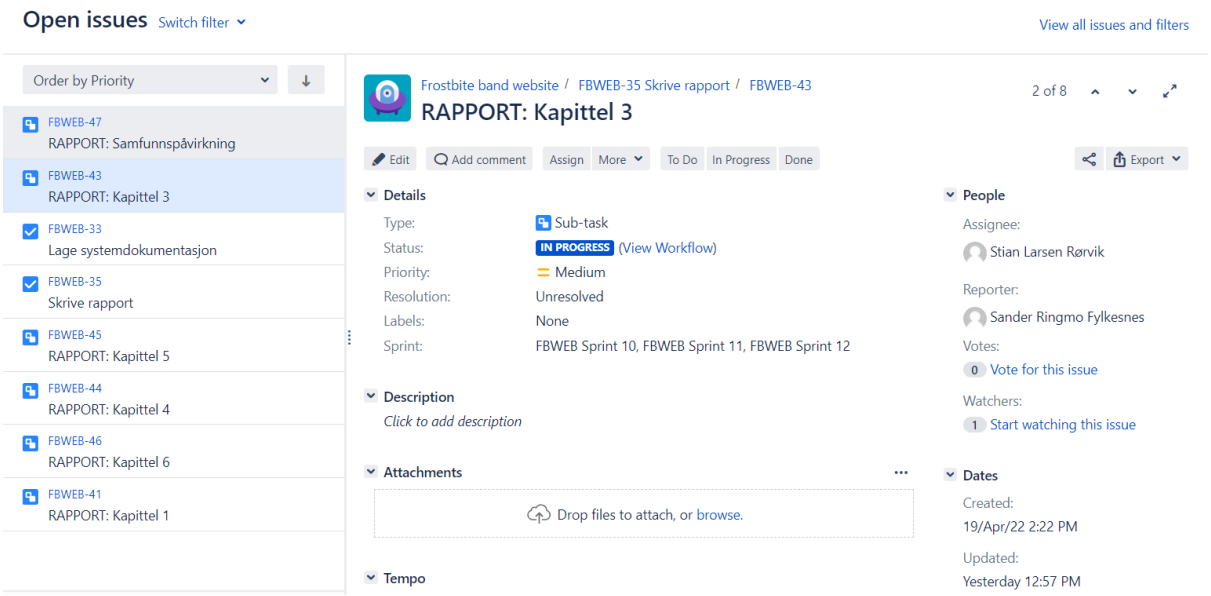
Som sagt består front-end av domene, hosting og lagring av kildekoden. Det er her selve nettsiden blir opprettholdt. Kildekoden består av HTML, CSS og JS. Disse språkene ble valgt fordi de består av den mest kjente og vanligste hoved-teknologien som blir brukt for å utvikle nettsider. Det passet altså perfekt for problemstillingene til prosjektet.

Kapittel 3.2.4 Back-end

Back-end består av variabel- og datalagring i en realtime database, samtidig har back-end også hovedansvar for autentiseringslogikken og sikkerheten til nettsiden. Dataen som blir lagret inneholder blant annet bilder og detaljert informasjon om utseende til selve nettsiden, for eksempel fargekoder, hyperlenker og tekstlig innhold. Grunnen til at denne dataen blir lagret i en ekstern back-end databasen er fordi det åpner opp muligheten for å redigere utseende på nettsiden uten å forandre på selve kildekoden. Nettsiden henter altså disse variablene fra databasen hver gang noen besøker nettsiden, og deretter justerer utseende på nettsiden basert på dataen som blir hentet. Denne dataen kan enkelt bli forandret av Frostbite gjennom den integrerte administrator menyen vi har implementert på nettsiden, Frostbite trenger derfor ikke å kontakte en utvikler hver gang de ønsker å forandre noe på nettsiden, de har full kontroll over innholdet selv. Dette er altså en effektiv løsning for de kompliserte problemstillingene.

Kapittel 3.2.5 Arbeids- og rollefordeling

Arbeids- og rollefordelingen foregikk i Jira. Hovedsakelig arbeidet vi med enkelte problemer i fellesskap, men mesteparten av arbeidsoppgavene ble fordelt utover gruppemedlemmene basert på tidligere kunnskap og spesifiserte roller.



Figur 11: Rollefordeling i Jira

Hver av oss tok på seg flere roller hver og fordelte oppgavene basert på det. Rollene forklarer hovedansvaret, men begrenser ikke hva oppgaver en kan gjøre utenfor det hovedansvaret. Hovedrollene som vi tok på oss tidlig i prosjektet ble der gjennom hele utviklingsprosessen. Under kan du lese om hovedrollen til hvert enkelt gruppemedlem.

Sander Ringmo Fylkesnes tok på seg SCRUM master rolle, det vil si sette opp sprint, arrangere møte, lage Tasks, User stories...osv. Utenfor det, var det tid for å jobbe med utviklingen av nettsiden, skrive på rapport og annet støttende materiale.

Mahmod Ahmed Mohammed hadde ansvar for design av nettsiden der en mellom laga trådskisse av nettsiden og endret trådskissa etter ønske fra Frostbite. Det ble også jobbet med de ulike vedleggene for rapporten og skriving på sjølve rapporten og utvikling av nettsiden.

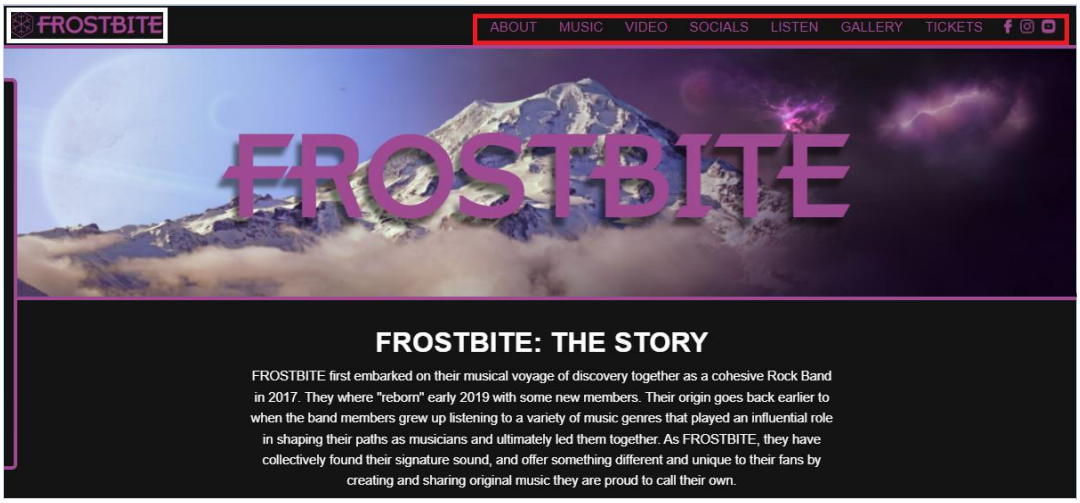
Stian Larsen Rørvik hadde hovedansvar for kodekvalitet (sørge for at koden er av god kvalitet), kodedokumentasjon (inkluderer blant annet dokumentasjon av funksjoner og metoder) og kodeverifikasjon (sørge for at koden gjør det den skal på en sikker og «fail-safe» måte). Dette innebærer spesielt administrator menyen og dens funksjonsrike innhold. Samtidig arbeidet han også med rapportskriving.

Kapittel 4: Resultater

Kapittel 4.1 Vitenskapelige Resultater

Dette er et systemutviklingsprosjekt og har derfor tynt med vitenskapelige resultat. Det var hovedsakelig to punkt (et beskrevet tidligere i dokumentet).

1. **Rekkefølge på innhold:**
   Menneske leser innhold på nettsider i toppen til venstre først ifølge denne studien (Yahoo, 2013). På andre plass kommer generelt toppen av nettsiden. Derfor har vi plassert innhold som logo og navn i denne seksjonen for å kjapt identifisere for leseren hva nettsiden er om. Dermed kan de og hoppe til hvor de vil på siden so kjapt som mulig.



Figur 12: Basert på flere studier vil besøkende først se innholdet i den hvite boksen, deretter innholdet i den røde.

2. **Fargeblindhet:**
   Dette er nevnt tidligere i dette dokumentet under kapittel 2.1.
   Det var viktig for oss at websiden er tilgjengelig for så mange som mulig, derfor er den designet slik at selv om en er fargeblind, så er siden like leselig.
   Vi har brukt høy kontrast tekst og enkle ikon som er uavhengig av fargesyn.
   Denne artikkelen viser til hvorfor dette er viktig: (Design Mantic, 2019)

Kapittel 4.2 Ingeniørfaglige Resultater

Kapittel 4.2.1 Status for problemstilling 1

Den største problemstillingen var hvordan vi skulle iverksette muligheten for tilpassing av websiden uten å forandre selve kildekoden. Dette er for at bandmedlemmene vil ha administrator rolle, men har lite erfaring med HTML/CSS/JS.

Dette ble løst med bruk av en ekstern back-end som kommuniserer med front-end. Administratorer (band medlem) bruker administrator funksjoner (utviklet av oss) for å oppdatere data på en ekstern DB (Firebase) med enkelt forstått pop-ups som forklarer steg for steg hva en må gjøre. Front-end henter da den oppdaterte informasjonen fra databasen.

Det tok flere tekniske løsninger for å løse denne problemstillingen. Det var funne at dette blir ganske knyttet til problemstilling 3 (delen om å lage egen back-end, eller bruke eksisterende løsning). Det ble satt opp en «Real-time» database med Firebase. Data med tekst format blir lagret i NoSQL databasen, mens annen media (bilde) blir lagret i Firebase Storage, begge hentet gjennom samme API.

Individ med administrator rolle kan bruke administrator menyen og følge steg for steg hva en må gjøre for å oppnå målet sitt (for eksempel bytte banner-bilde). Den vil da åpne en filutforsker og brukeren kan da velge bilde som de vil laste opp til Firebase (som nettsiden henter bilde fra). Med denne løsningen kan en da forandre innhold uten å måtte forandre noe i koden.

Kapittel 4.2.2 Status for problemstilling 2

En annen problemstilling er hva hosting løsning som passer best dette prosjektet. Det er nødvendig siden den skal være offentlig tilgjengelig.

Den endelige løsningen som vi ble enige om var One.com, som har en enkel hosting service.

Flere alternativer var tatt i betraktning for hosting av nettsiden. Alternativene ble Linode, Hostgator, BlueHost og One.com. Fordelene og ulempene blir diskutert i neste kapittel. One.com var den servicen som passet dette prosjektet best for en rimelig pris og har de funksjonene som var nødvendig. En kan lett oppdatere koden til nettsiden med å legge inn de oppdaterte filene og administrator treng ikke sette opp noe dedikert server som Nginx for å hoste siden på nett.

Kapittel 4.2.3 Status for problemstilling 3

Hva teknologier å anvende er en problemstilling som er relevant. Dette vil si blant annet om det er nødvendig å bruke rammeverk. Skal vi lage egen back-end, eller koble opp mot en eksisterende service?

Vi valgte å ikke ta i bruk noen spesielle rammeverk og brukte heller HTML/CSS/JS til å kode nettsiden. Vi valgte også å koble oss opp mot googles databasesystem «Firebase», som nå er den eksisterende servicen for vår back-end.

Vi diskuterte om vi ville bruke rammeverk eller ikke for å utvikle nettsiden tidlig i prosjektet. Et forslag var å bruke React.js, dette er et JavaScript-bibliotek vi tidligere har hatt erfaring med. Vi fant ut at dette ikke var nødvendig siden Frostbite kun ønsket en minimalistisk hovedside uten noen sub-sider. Derfor bestemte vi oss for å bruke standard HTML/CSS/JS for å utvikle nettsiden i stedet.

Valget om back-end service gikk også ganske greit. Vi valgte Firebase sin Realtime database for vårt prosjekt og skylageret «Firebase Storage» for lagring av mediefiler.

Kapittel 4.2.4 Status for problemstilling 4

En problemstilling ganske sentral i webdesign er hvordan UI/UX skal både passe estetisk til hvordan kunden vil ha det, men også passe på at grensesnittet er intuitivt for individ som besøker den.

Frostbite ønsket et minimalistisk design, dette førte til at innholdet ble på en og samme side. Vi brukte god avstand mellom de forskjellige seksjonene som førte til at nettsiden ble lett forståelig og enkel å navigere. Samtidig implementerte vi også en navigasjonsbar som da ble brukt som auto-scrolling til de forskjellige seksjonene på nettsiden. Nettsiden er også i hovedsak designet i samme stil som nettsidene til andre musikk band og det var også det Frostbite ville gå for. Utseende og innholdet ble selvfølgelig bestemt av Frostbite.

Kapittel 4.3 Administrative Resultater

Når det kommer til selve utviklingen av oppgaven, er fordeling av ansvar ganske viktig for å ha god fremgang med prosjektet og jevnt arbeid.

Grunnet at dette er en utviklingsoppgave ble en agil metode valgt. Det ble valgt å bruke en SCRUM metode. Prosessen bestod av en sprint hver uke der vi satt et mål for hva som skulle bli oppnådd og hvem som hadde ansvaret for hver «task» og «story». Etter hver sprint ble det foretatt et «sprint review» der vi kunne diskutere hva som gikk bra og hva som må forbedres for neste sprint.

Utenom selve sprintene var det holdt møter med veileder og arbeidsgiver annenhver uke. På de møtene fikk vi tilbakemelding som formet nettsiden videre.

Etter planen for gjennomføring (funnet i forprosjektplanen) holdte vi de viktige fristene, nettsiden ble finpusset fram til en uke før innlevering. User stories og «tasks» som ikke ble ferdig i planlagte sprinter ble rullet over i neste med gode resultater. Det vil si at noen mindre milepæler ble forskjøvet litt, men ingen av de kritiske.

Kapittel 5: Diskusjon

Kapittel 5.1 Diskusjon Generelle Resultater

Kapittel 5.1.1 Diskusjon plassering av innhold

Vi så på noen artikler om hvordan individ ser på en nettside for å finne en god måte å plassere innhold på.

Det som var gjort bra var at plasseringa som ble endt opp med å gi en god opplevelse der alt innhold er på en intuitiv plass og sida er klart oppdelt og bruker minst mulig museklikk

Det som kunne blitt bedre er plasseringen på mobile enheter. Der er noen tilpasninger gjort for mindre skjermer, men kunne ha blitt mer tilpasset om vi fikk tid til videre utvikling.

Kapittel 5.1.2 Diskusjon Fargeblindhet

For en nettside så er det viktig at den er so tilgjengelig som mulig. Mange individ er fargeblinde (spesielt for menn der rundt 5 til 8% er det) (Norges Blindeforbund, 2022), det er da optimalt om de kan bruke nettsiden uten hindring.

Løsningen som var kommet fram med er effektiv i å la siden være leselig selv om ikke kan se farger i det hele tatt. Det som spesielt hjelper er høy kontrast mellom bakgrunn og tekst og enkle ensfarget ikoner der fargesyn hjelpe stilen, men hemmer ikke funksjonaliteten.

Der er en bakside som kommer av dette. For å vedlikeholde denne tilgjengeligheten, so må musikkgruppa (Frostbite) forsikre seg å bruke lignende enkle stiler i fremtiden, som kan begrense kreativ fridom i framtiden.

Kapittel 5.2 Diskusjon Ingeniørfaglige Resultater

Kapittel 5.2.1 Diskusjon av resultat problemstilling 1

I vår mening så er løsningen vi kom fram med et godt svar på problemstillingen.

Å lage til en egen administrator meny i front-end av nettsiden gjør det mye enklere for band medlemmene å gjøre forandringer på innhold. Det er grunnet at en blir vist hvordan man gjør den handlingen en vil gjøre gjennom «pop-ups» etter som en trykker på knapper i administrator menyen. Alle funksjonene er skrevet på allmennspråk som er lett forståelig for administrator.

Denne løsningen med bruk av egenlaget administrator meny og separat back-end gjør det også mulig å se forandringen i ektetid, dermed trenger ikke avansert kunnskap om webutvikling so lenge forandringene er innenfor hva menyen lar deg gjøre.

Der er selvsagt svakheter med en slik løsning. En svakhet er avhengigheten til Firebase som back-end, det er ikke kjapt å bytte over til en annen løsning. Er mulig, men det vil ta ekstra tid å gjøre om på systemarkitekturen. Viss en vil gjøre større forandringer utenfor administrator menyen, må en forandre på koden til nettsiden.

Prosessen vi brukte for å utvikle produktet hadde stor betydning for hvordan vi kom frem til den. Frostbite fikk prøve løsningen flere ganger under utviklingen og tilbakemeldingene/ønskene vi fikk ble som sagt implementert.

Generelt så er Frostbite storfornøyd med løsningen.

Kapittel 5.2.2 Diskusjon av resultat problemstilling 2

For gruppen vår sin mening, så er løsningen på hosting passende. Som beskrevet i kapittel 4, så skal vi drøfte hosting løsningen i forhold til de andre.

One.com: Denne hosting servicen har flere positive sider. One.com er en «all-in-one» løsning, der domene kan være registrert med Storage E-post og hosting av webside i en. Dette gjør for en enkel løsning der en trenger ikke å konfigurere egen Nginx server for å hoste nettsiden. Grunnen til det er positivt er at en kan oppdatere filene ved hjelp av en GUI som er lett å bruke.

Det negative er om en vil selv konfigurere server, så er ikke muligheten for å gjøre det der og må settes opp separat fra one.com (filene kan fortsatt ligge der), men domene kan selvsagt bli på one.com. Dette er ekstra steg om en vil ha noe særegen hosting.

HostGator: Det positive med HostGator er at de har flere hosting planer en BlueHost og ifølge denne testen, har bedre ytelse: (isitwp, 2022). Ellers er disse servicene veldig like i hva du kan gjøre med dem. Der er også inkludert støtte for MySQL database.

Det negative er at HostGator er litt dyrere.

Linode: Det positive med Linode er at en kan konfigurere egen Nginx server (eller hva annen service en vil). Linode er mer ment for generelle web baserte løsninger (som f.eks. hosting av spill-servere).

Det som gjør at Linode ikke passer helt til dette prosjektet er at vi bruker en separat back-end og dermed ikke treng den funksjonaliteten, men heller en enklere service.

BlueHost: Det positive med BlueHost er den lave prisen for hosting, ellers er det veldig likt HostGator. Den har noen ekstra tilbud som gratis domene i et år og SSL sertifikat.

På den negative siden så er det ingen "features" som skiller seg ut fra de andre og ytelsen av hostingen er ikke den beste: (isitwp, 2022)

Resultat:

Når vi tar i betrakting de positive og negative sidene med servicene så vinner one.com. Grunnen kommer av at vi trenger ikke de mer avanserte funksjonene som de andre servicene tilbyr for å oppnå målet om å hoste siden.

Kapittel 5.2.3 Diskusjon av resultat problemstilling 3

Et resultat av vår problemstilling om hva teknologier som skulle implementeres under utviklingen var å ikke bruke rammeverk. Grunnen til at det ble slik er at rammeverk er vanligvis nyttige i «large-scale» systemer og applikasjoner, og med tanke på dette prosjektet var et slikt rammeverk derfor unødvendig. En annen grunn er at vi har mer erfaring med bruk av vanlig JavaScript enn React.js, og sammen med at det ikke var nødvendig med bruk av rammeverk, valgte vi derfor å ikke ta i bruk rammeverk.

Vi valgte Firebase sine back-end funksjoner fordi vi tidligere har brukt det som back-end og er kjent med bruken av Firebase. Vi følte det gjekk greit å bruke Firebase siden veldig mye av det tekniske er håndtert av Firebase, og det var derfor enkelt for oss å starte opp vår back-end. I tillegg passet det våre behov angående datalagring, autorisering, pris og sikkerhet. Firebase hjelper også oss med å koble nettsida mot Google Analytics slik at Frostbite kan lett kan analysere trafikken på nettsida. Kort sagt så samler Firebase mange funksjoner som vi trenger på en plass og gjør ting enklere for oss.

Men det er verdt å notere seg noen av problemene som oppstår ved bruk av Firebase. Et problem med Firebase blir at gratisversjonen ikke støtter større trafikk på nettsiden. Mediefiler som nettsida lagrer, har en bredbåndskvote og lagringskvote. Du får et visst antall lagringsplass og du kan sende et begrenset antall med data mellom bruker og Firebase Storage som betyr at Frostbite som er administrator på nettsida, ikke kan laste opp store filer på nettsida med mindre de har lyst til å gå over til en betalt versjon.

Et annet problem er at å bruke Google Analytics gjennom Firebase gjør det vanskelig for oss å gi brukere av nettsiden mulighet til å tilpasse hva typer informasjon nettsiden kan hente fra brukere gjennom «cookies». Akkurat nå kan brukere kun akseptere eller avvise alle cookies og har ikke mulighet for å tilpasse.

Kapittel 5.2.4 Diskusjon av resultat problemstilling 4

Om vi skal diskutere i hvilken grad vår løsning oppfyller krava for god UX/UI så kan vi si at vi er fornøyde med resultatet. Det er fordi at nettsiden har blitt utviklet gjennom mange iterasjoner med Frostbite der Frostbite har vært med på å utforme hvordan nettsiden skal se ut og om brukeren har en god opplevelse når han interagerer med nettsiden. Det betyr at løsningen vår kommer så nært som mulig ønskene til vår Frostbite. Frostbite er veldig fornøyd med hvordan UI/UX for nettsiden er implementert og i tillegg kan de tilpasse UI med administrator-funksjonene noe som er et pluss for dem.

Det er fortsatt noen forbedringer som en kan utføre på nettsiden. Noen av bildene på nettsiden er ikke satt opp til å ha alternativ tekst slik at når en person som er avhengig av skjermleser for å navigere nettsiden, så vil den personen (om han for eksempel er blind) ikke ha mulighet til å få med seg noen av bildene som er på nettsiden.

Kapittel 5.3 Diskusjon Administrative Resultater

Vi valgte å bruke en agil metode overfor en tradisjonell metode slik som for eksempel «waterfall» metoden. En av hovedgrunnene til at vi bestemte oss for å bruke en agil metode slik som SCRUM er at det er anbefalt at vi holder iterative møter hver andre uke med arbeidsgiver. Vi var også avhengig av tilbakemeldingene vi fikk fra møtene gjennom utviklingen av nettsiden og derfor ville en lineær arbeidsmetode ikke fungert for oss.

Kapittel 6: Konklusjon og videre arbeid

Vi kan konkludere med at kravene som Frostbite har gitt oss (se Oppgavetekst) er tilfredsstilt ut ifra tilbakemeldingene som Frostbite har gitt oss. Ytterligere ønsker fra Frostbite slik som å gi Frostbite mulighet til å tilpasse siden etter behov knyttet til problemstilling 1 (se under Kapittel 1.1: Problemstillinger) har også blitt tilfredsstilt og de er veldig fornøyde med arbeidet.

Konkluderende svar for problemstilling 1 er som nevnt i avsnittet over, at Frostbite har mulighet til å tilpasse nettsiden med tanke på utseende slik de vil. Den største utfordringen som problemstillingen stilte var at tilpassing av nettsiden skulle skje uten at administratorer må gå gjennom kildekode for å gjøre endringer, og den utfordringen har vi løst gjennom en administratormeny. Frostbite vet også om svakhetene av løsningen som er nevnt under diskusjon av resultatet til problemstilling 1 i kapittel 5 (kapittel 5.2.1), og de er klar over hvordan de kan håndtere de svake sidene av løsningen, slik som å ansette utviklere visst de vil migrere til annet system enn Firebase.

For problemstilling 2 som angår hosting av nettside kan vi konkludere med at Frostbite er fornøyd med valg av hosting service. Ettersom vi sammenlignet fordeler og ulemper ved de ulike hosting-løsningene i kapittel 5.2.2, kom vi fram til slutt at servicen som One.com tilbyr passer best for Frostbite.

Problemstilling 3 som handler om valg av teknologier slik som rammeverk og hvordan vi vil håndtere forholdet mellom back-end og front-ent har vi konkludert med at det ikke er behov for å bruke rammeverk for vårt prosjekt på grunn av at omfanget av vårt prosjekt ikke krever slik teknologi. Vi har også endt opp med å bruke Firebase som er en eksisterende service for back-end fordi vi har erfaring med å bruke Firebase og fordi den har samlet alle funksjonene vi trenger på en plass. Utfordringene med Firebase som er nevnt i kapittel 5.2.3 er ikke så relevante siden Frostbite har sagt at de er villig til å gå over til en betalingsplan om det blir nødvendig.

Konklusjonen for problemstilling 4 er at nettsiden har løst Frostbite sine krav for UI/UX for nettsida. Frostbite har vært med på å utforme nettsiden og vi kan trygt si at nettsiden sin UI er slik Frostbite vil ha det. Selv om siden mangler alternativ tekster for bildene på nettsiden, så er det ikke kritisk for at brukere med behov for skjermlesere har mulighet til å kjenne til dem. Siden Frostbite er et musikk-band så er det viktigste at brukerne får høre musikken, noe som de kan gjennom integrert YouTube og Spotify spillere.

Videre Arbeid

For de som har lyst til å jobbe videre med prosjektet kan finne koden på GitHub. Lenken til koden på GitHub finner dere i systemdokumentasjonen som befinner seg i vedleggene for dette dokumentet. Veiledning for hvordan man kloner et prosjekt fra GitHub til ditt utviklingsmiljø finner dere her (i dette eksempelet er det IntelliJ som blir brukt men det skal være likt for det fleste utviklingsmiljø). Koden er godt dokumentert og det er anbefalt å lese nøye gjennom dokumentasjonen for å få en helhetlig forståelse av koden.

Koden for back-end av prosjektet er sterkt avhengig av Firebase, så om du skal gjenbruke koden for et annet prosjekt så er det lurt å starte et nytt Firebase prosjekt visst du har tenkt legge til en back-end for ditt prosjekt. Siden Firebase bruker en «Realtime» NoSQL database så blir det kanskje lettere å gå over til AceBase (som er inspirert av Firebase) om du har lyst til å bruke en annen løsning enn Firebase. Du kan finne ut mer informasjon om AceBase her.

Samfunnspåvirking

Økonomi

Kostnad: Direkte kostnader under utvikling er veldig lave. Der var ingen kostnad for selve utviklingen av nettsiden i seg selv, det eneste som koster er hosting tjenesten one.com.

Det var valgt å gå for den billigste planen som de tilbyr, som er en delt server (den fysiske maskina er delt med andre prosjekt, vi har bare en virtuell maskin som den kjører på). Dette endte opp med 40Kr per måned, som er en veldig lav pris.

Der er en potensiell kostnad om musikkgruppa blir større, det er Firebase back-end.

Viss mer enn 100 koblinger er oppe samtidig, so vil de måtte oppgradere til en betalt versjon. Den betalte versjonen er heller ingen fast pris, men en «betal som du bruker» plan, det vil si lav pris med lav bruk og høyere pris med høyere bruk. Det har ikke ennå nådd den grensen.

Fortjeneste: Produktet generer ikke direkte fortjeneste i seg selv, men fungerer som en plattform hvor musikkgruppen kan vise fram portfolioen sin og linke til portaler hvor en kan bestille billetter. En kan si at nettsiden bidrar med å generere trafikk for bandet, som indirekte øker fortjeneste.

Miljøpåvirkning

Prosjektet har en veldig liten påvirkning på miljøet, men der er fortsatt en faktor.

Energiforbruk: Nettsida tar i bruk flere nettservicer som bruker elektrisk energi nemlig Firebase og One.com.

For å bruke minst mulig energi, so var en delt server brukt (det vil si bruk av VM, men med delt maskinvare ressurser). Fotavtrykket til one.com servere er og veldig lav i drift på grunn av 100% fornybar energi i følgje One.com: (One.com, 2022).

Firebase er det mye vanskeligere å finne ut om fotavtrykket. Siden Firebase er eid og operert av Google, kan en gå ut ifra at energi er like ren som Google Cloud bruker. Google Cloud lover å få sin energi fra 100% fornybare kilder innen 2030 (Google Cloud, 2022).

For å videre redusere energibruket kunne en ha Front-End og back-end på samme fysisk maskin, men det hadde krevd å gjøre om på hele systemarkitekturen.

Andre Bærekraft Vurderinger: Dette prosjektet har ikke andre store påvirkninger på samfunnet, men der er noen mindre effekter.

En effekt er hva det gjør for folk med fargeblindhet og/eller svakt syn. Grunnet sterk kontrast, solid farge bakgrunn og god klar plassering av element gjør siden mye lettere å lese for denne gruppen folk. Effekten blir at de mer sannsynlig finn ut mer om bandet. Dermed blir det kulturelt beriket av musikkgruppen sin musikk.

Diskutering av profesjonsetiske problemstillinger

Drøfting: Sander Ringmo Fylkesnes

Der er ikke mange relevante problemstillinger av denne natur for denne oppgaven.

En slik problemstilling som kan forekomme under utviklingen er hvordan å forholde seg til lover og regler som gjelder offentlige nettsider og hva en skal gjøre om arbeidsgiver motsier et slikt design.

Siden dette er om lov og regler so må en først forholde seg til de reglene og rammeverkene (f.eks. WCAG). Dette er en vanskelig konflikt å løse, men tilpasninger og kompromiss må skje om noe slikt oppstår. Å forklare hva som bryter regelverk til arbeidsgiver på en god måte er viktig for å komme fram til et design som er lovlig og arbeidsgiver blir nøgd nok med.

Drøfting: Mahmod Ahmed Mohammed

Problemstillinger som kan oppstå i gruppen er at gruppemedlem melder at de er syke når de ikke er det for å unngå å gjøre arbeid. Som gruppe skal vi hjelpe hverandre når noen står fast eller har vansker, spesielt visst de er syke og sliter med å gjøre sin del av jobben. Men å prøve å fremstå som syk uten å være syk vil være et stort brudd på tilliten mellom gruppemedlemmene og er derfor viktig at vi er ærlige med hverandre.

Drøfting: Stian Larsen Rørvik

Andre eksempler på profesjonsetiske problemstillinger som kan oppstå under dette prosjektet inkluderer blant annet motstridene krav. For eksempel om NTNU har krav som ikke samsvarer med det arbeidsgiver ønsker å ha på nettsiden, og arbeidsgiver ønsker funksjoner som ikke samsvarer med NTNU sine krav. Hva skal vi inkludere? hva skal vi droppe? Det er altså viktig for oss å velge riktig innhold slik at vi utvikler en nettside som oppfyller kravene til begge parter på en god og effektiv måte.

Referanser

Design Mantic. (2019, Ukjent dato). *Design Mantic*. Hentet April 22, 2022 fra https://www.designmantic.com/community/website-design-guide-color-blind.php

Google Cloud. (2022, Mai 4). *Google Cloud*. Hentet fra Google Cloud sustainability: https://cloud.google.com/sustainability

isitwp. (2022, Februar 5). *Isitwp.com*. Hentet fra Isitwp.com: https://www.isitwp.com/hosting-reviews/compare/bluehost-vs-hostgator/#:~:text=Here%E2%80%99s%20how%20HostGator%20stacks%20up%20against%20Bluehost%20in,45-day%20money%20back%20guarantee.%20...%20More%20items...%20

Loshin, P. (2022, Februar). *TechTarget*. Hentet April 21, 2022 fra https://www.techtarget.com/searchdatamanagement/definition/SQL

Mozilla . (2022, April 13). *mdn web docs*. Hentet April 20, 2022 fra https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS

Mozilla. (2022, Februar 18). *mdn web docs*. Hentet April 20, 2022 fra https://developer.mozilla.org/en-US/docs/Web/HTML

Mozilla. (2022, April 13). *mdn web docs*. Hentet April 21, 2022 fra https://developer.mozilla.org/en-US/docs/Web/JavaScript

Nginx. (2022, Mai 3). *nginx.org*. Hentet fra nginx.org: https://nginx.org/en/

Norges Blindeforbund. (2022, Mai 6). *Norges Blindeforbund*. Hentet fra Norges BlindeForbund: https://www.blindeforbundet.no/oyehelse-og-synshemninger/fargeblindhet?msclkid=d9b9dff4cd4911ecb2bd1368d0804772

One.com. (2022, Mai 4). *One.com*. Hentet fra One.com: https://www.one.com/en/info/powered-by-wind-turbines

Oracle. (2022, Ukjent dato). *Oracle*. Hentet April 28, 2022 fra https://www.oracle.com/database/what-is-a-relational-database/#:~:text=A%20relational%20database%20is%20a%20type%20of%20database,record%20with%20a%20unique%20ID%20called%20the%20key.

RedHat. (2017, Oktober 31). *RedHat*. Hentet April 29, 2022 fra https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

SCRUM. (2022, Mai 1). *SCRUM.org*. Hentet fra SCRUM.org: https://www.scrum.org/resources/what-is-scrum/

Tutorialsport. (2022, Mai 3). *tutorialsport.com*. Hentet fra tutorialsport.com: https://www.tutorialspoint.com/graphical-user-interface-gui

Yahoo. (2013, Mars 10). *Yahoo*. Hentet April 21, 2022 fra https://web.archive.org/web/20130310105837/http://styleguide.yahoo.com/writing/write-web/eye-tracking-where-do-readers-look-first

ScrollFade. (2021, Oktober 7) *GitHub*. Hentet Februar 16, 2022 fra https://github.com/meddlenz/ScrollFade

Slideshow. (Ukjent dato) *w3schools*. Hentet April 4, 2022 fra
https://www.w3schools.com/howto/howto_js_slideshow.asp

String-to-date. (2016, Juli 26*) Arivan Bastos, stackoverflow*. Februar 23, 2022 fra
https://stackoverflow.com/questions/5619202/parsing-a-string-to-a-date-in-javascript

Hex-to-RGB. (2011, April 11) *Tim Down, stackoverflow*. Hentet Februar 28, 2022 fra
https://stackoverflow.com/questions/5623838/rgb-to-hex-and-hex-to-rgb

Vedlegg

For å unngå konflikter med innholdsfortegnelsene til de forskjellige vedleggene (samt hovedrapporten) velger vi å slå sammen de endelige PDF filene i stedet for å lime vedleggene direkte inn i dokumentet. Du finner rekkefølgen på vedleggene under:

Vedlegg 1: Forprosjektplan

Vedlegg 2: Kravdokument

Vedlegg 3: Systemdokumentasjon

Frostbite Nettside
Forprosjektplan

Versjon 2.0

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
17.01.22	1.0	Opprettelse av dokument	Stian, Sander, Mahmod
18.01.22	1.1	Fortsettelse på dokumentet, la til risikovurdering	Stian, Sander, Mahmod
19.01.22	1.2	Fortsettelse på dokumentet	Stian, Sander, Mahmod
20.01.22	1.3	Fortsettelse på dokumentet, la til Hovedaktiviteter, samt opplisting av noen viktige datoer.	Stian, Sander, Mahmod
16.05.22	2.0	Oppdaterte dokumentet, fjernet seksjon 6 (sensitiv info)	Stian, Sander, Mahmod

Innholdsfortegnelse

1. Mål og rammer 4
1.1 Orientering 4
1.2 Problemstilling / prosjektbeskrivelse og resultatmål 4
1.3 Effektmål 4
1.4 Rammer 5
2. Organisering 5
3. Gjennomføring 5
3.1. Hovedaktiviteter 5
3.2. Milepæler 6
4. Oppfølging og kvalitetssikring 6
4.1 Kvalitetssikring 6
4.2 Rapportering 6
5. Risikovurdering 7

1. Mål og rammer

1.1 Orientering

Hvorfor denne oppgaven. Hvordan fikk dere tak i den.

Vi valgte denne oppgaven for flere grunner. En av grunnene er at den er ganske relevant for det vi har lært i løpet av studie, vi får dermed putte vår kunnskap i praksis. En annen grunn er at vi får jobbe med en ekstern arbeidsgiver. Det vil gi oss mer erfaring med å løse et «real world» problem.

Vi fikk tak i denne oppgaven gjennom instituttet.

1.2 Problemstilling / prosjektbeskrivelse og resultatmål

Problemstilling dere skal utforske i prosjektet. Dette er et første utkast som kan endres dersom premisser endres underveis.

Problemstilling: Frostbite treng en webside for bandet deres, den skal funke som hovedsiden til bandet, det skal også bli inkludert flere funksjoner som gjør nettside justerbar. Denne siden skal gjøre det lettere for bandet å markedsføre seg selv. Nettsiden skal være en plattform der de kan legge ut info til interesserte fans.

Vi skal planlegge, designe og programmere denne nettsiden.

Resultatmålene forteller hva som skal være oppnådd når prosjektet er ferdig.

Resultatmål:

Når prosjektet er ferdig, skal flere funksjoner og innhold være oppnådd. Dette inkluderer:

- Info (om bandet)
- YouTube direkteavspilling
- Linker til sosiale medier
- Integrert Spotify direkteavspilling
- Bilder fra tidligere konserter (i form av et galleri/bildekarusell)
- Konsert info med linkfunksjon til eksterne billettnettsider

1.3 Effektmål

Hva er målet for deg / gruppa og hvilke langsiktige effekter eller gevinster virksomheten søker å oppnå ved å nyttiggjøre seg av resultatet fra prosjektet.

Målet vårt:

Vårt mål er å få mer erfaring med å utvikle nettsider, alt fra selve prosessen til den mer tekniske delen som UI/UX design og koding. Et anna mål er å bli mer vant med å jobbe for eksterne arbeidsgivere (individ/ bedrifter utenfor NTNU) som kan komme godt i bruk når vi skal ut å søke jobb på arbeidsmarkedet.

Gevinst for Frostbite:

Gevinsten for Frostbite er at de får en fungerende nettside som de kan bruke kommersielt. Dette øker også deres online tilstedeværelse, som er veldig viktig i dagens markedsføringsmiljø.

1.4 Rammer

Behov for penger, utstyr og tid. Spesialbehov materialer og rom.

Nettsiden som blir utviklet i dette prosjektet vil bli hostet via en ekstern vertsleverandør, dette kommer til å koste penger når prosjektet nærmer seg slutten. Nødvendig utstyr innebærer bruken av datamaskiner, internett og relevante IDEer (som IntelliJ og WebStorm), samt resurser (bilder, musikk, og andre detaljer) som da vil bli sendt til oss fra oppdragsgiveren.

Vi kommer også til å potensielt ta i bruk labben på campus i sammenheng med prosjektet, i dette tilfellet tar vi i bruk L167 (om det er ledig), vanligvis mellom 08:00 til 14:00 på hverdager.

2. Organisering

Hvilke aktører er involvert i prosjektet.

Aktørene som er involvert i prosjektet er veileder fra NTNU, ekstern arbeidsgiver, og studentene fra NTNU som har fått tildelt oppgaven.

Veilederen til dette prosjektet er Kjell Inge Tomren. Arbeidsgiveren er bandet Frostbite, med Reinhardt Oma som kontaktperson. Studentene er Stian Larsen Rørvik, Sander Ringmo Fylkesnes og Mahmod Ahmed Mohammed.

3. Gjennomføring

3.1. Hovedaktiviteter

Opplisting av hovedaktiviteter.

Hva gjøres, hvem gjør det, hvorfor gjøres det, hvordan gjøres det. Når gjøres det, nødvendige forutsetninger før det kan gjøres, dokumentasjon / resultat av det som ble gjort.

Gruppearbeidet blir utført via agile arbeidsfaser for å oppnå maksimal fleksibilitet, dette sørger for at nødvendig arbeid blir utført der det trengs, samtidig som alle på gruppen får et innblikk i hva og hvordan ting blir gjort. Dette sørger for at alle får være med, og arbeide like mye. Arbeidet blir hovedsakelig utført av oss studenter, med innspill og ønsker fra oppdragsgiver. Vi planlegger hva som skal bli gjort etter møtene vi har med oppdragsgiver annen hver uke, da blir det diskutert hva som må bli forandret/forbedret eller eventuelle andre ønsker som arbeidsgiveren da har. Arbeidet blir utført i sprinter, timeplanen og issue-boardet til disse sprintene finner du på våre Jira/Confluence sider. Resultatet av arbeidet vil bli loggført som en del av rapportskrivingen.

3.2. Milepæler

Opplisting av kritiske datoer.

- Frist for første møte: 14.01.22
- Frist for forprosjektplan: 28.01.22
- Første wireframe skisser av nettsiden: 28.01.22
- Bestemmelse over hvilke teknologier/rammeverk skal brukes for utvikling av nettsiden: 28.01.22
- Første prototype som kan visest fram: 11.02.22 (er kun for å vise fram ide til design)
- Frist for alle wireframe-skisser med alfadesignet: 25.02.22
- Dato for seneste påbegynnelse av Hovedrapport: 04.03.22
- Dato for når vi bør ha fått til hosting: 18.03.22
- Dato når vi ha implementert Spotify og YouTube integrering: 1.04.22
- Ferdig med nettsidens funksjoner og begynt på «finpussing» av UX: 15.04.22
- Muntlig engelsk presentasjon: 22.04.22
- Levere første utakts av hovedrapport: 06.05.22
- Innlevering av ferdig versjon av rapport og ingen videre finpussing av nettsiden: 20.05.22

4. Oppfølging og kvalitetssikring

4.1 Kvalitetssikring

Hvordan sikre kvaliteten på alle arbeidene

Ved å bruke versjonsstyring verktøy som GitHub kan vi kontrollere prosjektet vårt på et mikronivå. Det gir oss en oversikt over hvilket arbeid enkelte på gruppen er ansvarlige for, og vil derfor være enklere å ha orden på hva som må gjøres bedre for hver enkelt. I tillegg vil vi ha ukentlige møter internt i gruppen (og annenhver uke med arbeidsgiver), der vi vil gå gjennom det som vi har gjort og hva som eventuelt kan gjøres betre.

4.2 Rapportering

Til hvem og hvor ofte

Vi rapporterer i form av et møte (zoom) til arbeidsgiver og veileder annenhver uke. Møte tar sted annenhver onsdag, vanligvis fra klokken 12.00-13.00.

5. Risikovurdering

Risikoanalyse som vurderer sårbarheter i prosjektet (hendelse, sannsynlighet, konsekvens og tiltak).

Sårbarheter i prosjektet kan være at folk blir syk. Spesielt nå med tanke på covid-19 så blir sannsynligheten høyere for at sjukdom oppstår i gruppa.

Potensielle konsekvenser av sykdom innebærer at vi ikke får arbeide like effektivt på grunn av manglende arbeidskraft. Alle er avhengig av hverandre for å utvikle prosjektet vårt. I vårt tilfelle vil det si at om en skal gjøre noen endringer på nettsiden som vi skal utvikle, men er avhengig av den syke personen til å utføre arbeidet fullstendig, så blir det dannet en flaskehals i utviklingsprosessen.

For å unngå at en slik situasjon oppstår kan vi prøve å opprettholde godt smittevern. I tilfelle vi blir syke kan vi delegere oppgaver på en slik måte at en person ikke blir avhengig av andre på gruppen og kan jobbe med andres oppgaver som ikke har med sjølve kodingen å gjøre.

En annen sårbarhet er om et gruppemedlem forlater gruppen av ulike årsaker. Sannsynligheten er ikke stor, men om det skjer så blir konsekvensen alvorlige. Arbeidsmengden på gruppen blir større og stresset kan påvirke produksjonskvaliteten. Det er viktig da at vi så best som mulig prøver å unngå en slik situasjon ved å snakke med vedkommende om hva en kan gjøre for å holde personen i gruppen.

Frostbite Nettside
Kravdokument

Versjon 2.3

Innhold

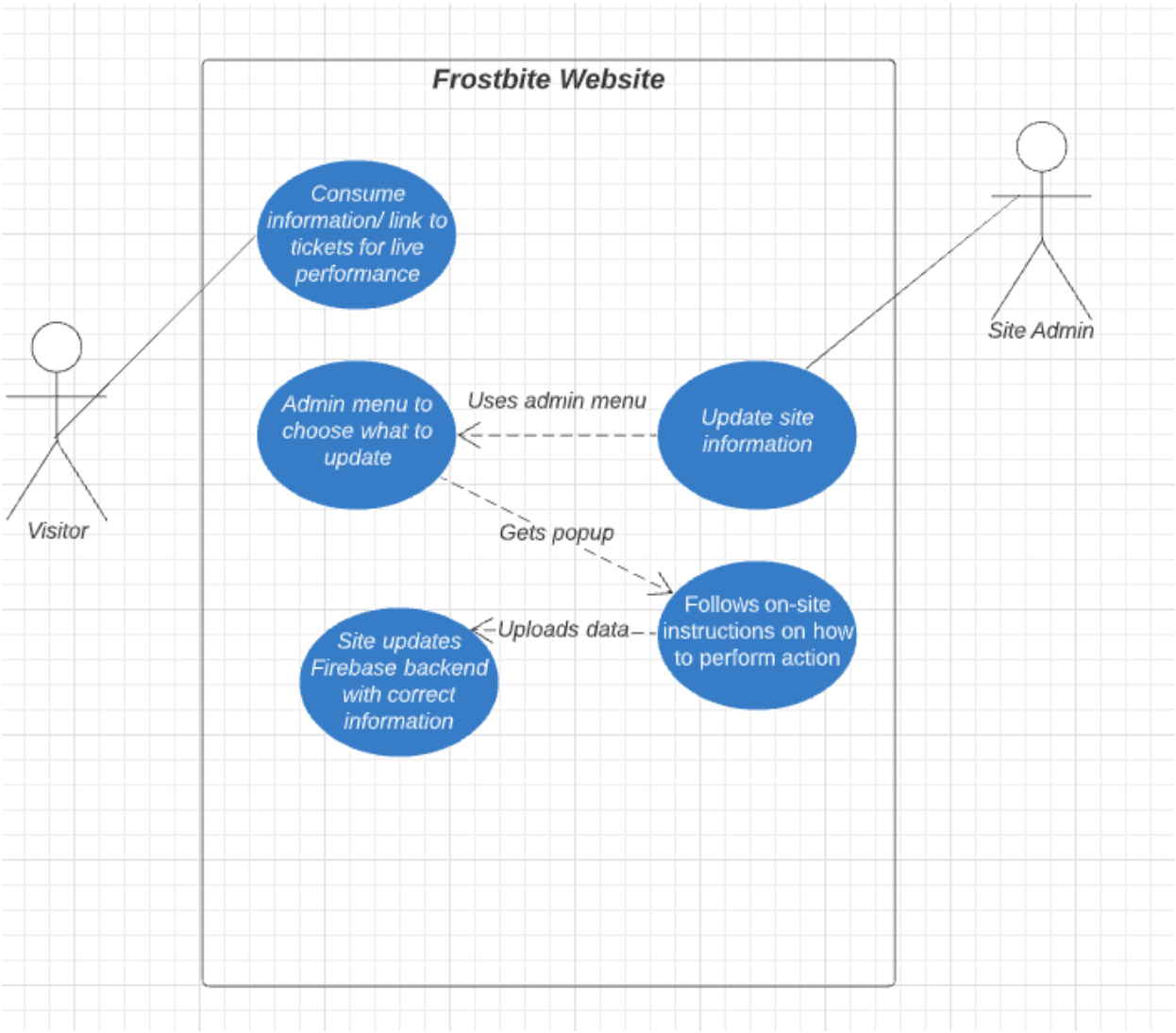
1. Introduksjon 2
2. Use-Case Diagram 3
3. User Stories 4
4. Domenemodell 5
5. Prototyper 6
5.1 Wireframe 6

1. Introduksjon

Hensikten med dette dokumentet er å dokumentere kravene det ferdige produktet må oppnå gjennom en rekke med forskjellige modeller. Disse modellene inneholder altså relevant informasjon om det spesifikke innholdet som nettsiden må ha når prosjektet avsluttes. Disse modellene vil gi deg en utdypet ide om hvordan den endelige siden ser ut. Vennligst merk at design, farge og typer innhold i de forskjellige seksjonene på nettsiden ble bestemt av Frostbite.

Kravdokumentet består hovedsakelig av: Use-Case diagram. user stories, domene modell og endelige versjon av designet til nettsiden (i form av en wireframe).

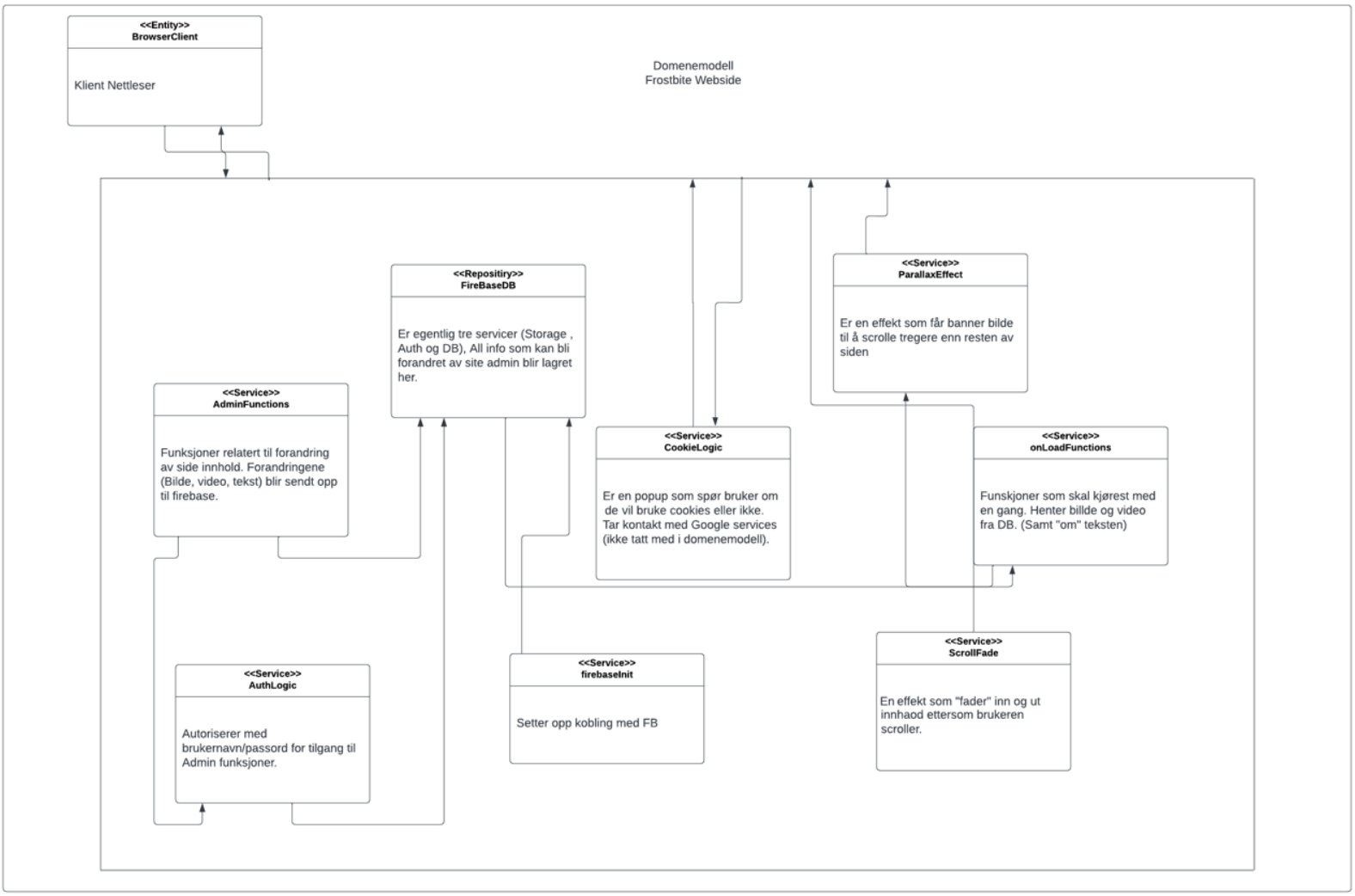
2. Use-Case Diagram



3. User Stories

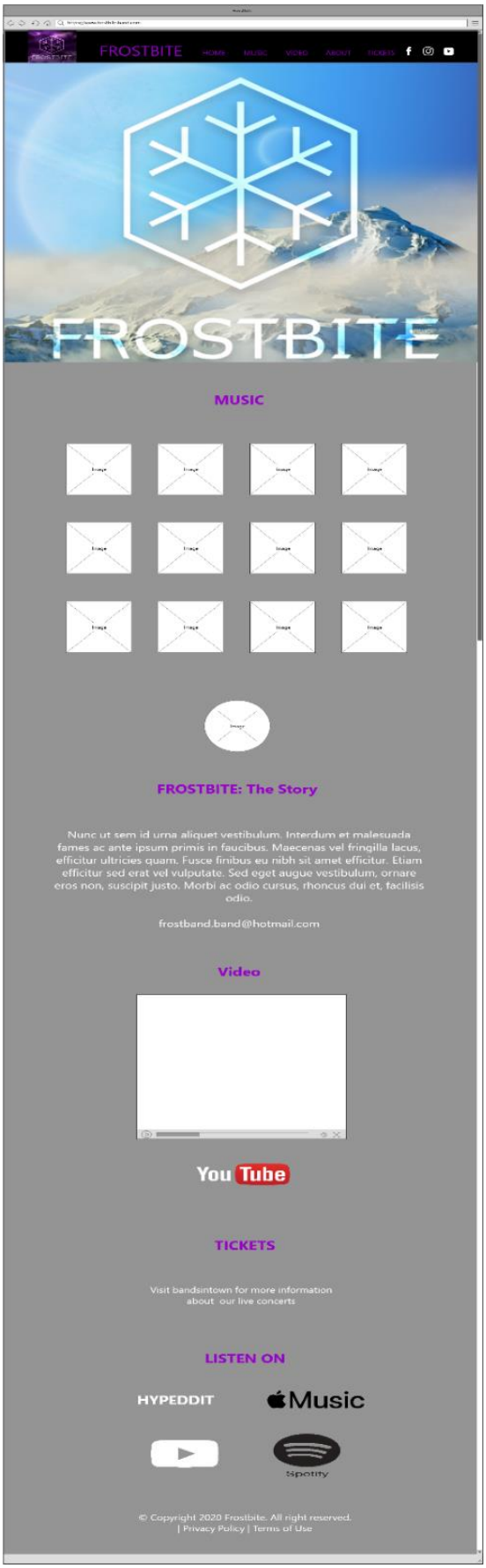
1. Som en besøkende vil jeg kunne se en minimalistisk nettside med masse relevant innhold om bandet «Frostbite»
2. Som et bandmedlem vil jeg kunne vedlikeholde og oppdatere siden uten hjelp fra eksterne utviklere, selv når prosjektet er avsluttet
3. Som et bandmedlem vil jeg kunne forandre outline farge og banner for å tilpasse stilen.
4. Som et bandmedlem vil jeg kunne forandre forsidebildet
5. Som et bandmedlem vil jeg kunne oppdatere «About us» teksten
6. Som et bandmedlem vil jeg kunne legge til/slette låter og album
7. Som et bandmedlem vil jeg kunne oppdatere YouTube videoen
8. Som et bandmedlem vil jeg kunne legge ut bilder i form av et bildegalleri/karusell
9. Som et Bandmedlem vil jeg kunne legge ut billettlenker slik at folk kan bestille billetter.
10. Som et bandmedlem vil jeg ha oversikt over trafikken på nettsiden gjennom et analyseringsverktøy

4. Domenemodell



5. Prototyper

5.1 Wireframe



Frostbite Nettside Systemdokumentasjon

Versjon 3.1

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
08/04/2022	<1.0>	Første versjon	Mahmod A. Mohamed Stian L. Rørvik Sander R. Fylkesnes
03/05/2022	<2.0>	Andre versjon	Mahmod A. Mohamed
05/05/2022	<3.0>	Tredje versjon	Mahmod A. Mohamed
07/05/2022	<3.1>	Oppdaterte font til Verdana, 10pt	Stian L. Rørvik

Contents

1. Introduksjon 4

2. Arkitektur 4

2.1 Nettleser 5

2.2 Web server 5

2.3 HTML, CSS & Javascript 5

2.4 Node Server 5

3. Prosjektstruktur 6

4. Databasemodell 8

5. Sikkerhet 10

6. Installasjon og kjøring 10

7. Dokumentasjon av kildekode 11

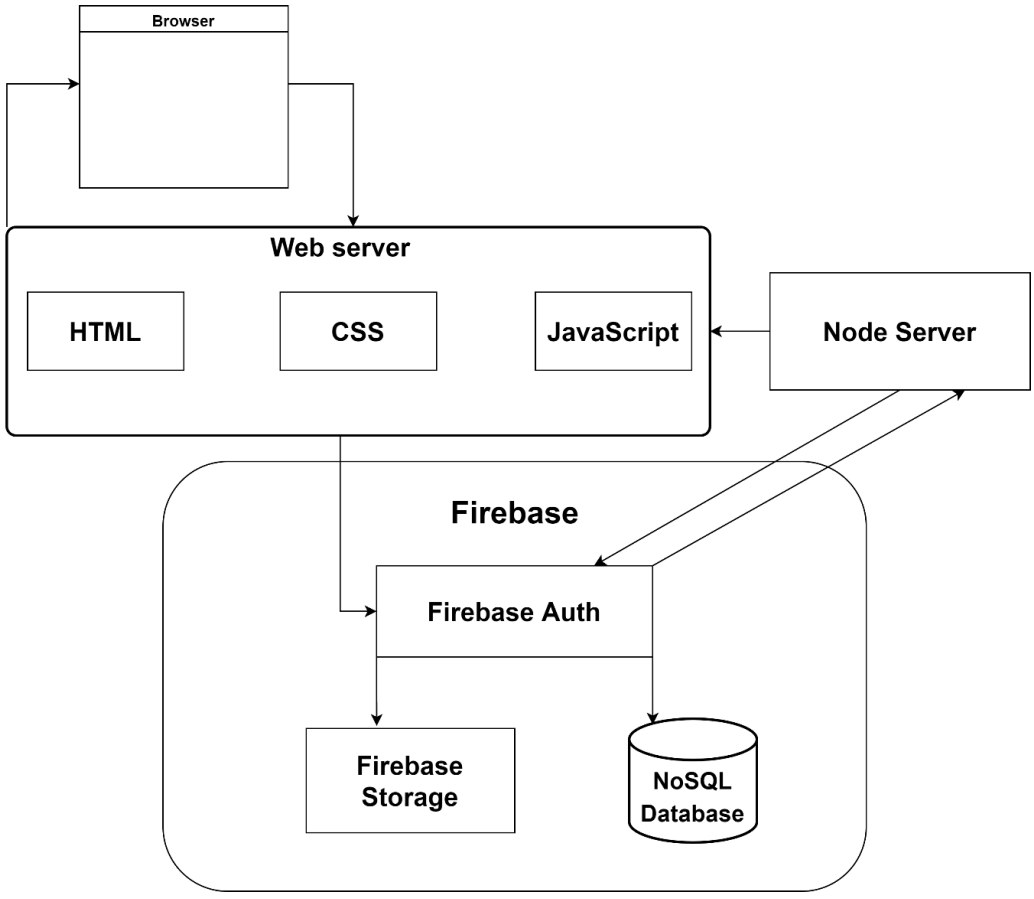
8. Kontinuerlig integrasjon og testing 11

9. Referanser 12

1. Introduksjon

Dette dokumentet skal gi en oversikt over de tekniske metodene vi brukte i utviklinga og beskrive hvordan de fungerer. Dokumentet vil gå gjennom arkitekturen til nettsida der de viktigste komponentene blir vist gjennom en arkitekturskisse, en beskrivelse av prosjektstrukturen og hvordan prosjektet er strukturert, databasemodellen til løsningen, hvordan løsningen er beskyttet mot uautorisert tilgang, hvordan du kan kjøre prosjektet vårt på egen maskin, dokumentasjon av kildekode og til slutt hvordan andre programmerere kan fortsette videre på prosjektet om klienten skulle ønske det.

2. Arkitektur



Figur 1: Dette er en arkitekturskisse som viser informasjonsflyten i systemet

2.1 Nettleser

Nettleseren (browser på engelsk) er en programvare som lar oss få tilgang til «World Wide Web» også kjent som weben. Den er en klient som sørger for å hente og vise nettsider fra weben ved å sende en http-forespørsel til en web server, få respons fra web serveren og vise responsen, som til vanlig er ei nettside, gjennom nettleseren. Brukeren av nettleseren taster inn en URL (Uniform Resource Locator) som kort sagt er en beskrivelse på hvor nettsiden er lokalisert i web serveren. En URL starter som regel med HTTP eller HTTPS som er protokollen for hvordan nettleseren spør etter informasjon fra web serveren (Mozilla, 2022). HTTPS er mest brukt på grunn av at overføring av data er kryptert og derfor sikrere mot angrip (mer om det under Sikkerhet).

2.2 Web server

En web server består av både hardware og software. Hardwaren er en datamaskin som inneholder filene som et nettsted består av slik som HTML, CSS, skript filer og softwaren. Softwaren kan bestå av flere komponenter, men er på et minimum en HTTP-server som brukeren kan interagere med ved å skrive inn domenenavnet til en nettstad som serveren hoster. En web server hoster filene på enten en statisk eller dynamisk server. Statisk web server lagrer og utveksler filer som ikke endrer innhold med klienten. En dynamisk web server består av statisk server i tillegg til andre software slik som applikasjonsserver og en database. Her skjer lagring og utveksling av filer dynamisk dvs. at innholdet blir oppdatert før den blir sendt til nettleseren (Mozilla, 2022).

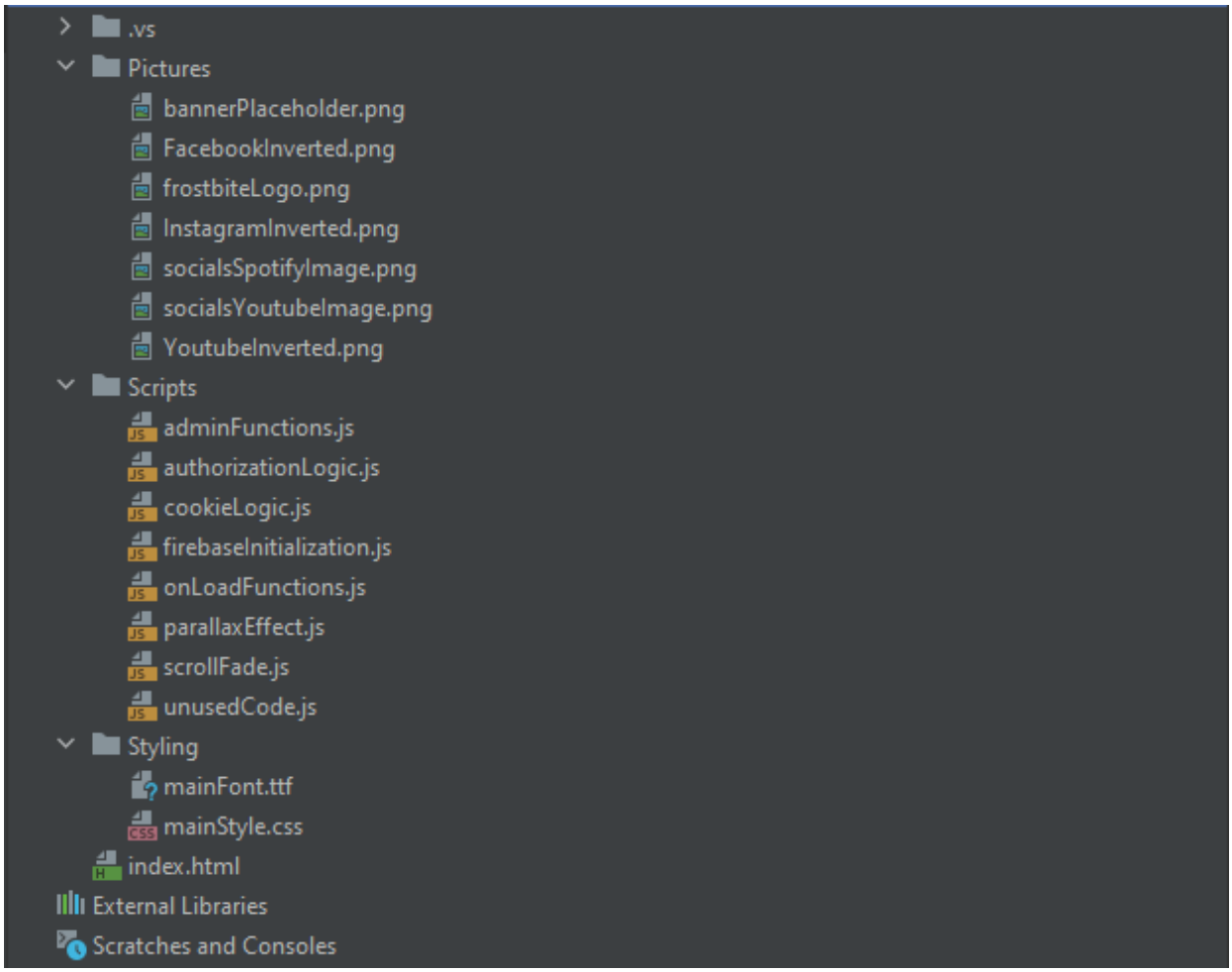
2.3 HTML, CSS & Javascript

Nettsidene som nettleseren viser, er i hovedsak skrevet i HTML (HyperText Markup Language). Oppgaven til HTML er å strukturere nettsidene ved å bruke HTML element og er skrevet i klartekst. CSS (Cascading Style Sheets) blir brukt til å designe en nettside. Den kan styre posisjonene til elementene i HTML dokumentet, fargene, størrelse av elementene, avstand mellom dem osv. Javascript er et programmeringsspråk mest brukt for å programmere dynamiske funksjoner for nettsider (Ubah, 2021).

2.4 Node Server

Node serveren er ett cross-plattform Runtime miljø som lar utviklere bygge server og nettverk applikasjoner med JavaScript. Dette prosjektet bruker node for async operasjoner for kommunikasjon med firebase databasen. Det gjør at vi slipper å bruke «Promises» og bruker heller «async/await» som har lettere syntaks (Node.js, u.d.).

3. Prosjektstruktur



Figur 2: Dette er prosjektstrukturen til koden vår tatt fra en IDE (integrert utviklingsmiljø)

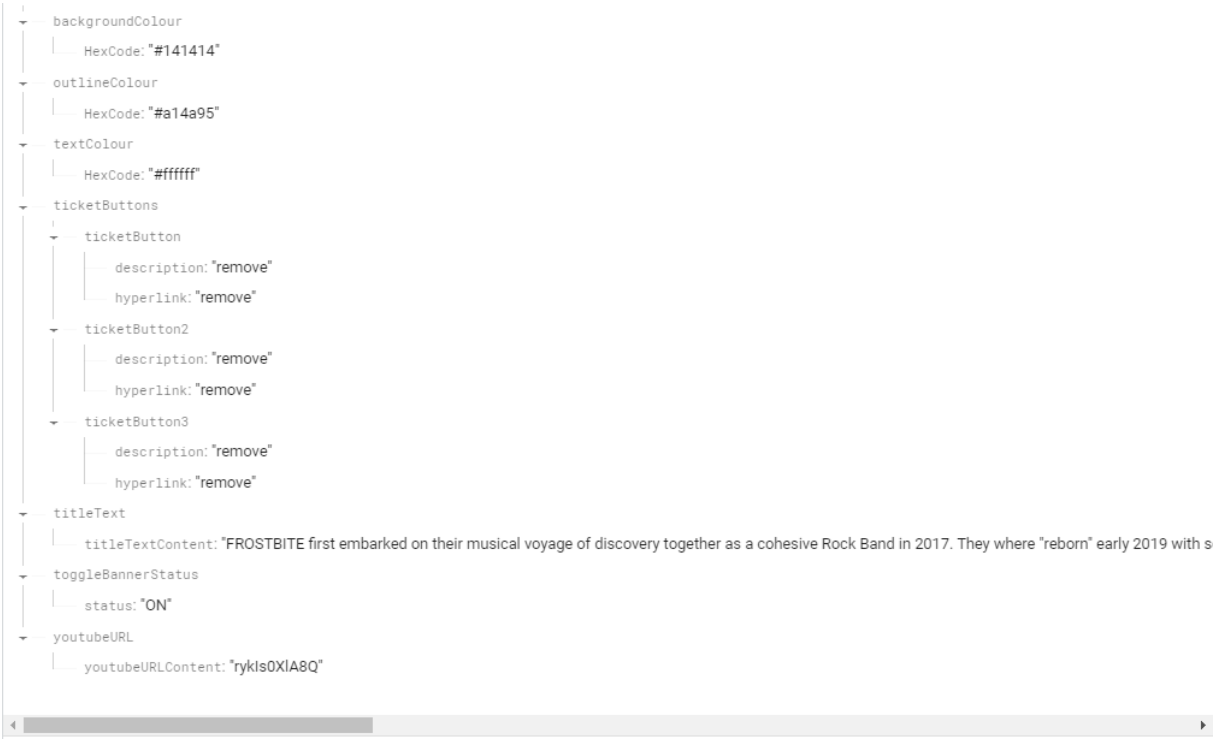
HTML kode	HTML - GitHub
CSS kode	CSS - GitHub
JavaScript kode	Scripts - GitHub

Prosjektet vårt er lasta opp på GitHub slik at gruppa vår kan jobbe med prosjektet ved hjelp av Gits versjonskontroll system. Det betyr at vi kan lagre prosjektet vårt i ulike versjoner for å ha bedre kontroll over endringene som blir gjort i prosjektet og fungerer også som en online back-up. Det lar oss også jobbe i lag på samme kode på avstand.

Prosjektfilen er delt inn tre mapper: Pictures, Styling og Scripts. Pictures-mappen inneholder ulike bilder vi har brukt til å designe nettsida. De fleste bildene slik som galleri bildene og banneret er lagret i Firebase Storage. Styling mappen inneholder alle CSS brukt til å style nettsiden og også fontene som blir brukt på nettsiden. Scripts-mappen inneholder alle JavaScript-filene som sørger at de dynamiske funksjonene på nettsiden fungerer, samt andre funksjoner slik som å hente informasjon fra databasen eller styre innloggingslogikk. Til slutt har vi HTML filen som er skjelettet til nettsiden. VS mappen, som er en Visual Studio mappe, er ikke en del av prosjektet, men har blitt med i prosjektet gjennom versjonsstyringen og er ikke relevant for prosjektet vårt.

Koden på prosjektet kan bli funnet på GitHub ved å enten klikke på linkene i tabellen over eller besøke https://github.com/Thelangostudent/Frostbite-Website

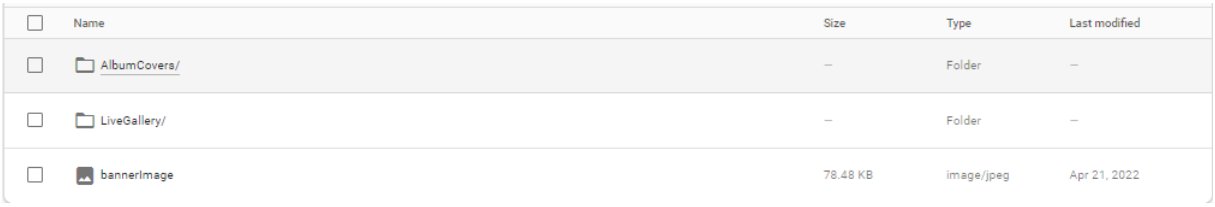
4. Databasemodell



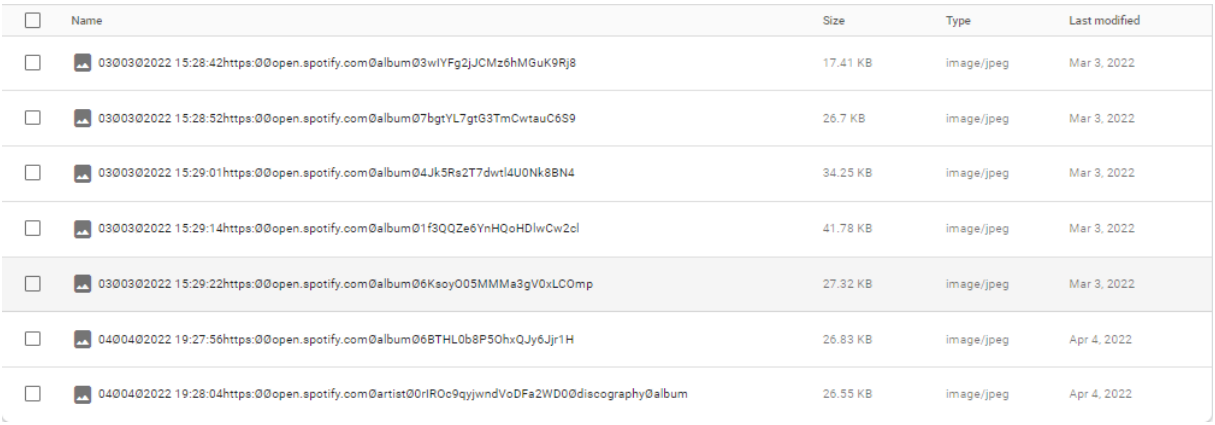
Figur 3: En oversikt over strukturen til databasen vår. Databasen vi bruker er Firebase sin Realtime Database

Prosjektet bruker en Realtime database som er av typen NoSQL database. NoSQL database er en ikke-relasjonell database som kan håndtere store mengder data som er ikke-strukturert og raskt endrende. Det betyr også at den ikke bruker en strukturert tabell slik som i relasjonelle databaser. Det var ingen spesiell grunn til at vi brukte NoSQL database framfor SQL database. Grunnen til at vi valgte NoSQL var fordi at vi allerede er kjent med Firebase fra et tidligere prosjekt, og siden Firebase ikke bruker SQL så ble det naturligvis NoSQL vi brukte (Schaefer, u.d.).

Dataene er strukturert som et JSON-tre som vist i figur 3. NoSQL databaser er delt i ulike typer. Realtime databasen til Firebase er av typen dokument database dvs. at data er representert som JSON objekt. Dette gjør at dataen kan bli strukturert fleksibelt og vi kan skalere data horisontalt. Horisontal skalering i dette tilfelle betyr at vi kan fordele data over flere noder i staden for å skalere opp en singel node. Siden dataen ikke er avhengig a hverandre slik som i relasjonell database kan vi fritt distribuere den utover flere noder (Google Firebase, 2022).



Figur 4 Her vises de overordnede kategoriene for de ulike mediefilene vi bruker på nettsiden



Figur 5: Dette er slik det ser ut når du klikker på AlbumCovers kategorien som er vist i Figur 4. Her ser dato og tid bildet ble lagt til Firebase Storage og URL-en til bildet

Firebase Cloud Storage er en mekanisme i firebase som lar deg lagre store filer slik som bilder og videoer. Firebase Storage lar deg laste ned/opp filer uansett nettverkskvalitet. Sikkerheten er integrert med Firebase Authentication slik at en kan identifisere brukere som bruker Storage. Du kan autorisere hvem som har tilgang til visse ressurser i Storage og du kna også validere data slik at for eksempel bare bilder med en viss størrelse har lov til å bli lastet opp på nettsida. Du kan lese mer om Firebase Authentication under Sikkerhet (Google Firebase, 2022).

5. Sikkerhet

Vi bruker Firebase Authentication (FA) for administrator systemet til nettsida vår. Det er flere måter å implementere FA på slik som passord autentisering, email link autentisering eller sende SMS til mobilnummer. Vi har valgt å bruke passord autentisering. Måten det fungerer på er at bruker din email og et passord, enten ditt eget passord for brukere som registrerer seg, eller slik som vi har gjort det, laget et ferdiglagd lagd passord for en bruker siden det er kun en administrator konto som blir brukt.

Passordet er beskyttet ved hjelp av hashing og salting. Scrypt (uttalt ess krypt) er den algoritmen som blir brukt av Firebase for å hashe passordet. Hashing er enkelt sagt en hash funksjon som tar in input slik som et passord og konverterer det til en streng med fast lengde. Scrypt er en hashing algoritme slik som SHA-1 og SHA-2, men er designet for å gjør det vanskelig for hackere å bruke «brute force» angrep ved å gjøre beregning av hashen kostbar både for tid og minne. Salting er å legge til tilfeldig data til for eksempel et passord før det blir hashet slik at en for eksempel kan hindre folk i å identifisere flere like passord (Google Firebase, u.d.).

Siden nettsida bruker HTTPS protokoll til å kommunisere med serveren er også utveksling av informasjon mellom server og klient kryptert (Mozilla, 2021). Det gjør at personvernet blir beskyttet også mot såkalt «man in the middle-angrep» der et eksempel på det kan være avlytting av informasjonen (Wikipedia, 2022).

6. Installasjon og kjøring

Du kan klone prosjektet fra GitHub lenken under Prosjektstruktur og åpne prosjektet i en editor. Du trenger ikke å laste ned noen eksterne ressurser. Nettsiden kan også bli sett direkte her: Frostbite (frostbiteband.com)

7. Dokumentasjon av kildekode

Dokumentasjon av kildekoden er tatt med i kildekoden, altså all kode er kommentert.

Eksempel på dokumentasjon av koden:

```
<!-------------------------------------------------------------- ADMIN MENU ---------------------------------------------------------------------------------------->
<!-- The admin container is a window with admin functions that is only visible to an admin that has successfully logged in with the relevant information,
     the buttons are only usable to an authorized admin, the authorization is checked whenever a button is pressed. A person without a valid admin session can not
     make any "set/update/delete" calls to the backend server even if they reverse engineered the website.
     All of the admin functions are locked behind our strict firebase security -->
<div id="adminContainer">

    <h3 id="adminTitle">Admin menu</h3>

    <!-- The changeBackgroundColour button activates the corresponding function in the adminFunctions javascript file when it is pressed. It allows an
         authorized admin to make changes to the background colour of the page by entering a new HEX colour code. The HEX code is saved in firebase and automatically applied
         to the page after it has been set, this is accomplished in the onLoadFunctions javascript file. -->
    <div class="alignInMiddle">
        <button id="changeBackgroundColour">Change Background Colour</button>
    </div>

    <!-- The changeOutlineColour button activates the corresponding function in the adminFunctions javascript file when it is pressed. It allows an
         authorized admin to make changes to the outline colour of the page by entering a new HEX colour code. The HEX code is saved in firebase and automatically applied
         to the page after it has been set, this is accomplished in the onLoadFunctions javascript file. -->
    <div class="alignInMiddle">
        <button id="changeOutlineColour">Change Outline/Button Colour</button>
    </div>

    <!-- The changeTextColour button activates the corresponding function in the adminFunctions javascript file when it is pressed. It allows an
         authorized admin to make changes to the test colour of the page by entering a new HEX colour code. The HEX code is saved in firebase and automatically applied
         to the page after it has been set, this is accomplished in the onLoadFunctions javascript file. -->
    <div class="alignInMiddle">
        <button id="changeTextColour">Change Text/Hover Colour</button>
    </div>

    <!-- The toggleBannerText button activates the corresponding function in the adminFunctions javascript file when it is pressed. It allows an
         authorized admin to toggle the large banner headline "Frostbite" on/off. This is useful in case the band wants to apply other information directly on the background
         image instead, it allows admins to turn the banner headline off if it comes in the way for such a case -->
    <div class="alignInMiddle">
        <button id="toggleBannerText">Overlay: ON</button>
    </div>
```

Figur 6: Eksempel på dokumentasjon av kildekode

8. Kontinuerlig integrasjon og testing

All kildekode blir lastet opp og slått sammen felles i GitHub, deretter blir koden automatisk deployed til en testversjon av nettsiden via «GitHub Pages», på denne nettsiden sjekker vi for eventuelle feil som har oppstått under den automatiske deployingen.

Om arbeidsgiveren vil ha videreutvikling på nettsida, så er koden tilgjengelig for offentligheten. Framtidige utviklere som skal jobbe videre med nettsida kan klone koden fra GitHub og laste opp ny kode filer på one.com og oppdatere siden.

Vi har også testen nettsiden kontinuerlig gjennom nettleseren ved å ta i bruk tilbakemeldinger og observasjoner fra arbeidsgiver (bandmedlemmene) annenhver uke, bandmedlemmene fikk også teste nettsiden selv mens vi tok notater i disse møtene.

9. Referanser

Google Firebase. (2022, April 22). *Cloud Storage for Firebase*. Hentet fra Firebase: https://firebase.google.com/docs/storage

Google Firebase. (2022, April 25). *Structure Your Database*. Hentet fra Firebase: https://firebase.google.com/docs/database/web/structure-data

Google Firebase. (u.d.). *Firebase Authentication Password Hashing*. Hentet fra Firebase Open Source: https://firebaseopensource.com/projects/firebase/scrypt/

Mozilla. (2021, Oktober 08). *MDN Web Docs*. Hentet fra HTTPS: https://developer.mozilla.org/en-US/docs/Glossary/https

Mozilla. (2022, Mars 05). *What is a web server?* Hentet fra MDN Web Docs: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server

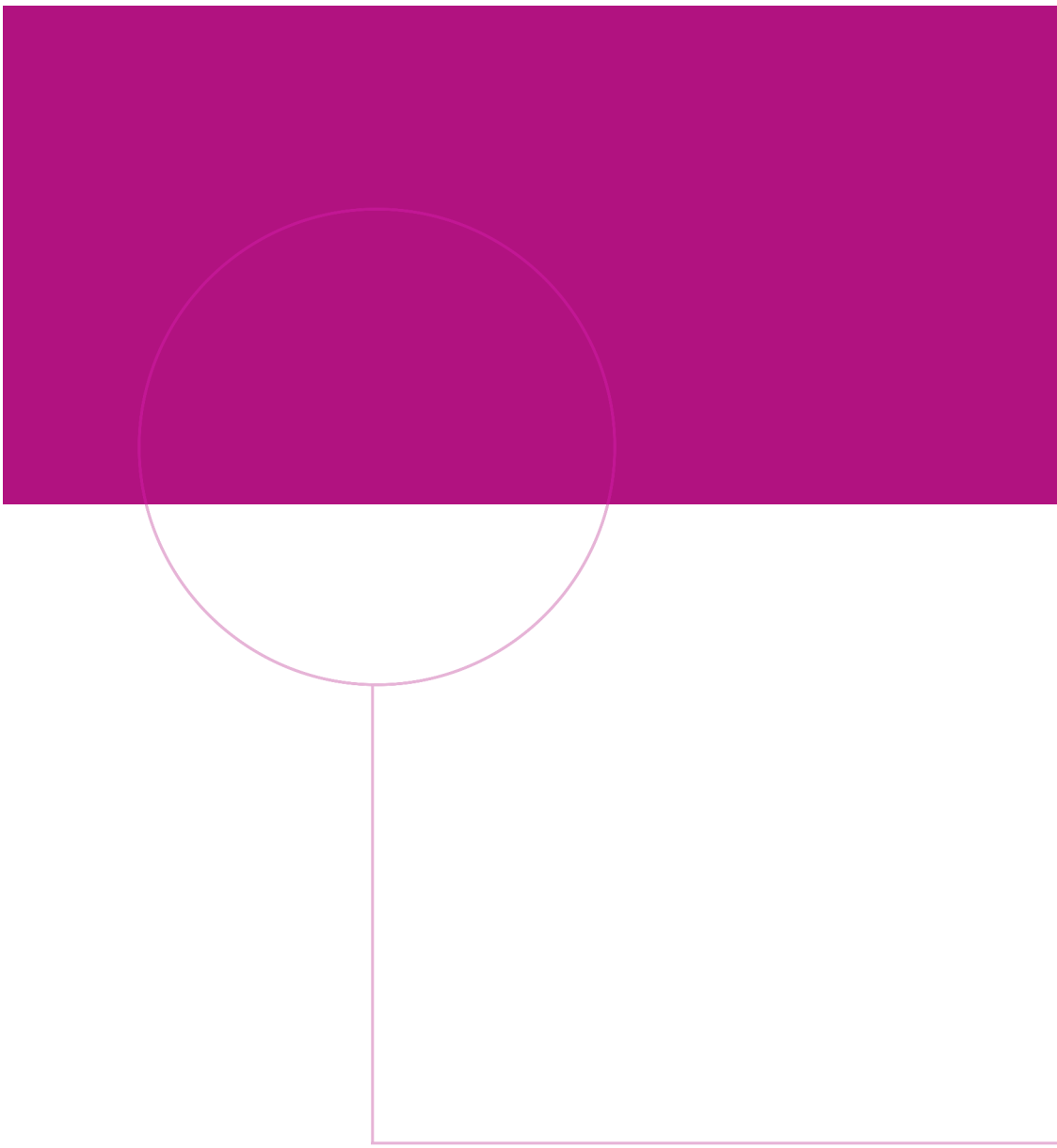
Mozilla. (2022, Februar 01). *What is the difference between webpage, website, web server, and search engine?* Hentet fra MDN Web Docs: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Pages_sites_servers_and_search_engines

Node.js. (u.d.). *Modern Asynchronous JavaScript with Async and Await*. Hentet fra Node JS: https://nodejs.dev/learn/modern-asynchronous-javascript-with-async-and-await

Schaefer, L. (u.d.). *What is NoSQL?* Hentet fra MongoDB: https://www.mongodb.com/nosql-explained

Ubah, K. (2021, August 10). *Learn Web Development Basics – HTML, CSS, and JavaScript Explained for Beginners*. Hentet fra freeCodeCamp: https://www.freecodecamp.org/news/html-css-and-javascript-explained-for-beginners/

Wikipedia. (2022, Mars 25). *Wikipedia*. Hentet fra Man-in-the-middle attack: https://en.wikipedia.org/wiki/Man-in-the-middle_attack



NTNU

Kunnskap for en bedre verden