Bacheloroppgave

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

Lasse Seivaag
Jørgen Steig

Utvikling av en Web-Plattform for Comparative Judgement

Bacheloroppgave i Dataingeniør
Veileder: Olav Skundberg

Mai 2021

NTNU
Norwegian University of
Science and Technology

Lasse Seivaag
Jørgen Steig

Utvikling av en Web-Plattform for Comparative Judgement

Bacheloroppgave i Dataingeniør
Veileder: Olav Skundberg
Mai 2021

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

NTNU
Kunnskap for en bedre verden

Forord

Dette prosjektet ble gjennomført av Lasse Seivaag og Jørgen Steig som en bacheloroppgave ved Institutt for datateknologi og informatikk (IDI) og Fakultet for informasjonsteknologi og elektroteknikk (IE) ved NTNU. Prosjektet foregikk i vårsemesteret 2021, i samarbeid med representanter for Institutt for lærerutdanning (ILU) ved NTNU.

Oppgaven ble valgt fordi vi ønsket å lage et produkt som kom til å bli tatt i bruk etter endt bachelor, og det virket som en interessant oppgave med mulighet til å velge mellom flere teknologier for å øke kompetansen vår. Muligheten til å bistå forskning ved NTNU var en stor motivasjon for oss, og gjorde at vi ønsket å utvikle et så godt produkt som mulig.

Vi vil takke NTNU Hjelp for å ha hjulpet oss med et problem med e-post konfigurasjon under utviklingen. Vi vil også takke venner, familie og oppgavestillers kolleger ved NTNU for støtte, oppmuntrende tilbakemeldinger og deltakelse i brukertester. Stor takk til oppgavestillere Tore Alexander Forbregd og Hermund André Torkildsen for muligheten til å gjennomføre dette prosjektet, hjelp under testing av produktet og for å ha utviklet analyse-modulen.

En spesielt stor takk til veileder Olav Skundberg for tett oppfølging, god kommunikasjon og tips til både dette prosjektet og framtidige prosjekter vi vil støte på i arbeidslivet.

Lasse Seivaag

Jørgen Steig

20.05.2021 Trondheim

Oppgavetekst

Arbeidstittel: Web-plattform for Adaptive Comparative Judgement

Hensikten med oppgaven:
Hensikten med oppgaven er å utvikle et system der forskere ved Institutt ved lærerutdanning NTNU, kan samle inn data for forskning og videre analyser.

Kort beskrivelse av oppgaveforslag:
Comparative judgement er en forskningsmetode for å gjennomføre studier der man ønsker å finne en (lineær) rangering av objekter (f. eks. elev-/studentbesvarelser og påstander). Deltakere (brukere av systemet som skal utvikles) blir presentert for to og to objekter der de må velge den ene over den andre. Se utfyllende kommentarer.

Oppgaven passer for (kryss av de(t) som passer og skriv evt. en kommentar til oss):	- Prosjektoppgave, Dataingeniør (Systemutvikling og/eller
Kan oppgavestiller stille arbeidsplass med nødvendig utstyr og programvare:	3D-grafikk) kan stille med sever.
Hvis ikke, hva kreves av maskin og programvare:	PC og utviklingsmiljø
Oppgaven passer best for, antall studenter:	- 2 - 3
Opplysninger om oppgavestiller	- 4
Er du fra bedrift/virksomhet eller er du student med en egendefinert/selvlaget oppgave? - Bacheloroppgave	- Bedrift/virksomhet

Navn på bedrift/organisasjon/student: Institutt for lærerutdanning

Addresse Gunnerus gate 1

Postnummer 7012

Poststed Trondheim

Navn på kontaktperson/veileder: Tore Forbregd

Telefon: 92446236

Epost: tore.a.forbregd@ntnu.no

Navn på kontaktperson 2/veileder 2: Hermund Torkildsen

Epost kontaktperson 2/veileder 2: hermund.a.torkildsen@ntnu.no

Utfyllende kommentarer til hva oppgaven gjelder:
Det er i all hovedsak to brukere av systemet som vi ønsker å utvikle: (1) deltakere/dommere og (2) forskere.

Plattformen skal gi forskere mulighet til å gjennomføre Comparative Judgement-studier. En studie består av en mengde objekter og et overordnet spørsmål; for eksempel objektene {Vår, Sommer, Høst, Vinter} og overordnet spørsmål: “Hvilken årstid foretrekker du?” Objekter som deltakerne/dommerne blir bedt om å sammenligne kan være av flere formater; for eksempel ren tekst, pdf-dokumenter, bilder eller word-dokumenter. Et minstekrav for en MVP er påstander i form av ren tekst. Forskere må

kunne opprette sånne tester/studier i systemet.

En deltaker/dommer blir invitert via lenke/kode som er sikret med et passord. Denne brukertypen skal da gjennomføre testen, og sammenligne to og to objekter. Den naive tilnærmingen tilsier at hver deltaker må gjennomføre n · (n-1) sammenligninger for et testsett med n objekter. Ved bruk av statistiske metoder kan man i gjennomsnitt klare seg
med langt færre sammenligninger. Systemet benytter en adaptiv utvelgelsesstrategi der det velger ut par til sammenlikning der modellen trenger mer informasjon (https://en.wikipedia.org/wiki/Adaptive_comparative_judgement).

Forskere kan be systemet om å utføre analyser som f.eks. reliabilitet og prognoser av ulike statistiske modeller. Forskere må kunne hente ut analyser og rådata (minstekrav for en MVP: rådata).

Produkteiere vil overlevere konkrete system-/funksjonskrav ved prosjektoppstart.

Teknologi:
Det er ønskelig, men ikke et krav, at man benytter skybaserte løsninger som f.eks. firebase + cloud functions som backend.
Det er selvsagt mulig å bygge egen backend, f.eks node.js + express + mongoDB/SQL.

Det er ønskelig at man benytter Vue.js/svelte.js som rammeverk for frontend.

Nyttige lenker:
* https://nodejs.org/en/
* https://www.mongodb.com/
* https://expressjs.com/
* https://vuejs.org/
* https://svelte.dev/

Utvalg av javascript bibliotek som kan benyttes i ACJ:
* https://www.npmjs.com/package/comparative-judgement
* https://github.com/creynders/cj

Dette er den originale oppgaveteksten, og vi har i samarbeid med produkteier utarbeidet visjonsdokument [1] og kravdokumentasjon [2] som baserer seg på oppgaveteksten og endrer eller tydeliggjør noen av kommentarene i denne.

Sammendrag

Prosjektet oppstod som et behov fra produkteier for å ha en egen plattform for å utføre comparative judgement studier. Behovet kom fra at eksisterende løsninger var problematiske med tanke på kontroll over data, støttede filtyper, mangel på administrativ kontroll og komplekst design som var til hinder for produkteier.

Utfordringen vår var å lage denne web-plattformen slik at produkteier kan oppnå effektmålene sine, som kan summeres til å være mindre avhengig av tredjepartsløsninger for forskning og å unngå problemene med eksisterende løsninger.

Planen har fra starten av vært at produkteier skal videreutvikle produktet og spesielt teste ut forskjellige algoritmer for utvalg av spørsmål, og vi har dermed prioritert funksjonalitet og design. Siden produktet skal videreutvikles har vi valgt teknologier som produkteier har erfaring med, slik som Svelte på frontend, og NodeJS og MongoDB på backend.

Prosjektet har blitt utført med fokus på god kommunikasjon mellom produkteier og utviklere med hensikt å sørge for at produktet blir som ønsket. Det er også utført brukertester for å få tilbakemelding om navigerbarhet, brukervennlighet, hastighet og funksjonalitet. Dette har vært nyttig for å både teste og forbedre produktet, samt å bestemme hva som burde prioriteres videre i utviklingen.

Resultatet av dette prosjektet er en fungerende plattform for å utføre comparative judgement studier som kan utvides eller endres etter ønske og behov, med et simplistisk design, slik som produkteier ønsket. På bakgrunn av dette konkluderer vi med at utfordringen er løst.

Innholdsfortegnelse

Figurer og illustrasjoner

Kapittel 1: Introduksjon og relevans

Prosjektet oppstod grunnet et behov for en egen plattform til å utføre comparative judgement-studier. Det finnes eksisterende plattformer som anvender comparative judgement, men disse har diverse begrensninger som gjør dem mindre egnet, eller tungvinte å bruke for studiene produkteier utfører. De eksisterende plattformene har blant annet begrensninger når det kommer til kontroll over resulterende data, støtte for forskjellige filtyper eller mangel på administrative rettigheter som kreves av produkteier. Som resultat av dialog med produkteier under utredning av visjonsdokumentet [1] har vi utarbeidet en rekke effektmål som beskriver hva produkteier ønsker å oppnå med plattformen.

1.1 Effektmål

- Å være mindre avhengig av tredjepartsløsninger for forskning.
- Å ha en plattform som er enkel å bruke, blant annet ved å ha et simplistisk design uten mer funksjonalitet enn det som kreves for typen studier som er relevante.
- Å ha muligheten til å utvide plattformen med ny funksjonalitet eller endre eksisterende funksjonalitet etter ønske og behov.
- Å ha kontroll over egen forskningsdata og kunne sikre overensstemmelse med gjeldende lovverk.
- Å ha full administrativ kontroll over hele systemet. Dette er spesielt viktig når masterstudenter skal bruke plattformen for å kunne sikre innsyn og integritet av data.

1.2 Utfordringer

Hovedutfordringen vår var å produsere en fungerende web-plattform for å gjennomføre comparative judgement-studier som ikke lider av de samme problemene som eksisterende løsninger, og dermed oppfyller effektmålene. For å gjøre det lettere å løse denne utfordringen tok vi hensyn til tre under-utfordringer.

- Hvordan tar vi hensyn til at systemet skal videreutvikles?
- Hvilken infrastruktur trenger vi, hvilken teknologi og programvare kan vi benytte?
- Hvordan minimerer vi friksjonen brukere av plattformen opplever slik at vi kan sørge for en god brukeropplevelse?

Aspekter som skaper friksjon for brukere er hastighet, brukergrensesnitt og tilbakemelding på input fra brukeren.

1.3 Rapportens struktur

I kapittel 2 tar vi for oss teori vi har benyttet i prosjektet. Dette inkluderer teori for utviklingsmetode, teknologi og comparative judgement. Videre har vi konkretisert hvilke teknologier og metoder vi har benyttet i prosjektet i kapittel 3, og forklart hvorfor vi har valgt nettopp disse. Med denne bakgrunnskunnskapen går vi over resultatene vi har produsert i kapittel 4, som diskuteres videre i kapittel 5, med hensyn på planen og målene for prosjektet. I kapittel 6 trekker vi konklusjoner basert på de forrige kapitlene, og gir forslag til videre arbeid for å forbedre plattformen.

1.4 Ordliste, akronymer og forkortelser

API - Application Programming Interface. Et grensesnitt som anvendes for å kommunisere mellom forskjellige programmer.
Boilerplate kode - Kode som repeteres flere ganger med liten variasjon. Anses generelt som en negativ ting, grunnet at det kaster bort tid og ressurser til de som skriver og leser koden.
Brute force - Å teste alle mulige kombinasjoner til man finner resultatet man ønsker.
Cluster - Flere sammenkoblede databaseinstanser som inneholder forskjellig data. Brukes for å skalere lesing, skriving og kapasitet for databasen.
Comparative judgement - En vurderingsmetodikk som baserer seg på å sammenligne par med alternativer i stedet for å vurdere hvert alternativ individuelt.
Container - Et sammenpakket program-miljø som inneholder alt som trengs for å kjøre en applikasjon, foruten de delene operativsystemet tilbyr.
DOM - Document Object Model. En objekt-orientert representasjon av strukturen og innholdet til en nettside, som kan manipuleres med kode.
Dommer - En person som svarer på comparative judgement undersøkelser.
IT - Informasjonsteknologi.
MVP - Minimum Viable Product.
OS - Operativsystem.
Overhead - Ekstra ressurser som kreves for å utføre en oppgave.
Plattform/System/Produkt - Alle deler av programvaren vi utvikler.
Rainbow table - En tabell som kobler sammen input og output av en hash-funksjon, slik at man kan søke på et output fra hash-funksjonen og motta korresponderende input.
Replica set - Flere sammenkoblede databaseinstanser som inneholder samme data. Brukes for økt redundans, reliabilitet og lese-kapasitet.

Sanitering - Å endre input slik at det ikke kan tolkes som valid programlogikk.
Tilstandsløs - Et begrep som beskriver et system hvor tilstand ikke endres basert på tidligere forespørsler.
Transaksjon - En prosess med flere steg, som ofte avhenger av hverandre.
Virtuell maskin - Et program som emulerer en datamaskin. Denne datamaskinen kan anvende den faktiske datamaskinen sine ressurser, men for eksempel kjøre et forskjellig OS.

Kapittel 2: Teori

2.1 Comparative judgement

Comparative judgement er en vurderingsmetodikk som baserer seg på en serie av sammenligninger mellom par av elementer, i stedet for å vurdere hvert element individuelt på en absolutt skala. I en comparative judgement-studie blir en dommer presentert med et spørsmål og to svaralternativer. Dommeren skal dermed velge hvilken av disse de foretrekker med hensyn på spørsmålet. Denne prosessen skjer i flere iterasjoner for hver dommer.

Basert på dommernes svar kan det genereres en rangert liste over hvilke alternativ som gjorde det best, som kan brukes som grunnlag for forskning eller vurdering. Antall svaralternativer kan være vilkårlig stort, men det vil kreve flere respondenter for å sikre at rangeringen er troverdig.

Motivasjonen for å rangere alternativer på denne måten i stedet for å vurdere hver besvarelse for seg selv er at man kan sikre en høyere grad av validitet i svarene. Vi kan eksempelvis forestille oss en situasjon der en dommer ønsker å rangere hvor tunge et sett med vekter er. Dommeren kan løse dette ved å løfte hver vekt individuelt og anslå dens tyngde, og deretter rangere dem. Problemet med denne løsningen er at informasjonen vi mottar om vektens tyngde ikke er nøyaktig, som kan føre til at vi rangerer vektene feil.
Dersom dommeren i stedet sammenligner par med vekter og noterer ned hvilken som er tyngst, er det rimelig å anta at dommerens svar har en høyere grad av nøyaktighet.
Blant annet Jones og Alock argumenterte for dette i 2012: “people are far more reliable when comparing one thing with another than when making absolute judgements” [3, s. 64]. Den økte nøyaktigheten kommer på bekostning av at vi bare kan si noe om svaralternativene i relasjon til de andre svaralternativene. Vi har med andre ord mistet all sans for hvor tunge

vektene faktisk er, siden de bare sammenlignes med hverandre. Informasjonen vi mottar er svakere, men mer nøyaktig [4, s. 77].

I oppgaveteksten beskrives vurderingsmetodikken som "Adaptive Comparative Judgement". Dette er en spesifikk versjon av comparative judgement-studier der svaralternativene som dommeren skal velge mellom blir valgt ut basert på tidligere svar. Modellen tilpasser seg med andre ord basert på hvilken informasjon den mangler for å øke reliabiliteten og minimere antall svar som kreves. Det finnes flere forskjellige måter å implementere dette på, men et av de enkleste konseptene som kan tas hensyn til er transitivitet. Dersom en dommer mener at A er bedre enn B, og at B er bedre enn C, er det lite hensikt i å spørre dommeren hvilken den foretrekker av A og C. Vi vet allerede at dommeren vil foretrekke A, og kan dermed spare dommeren fra å måtte svare på unødvendige eller "dumme" spørsmål.

2.2 Utviklingsmetode

En utviklingsmetode i systemutvikling er et sett med regler, normer og tenkemåter som er ment til å veilede utviklingen av et IT-system. Dette kan innebære alt fra hvordan man planlegger og utfører prosjekter, hvilken dokumentasjon som burde skrives, hvilke steg og rutiner som eksisterer og til hvilket tidspunkt disse skal utføres. Rutinene i disse metodene har som hensikt å sikre oversikt og effektivitet i prosjektet. I dag anvendes i hovedsak utviklingsmetoder som er basert på det agile manifesto [5]. Disse har i tillegg et spesifikt fokus på å sikre at prosjektet er fleksibelt nok til å tåle endringer i kravspesifikasjoner underveis i utviklingen. Dette oppnås ved å gjennomføre prosjektet på en iterativ og inkrementell måte, der det jevnlig kommuniseres med produkteier underveis.

2.3 REST-API

Vi har utviklet en web-plattform som krever at en bruker kan utveksle informasjon med systemet. En enkel og ofte brukt arkitektur for dette formålet er et Representational State Transfer(REST) API. RedHat [6] beskriver et REST-API som et API som følger spesielle kriterier. Noen av disse kriteriene er:

- En klient-tjener arkitektur med ressurser (noe konkret som kan bli gitt en identifikator)
- Kommunikasjon mellom klient og tjener er tilstandsløs og informasjon sendes på en bestemt form som for eksempel JSON over HTTP.

- En logisk oppbygning av URI-er med metoder som GET, PUT, POST og DELETE, slik at klienten vet hvordan man kan hente, manipulere og bruke ressursene.

2.4 Database

For å kunne ta vare på data som blir generert som følge av bruken av systemet må vi ha en plass å lagre dette. Det er også logisk at dataene lagres i et system og med en struktur slik at man kan gjøre spørringer for å finne igjen, manipulere, slå sammen eller telle denne dataen. Det er nettopp dette et databasesystem tilbyr. Det finnes mange forskjellige typer databasesystemer, men to av de mest brukte er relasjonsdatabaser og dokumentdatabaser [7].

Relasjonsdatabaser baserer seg på tabeller med kolonner (type data) og rader (et innslag av data), og det er ofte et mål å splitte tabeller slik at man kan relatere en tabell til flere andre. Som et eksempel på dette kan vi se for oss en tabell med "personer", og en tabell med "bedrifter". Både en person og en bedrift har ofte en adresse og et telefonnummer. Det er da hensiktsmessig å ha adresse og telefonnummer som egne tabeller, som kan relateres til en person, og/eller en bedrift. Denne prosessen kalles "*normalisering*" [8] og benyttes blant annet for å redusere mengden med duplisert informasjon og å sikre integriteten av data. Relasjonsdatabaser krever også at alle innslag i en spesifikk tabell har de samme kolonnene.

Dokumentdatabaser tar en annen tilnærming, der man samler dokumenter (innslag) i en collection, som kan sammenlignes med en tabell. Det stilles ingen krav til at disse dokumentene er strukturert på samme måte, eller inneholder samme type data. Dette gir mer fleksibilitet enn en relasjonsdatabase, men har andre utfordringer, som for eksempel at man ikke kan garantere at et gitt dokument har feltene man forventer. I stedet for å normalisere dokumentdatabaser er det vanlig å inkludere all relatert data i dokumentet, altså at et dokument kan inneholde et nytt dokument. Dette sørger for at man slipper flere kall til databasen for å hente informasjonen man ønsker, men det kommer på bekostning av mulig duplisert data.

2.5 Virtualisering og Containerisering

Virtualisering og containerisering er metoder og teknologier som brukes for å lage egne miljø for applikasjoner. Dette er med på å forbedre sikkerheten og øke portabiliteten til applikasjonen.

Microsoft [9] beskriver en rekke likheter og ulikheter mellom disse teknologiene.
For virtualisering har man full isolasjon mellom host-maskinen og den virtuelle maskinen. Den virtuelle maskinen kjører et eget operativsystem med egen kernel, som kan være den samme som host-maskinen, eller noe helt annet. Virtualisering er ressurskrevende da man kjører et helt OS på toppen av et annet OS, som er isolert fra hverandre. Hver enkelte virtuelle maskin må derfor oppdateres hver for seg. Containere har derimot lavere overhead fordi de kan benytte seg av host-maskinens operativsystem, og inkluderer bare programvaren som kreves for å kjøre applikasjonen den inneholder, slik som programvarebibliotek eller annen data. I stedet for å oppdatere hver spesifikke container oppdaterer du byggeblokkene (image) disse baserer seg på, slik at alle containerne som bruker samme byggeblokker får samme oppdatering. Av sikkerhetsmessige årsaker benyttes det et tynt lag med virtualisering/isolering av for eksempel brukere, filsystem og nettverk også i containere.

Ut ifra dette og egen erfaring kan vi si at virtualisering krever mer ressurser både i bruk og vedlikehold i forhold til containere. Samtidig har virtualisering bedre sikkerhet grunnet at det benyttes full isolasjon, mens containere bare har delvis isolasjon.

2.6 Sikkerhet

2.6.1 HTTPS

Store Norske Leksikon [10] beskriver Hyper Text Transport Protocol Secure (HTTPS) som en protokoll som sørger for sikker overføring av data via HTTP på applikasjonslaget og Transport Layer Security (TLS) eller Secure Sockets Layer (SSL) på transportlaget. TLS er etterkommeren til SSL, og brukes dermed helst i dag. Protokollen kan bekrefte identiteten til tjeneren man kobler seg opp mot via sikkerhetssertifikater, og krypterer data som sendes til og fra tjeneren slik at man forhindrer at uønskede tredjeparter kan se hva som overføres.

2.6.2 Hashing

En kryptografisk hash-funksjon er en enveis funksjon som basert på et gitt input returnerer en hash av en bestemt lengde. Samme input vil generere samme output hver gang, og en liten endring i input burde føre til en stor endring i output. Dette kan brukes for å unngå å lagre passord i klartekst. I stedet for å lagre passord i klartekst kan man lagre resultatet man får fra

hash-funksjonen med passordet som input. Ved innlogging sammenligner man den lagrede hashen med hashen av passordet brukeren prøver å benytte seg av. Det at hash funksjoner er "enveis" betyr at man ikke direkte kan finne hvilken input verdi som ga en gitt output verdi. For å finne input verdien (brukerens faktiske passord) må man gjette seg fram og kjøre hash-funksjonen med korrekte innstillinger fram til man får samme resultat som hashen.

Det finnes flere forskjellige hash-funksjoner som kan anvendes til lagring av passord, men det finnes et lite utvalg som blir brukt betydelig mer enn andre. Disse er av stor interesse for angripere, og det har dermed blitt utviklet såkalte "*rainbow table*s" for de mest populære hash funksjonene. Rainbow tables er store tabeller som korrelerer input i hash funksjonen til output. Disse kan dekke store mengder med vanlige passord, og gjør det mulig for en angriper å bruke hash funksjonen sitt output (det som ligger i databasen) og finne ut hash funksjonen sitt input (brukerens faktiske passord) dersom passordet er svakt. For å motvirke dette anvendes ofte et "salt" i kombinasjon med hashing. Et salt er ikke mer enn en lang pseudo-tilfeldig generert (helst ved bruk av en **C**ryptographically **S**ecure **P**seudo**r**andom **N**umber **G**enerator) tekststreng som legges til på slutten av passordet før det sendes inn i hash-funksjonen. Dette saltet lagres også i databasen, og tvinger angripere til å brute-force alle kombinasjoner på nytt, siden den resulterende hashen vil være basert på det unike saltet.

2.6.3 Cross Site Scripting, injection og Re/DoS

Cross Site Scripting (XSS) er et angrep hvor angriperen klarer å manipulere en annen brukers visning eller opplevelse av en nettside, og kan dermed lokke brukeren til å gi fra seg informasjon. Dette angrepet skjer vanligvis via input-felter som ikke blir validert, og at brukergenerert data blir brukt som HTML og/eller Javascript på frontend. For å unngå denne typen angrep er det viktig å validere og sanitere input fra klienter på tjenersiden, og behandle data man mottar fra tjeneren på korrekt måte på klientsiden [11].

Injection er et angrep hvor en form for input blir misbrukt slik at inputen blir en del av programlogikken. "Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. Injection flaws occur when an attacker can send hostile data to an interpreter." [12].
For å unngå denne typen angrep foreslår OWASP å skille data fra kommandoer og spørringer, bruk av trygge API-er, eller anvendelse av Object Relational Mapping.

Denial of Service (DoS) er et angrep som har som mål å gjøre en tjeneste utilgjengelig for vanlige brukere, typisk gjort ved å få den til å krasje, eller overbelaste tjenesten ved å øke ressursbruken. ReDoS er en form for DoS angrep som bruker regular expression (regex) som angrepsvektor. Regex, og spesielt extended regex har noen metoder som utføres i polynomisk tid. Disse kan misbrukes til å forbruke mye CPU-tid hos tjeneren, dersom det ikke er noen sperre på lengden av input. Det er også viktig å sanitere brukerinput som brukes i regex, slik at en angriper ikke kan endre regex-uttrykket etter egen vilje.

Kapittel 3: Valg av teknologi og metode

I dette kapittelet går vi over hvilke teknologier og metoder vi anvender, hvorfor vi har valgt nettopp disse, og erfaringen vår med dem. For mer utfyllende informasjon henviser vi til systemdokumentasjonen [13].

3.1 Database

I oppstartsmøte med produkteier [14, s. 7] ble vi enige om å bruke MongoDB som databasesystem. Valget stod i mellom MySQL og MongoDB basert på oppgaveteksten og viten om at produkteier skal videreutvikle produktet og har erfaring med disse to databasene. Tidligere i studiet har vi bare anvendt MySQL, så vi anså dette som en mulighet til å utvide vår kompetanse som programmerere, og valgte derfor MongoDB.

MongoDB er en dokument-basert NoSQL database som lagrer data i BSON (Binary JSON) format. MongoDB har noen viktige begrensninger som må tas i betraktning gjennom utviklingen av produktet: Maks størrelse på et enkelt dokument er 16MB inkludert embedded dokumenter, og transaksjoner støttes ikke uten replica set og/eller clustering.

3.2 Backend, REST-API

I oppgaveteksten nevnes det at det er ønskelig å ta i bruk skyløsninger som firebase og cloud-functions. Gjennom research og oppstartsmøte med produkteiere [14] kom vi fram til at å bruke skytjenester ikke nødvendigvis løser problemer med GDPR, og kan føre til uforutsette kostnader. På bakgrunn av dette bestemte vi oss for å utvikle en egen backend-løsning med Node.js og express, slik det ble foreslått som alternativ i

oppgaveteksten. Denne backend løsningen er et REST-API som fungerer som knutepunkt for frontend og resten av tjenestene på backend.

Vi har erfaring med både node og express fra før via fag ved NTNU og egne prosjekter. Vi benytter mange biblioteker/middlewares i løsningen vår: apidoc, axios, body-parser, cookie-parser, cors, dotenv, escape-string-regexp, express, express-fileupload, jsonwebtoken, mongo-escape, mongoose, nodemailer, sjcl og crypto. Disse har blitt valgt basert på behov, tidligere erfaring, popularitet basert på statistikk fra https://npmjs.com og forslag fra for eksempel stackoverflow. For mer informasjon om bruk og hensikt med disse bibliotekene viser vi til systemdokumentasjon [13].

3.3 Frontend rammeverk

Som frontend rammeverk hadde produkteier et ønske om at enten Vue.js eller Svelte.js ble benyttet. Disse er begge komponent-baserte rammeverk for frontend-utvikling. Det vil si at man abstraherer kode, både Javascript, HTML og CSS, inn i logiske komponenter, som kan gjenbrukes flere steder i plattformen. Valget stod mellom disse to grunnet at produkteier hadde et ønske om å videreutvikle og vedlikeholde plattformen etter endt bachelor, og det var disse de hadde erfaring med. På oppstartsmøtet bestemte vi oss for å gjøre utviklingen i Svelte, basert på produkteiers ønske og for å øke vår egen kompetanse i en voksende teknologi.

Til tross for at Svelte ikke er like mye brukt som Vue [15], så er det en kommende teknologi som skiller seg ut fra de tre store; React, Angular og Vue. I motsetning til resten er Svelte en kompilator, som kompilerer komponentene sammen til en stor optimalisert Javascript fil. Dette betyr blant annet at Svelte ikke anvender noen form for virtuelt DOM, som bidrar til mindre overhead. Rich Harris, skaperen av Svelte, demonstrerte effektene av dette i en presentasjon ved *You Gotta Love Frontend* 2019 [16].

Siden Svelte-komponenter er av samme struktur som et vanlig HTML-dokument, og ikke er avhengig av egne funksjoner for å oppdatere state (som med hooks i React) produserer det mindre “boilerplate”-kode, som kan gjøre det enklere å lese og skrive.

3.4 Docker

Docker er en full software pakke for å lage og kjøre containere. I møte 27.01.2021 ytret produkteier ønske om å kjøre plattformen i Docker med bakgrunn i sikkerhet og erfaring. Vi er enige i at dette er et logisk valg, og det kan forenkle vedlikehold og modularisering av plattformen. Produkteier foreslo å bruke docker-compose for å forenkle oppsettet, da dette tillegget automatiserer byggeprosessen og sammenkoblingen av de forskjellige tjenestene man setter opp.

Vi har litt erfaring med docker fra faget TDAT2004 ved NTNU, men ingen erfaring med å kjøre flere avhengige tjenester ved siden av hverandre med Docker. Det har derfor gått med en god del timer til research på dette, samt oppsett og testing på en tjener produkteier skal bruke som produksjonsmiljø.

3.5 Utviklingsmetode

Prosjektet har blitt gjennomført med det agile manifesto [5] som hovedinspirasjon. Vi har spesielt hatt fokus på god kommunikasjonsflyt, å være åpne for endringer og å involvere produkteier underveis i utviklingen, blant annet i form av demoer. Dette gjøres for å sikre at prosjektet ender opp så nær produkteiers visjon som mulig, samt for å fasilitere og håndtere mulige endringer i produkteiers visjon.

Vi har ikke fulgt et spesifikt utviklingsrammeverk, men vi har lånt noen idéer fra kanban og SCRUM. For eksempel en digital oppgave-tavle på https://trello.com/ hvor vi har "klistrelapper" med gjøremål og bugs som blir plassert under kategorier som "planlagt", "ferdig" og "til demo". Vi har også hatt møte med produkteier med to-ukers mellomrom, hvor vi har vist fram det vi har utviklet så langt for å få tilbakemelding og planlegge videre arbeid og endringer av visjon og krav. Sistnevnte rutine kan sammenlignes med sprinter i SCRUM. I tillegg til disse møtene har vi hatt lav terskel for å stille spørsmål mellom møtene på Microsoft Teams. Med dette har vi en iterativ og inkrementell metode der vi følger en rutine for å justere det som finnes, og bygger videre på forrige versjon av produktet.

Vi valgte å gjøre det på denne måten da vi anså eksisterende rammeverk som SCRUM som lite effektive for et team på to personer. Spesifikt SCRUM har enkelte rutiner, for eksempel

stand-up møte og sprint backlog, som mister mye av sin verdi når teamet bare består av to personer som allikevel kommuniserer konstant. I større team er slik strukturert kommunikasjon essensielt for at medlemmene i teamet skal ha god oversikt over hva som gjøres. Grunnet at vi kommuniserte ved hjelp av digitale møteverktøy mens vi arbeidet, anså vi eksisterende krav til dokumentasjon som vel dekkende og tidskrevende. Tiden brukt på å følge en for tung utviklingsmetode ville gått på bekostning av produktet vi utviklet og funksjonaliteten produkteier ønsket. Grunnleggerne av SCRUM, Ken Schwaber og Jeff Sutherland [17, s. 7], påstår selv at tre til ni er optimal størrelse på et SCRUM-team, og at team på færre enn tre resulterer i mindre produktivitetsgevinst.

3.6 Arbeids- og rollefordeling

Teamet vårt bestod av to studenter. Vi prøvde å fordele arbeidsmengde og ansvar så likt som mulig. Eksempelvis rullerte vi på hvem som var møteleder og referent i møter med både produkteier og veileder. Vi hadde med unntak av at Lasse styrte oppsett og drift av server ingen andre bestemte roller. Når det kom til hvem som skulle jobbe med spesifikke aspekter av systemet, valgte vi naturlig oppgaver vi følte vi kunne løse, eller ønsket å prøve å løse. Lasse jobbet mest med backend, og Jørgen mest med frontend. Med denne fordelingen hadde vi som mål å både oppnå et godt resultat ved å utnytte det vi er gode på, men også utfordre oss selv og øke kompetansen vår.

Kapittel 4: Resultater

Vi har produsert en komplett web-plattform bestående av fire moduler:

- Frontend, som er en nettside.
- Backend, som er et REST-API.
- En database hvor vi kan lagre data.
- En analyse-modul som ble skrevet av produkteier.

Systemet er funksjonelt klart til å brukes ved overleveringstidspunktet, men det kan definitivt forbedres. Videre i dette kapittelet går vi over noen designvalg, måloppnåelse basert på brukernes behov, planen for prosjektet og andre aktiviteter som har hjulpet oss med utviklingen av plattformen.

4.1 Vitenskapelige resultater

4.1.1 Brukertester

Det ble gjennomført brukertester fra 12. til 16. april for å samle ideer om hva som kunne forbedres. Vi fikk 14 respondenter til å prøve systemet som dommer, hvorav ni svarte på en Google Forms-undersøkelse. Det ble totalt syv stykker som testet systemet som forskere. En mer grundig analyse av svarene på brukertestene, samt hvilke forbedringspunkter som ble prioritert finnes i vedlegget analyse av brukertester [18].

Brukertesten for dommere besto av å trykke på en link til en undersøkelse, eller eventuelt taste inn en PIN-kode på forsida. Deretter ble de bedt om å lese informasjonssiden til undersøkelsen, og gjennomføre 10 sammenligninger. Brukerne sammenlignet forskjellige aspekter av hva som er viktigst med en nettside, og vi fikk på den måten en form for dobbel tilbakemelding på hva vi burde prioritere fremover. Denne brukertesten omfattet bare en liten del av systemet, men var allikevel viktig i og med at det er denne delen av systemet som vil bli brukt av flest og er prioritet for produkteier.

Sammenligningene som dommerne utførte i brukertesten resulterte i en lineær rangering av hva de anså som viktigst med en nettside [18].

OPTION	INFIT	OUTFIT	PROPSCORE	THETA	WINS	LOSSES	TOTAL COMPARISONS
Speed/Low loading times	0.9774	0.8634	0.7895	1.7222	15	4	19
A well designed navigation bar or menu	0.8592	0.6199	0.7619	1.0237	16	5	21
Good user feedback (error messages, loading indicators, change look of an element on hover or on click etc.)	0.9029	1.0815	0.6667	0.8964	10	5	15
Icons that describe actions or links instead of text	1.1289	0.9524	0.6667	0.7292	14	7	21
A responsive UI (changing layout/sizes for mobile f.ex)	0.9686	0.8444	0.625	0.6996	15	9	24
Universal Design	0.7181	0.5674	0.6087	0.3903	14	9	23
Icons that describe actions or links in addition to text	1.3021	1.4476	0.5294	0.3438	9	8	17
A clean and modern design	1.2677	1.3388	0.4783	0.1199	11	12	23
Option to toggle between dark/light theme	1.1569	1.1828	0.48	-0.3263	12	13	25
Well contrasted colors	0.5882	0.5151	0.3077	-0.8688	4	9	13
Being able to customize the look and feel	1.105	1.1508	0.25	-1.3426	6	18	24
The ability to navigate without a mouse	1.027	0.8578	0.2	-1.6247	4	16	20
Support for text-to-speech	0.9023	0.6459	0.1739	-1.7628	4	19	23

Figur 1: Skjermdump av statistikk-oversikten til undersøkelsen som ble besvart i brukertestene

Disse er ikke spesifikke tilbakemeldinger til vår nettside, men brukernes generelle preferanser, som vi anvendte til å hjelpe oss med å prioritere hva som burde fikses på.

Svarene i Google Forms var generelt sett positive. 8 av 9 respondenter “Fant lett frem til alt jeg skulle” og mente at “Systemet fungerte feilfritt”. Det var mest misnøye med designet, som 5 av 9 mente at det var “et par ting som burde forbedres med”.

I brukertesten for forskere ble de bedt om å logge seg inn på en konto, opprette en undersøkelse og finne frem til hvordan de kan dele denne undersøkelsen med dommere. De ble deretter bedt om å finne frem til statistikk for undersøkelsen som dommerne svarte på i sin brukertest, og laste ned rådataene til denne.

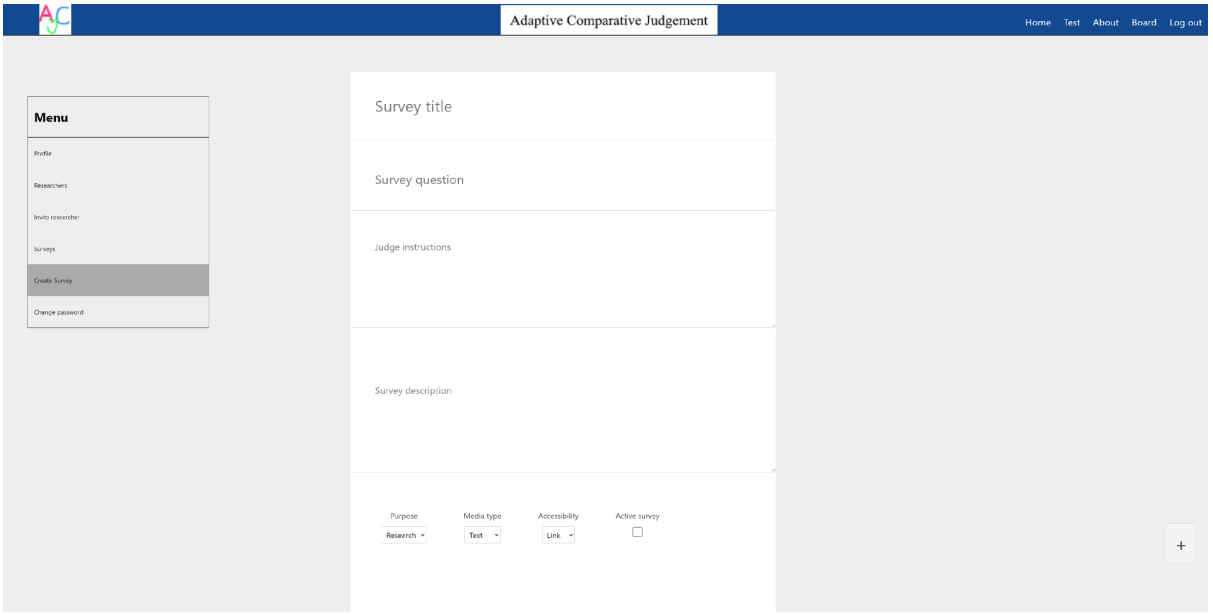
Brukertesten til forskere viste flere forbedringspunkter. 4 av 7 respondenter “Fant frem til alt, men mente at noen ting var lite intuitive” og mente at “Det oppsto en liten feil i testingen min”. Basert på svarene i undersøkelsen, og at 3 av 7 mente det “Kunne vært lettere” å opprette en undersøkelse, virket det som om de fleste problemene stammet fra “Create Survey” siden. Gjennomgående forvirringsmomenter inkluderte plasseringen av “Add item”-knappen og input-feltet “Expected Comparisons”, som mange respondenter slet med å forstå hva var, samt hvorfor de fikk beskjed om invalid input.

Designmessig synes 6 av 7 respondenter at “Det er et par ting som burde forbedres med designet”. Det var større variasjon i begrunnelsene for dette enn det var for dommere, men

hovedpunktene var større mellomrom mellom feltene i "Create Survey"-siden, og generelt bedre fargekontrast på nettsiden.

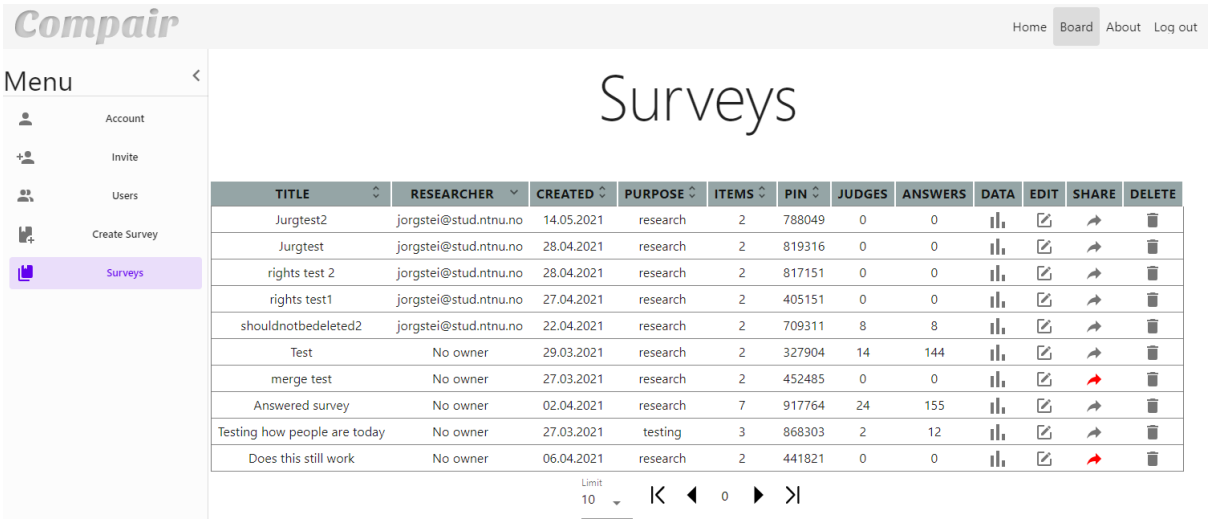
4.1.2 Design

Vi hadde to iterasjoner av designet på nettsiden. Det første designet ble laget fra bunnen av med rå CSS og HTML, og det andre benyttet UI biblioteket "Svelte Materialify". Vi byttet til Svelte Materialify etter ønske fra produkteier rett før vi utførte brukertester.



Figur 2: Skjermdump av den gamle siden for å lage en undersøkelse, før vi begynte å anvende Svelte Materialify

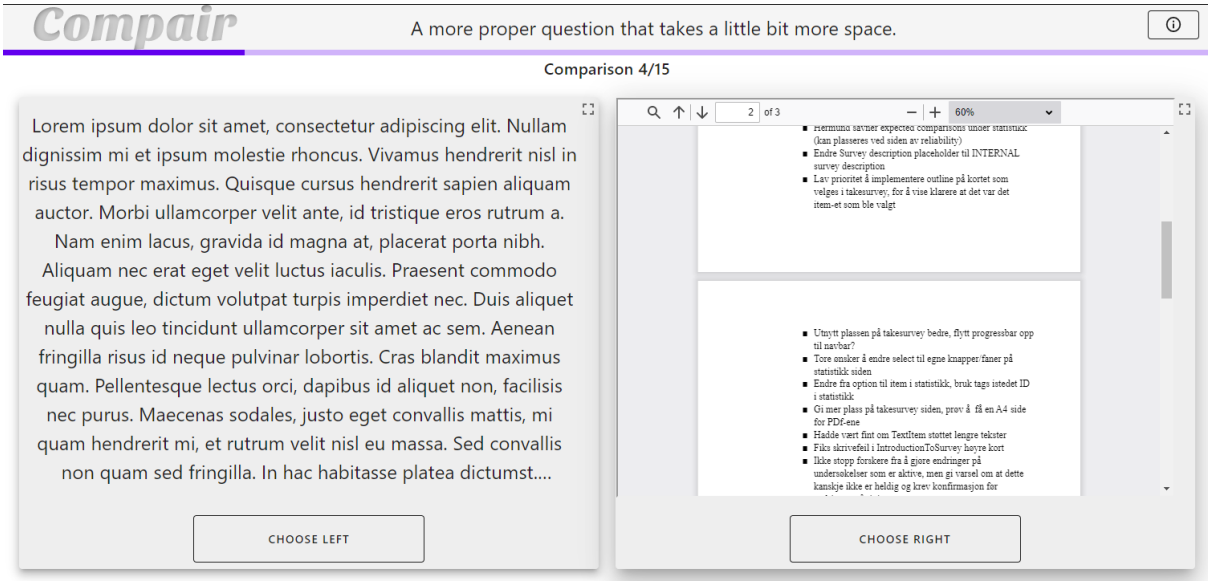
I figuren ovenfor ser vi første iterasjon av designet. Selv om det fungerte, var vi helt enige med produkteier at denne trengte en oppdatering.



Figur 3: Skjermdump av oversikten over undersøkelser etter vi begynte å anvende Svelte-Materialify

Vi benyttet oss av material design ikoner for å gjøre det lettere for brukere å identifisere menyvalg og lenker/funksjoner. Navigasjonsbar og menyen på administrasjonssiden viser også hvor du befinner deg ved å endre bakgrunnsfarge.

Når en undersøkelse skal besvares, spesielt på nettbrett, var det viktig for produkteier at alt innholdet fikk plass på et skjermbilde.



Figur 4: Skjermdump av siden for å ta en undersøkelse, med en lang tekst og en PDF som alternativ.

I figuren over kan vi se en undersøkelse med både tekst og PDF som svaralternativ. Vi ser at navigasjonsbaren har endret utseende til å inneholde spørsmålet i undersøkelsen og en knapp for å åpne en informasjonsside. Vi har også to indikatorer på progresjonen av undersøkelsen. Dette har som mål å vise dommeren hvor mange spørsmål de har igjen å besvare, og kan hjelpe med at de ikke forlater undersøkelsen midt i fordi de har gått lei og ikke vet hvor mye som gjenstår. Vi ser at tekst som ikke vil passe på en side blir forkortet og påført ellipser, og støtter fullskjerms-visning med scrolling ved å klikke på knappen oppe i høyre hjørne i kortene.

4.2 Ingeniørfaglige resultater

4.2.1 Oppnåelse av brukernes behov

Nedenfor er en tabell basert på brukernes behov fra visjonsdokumentet [1] som vi kom fram til sammen med produkteier. Behovene med prioritet 1 er det vi ble enige om skulle inngå i

MVP. Vi har lagt til i hvilken grad vi mener vi har oppfylt behovet med en kommentar på hvorfor i kolonnen til høyre.

Behov	**Prioritet (1-4)**	**Foreslått løsning**	**Grad av oppnåelse**
1 - Registrer admin-bruker	1	En admin-bruker blir opprettet manuelt ved oppsett av produktet. Deretter kan denne brukeren benyttes til å lage nye admin brukere	**Høy** Første admin-bruker registreres automatisk ved oppsett basert på miljøvariabler. Denne kan deretter brukes for å invitere andre admin-brukere.
2 - Registrer forskere	1	Tillate admin-bruker å opprette nye forsker-brukere, evt. invitere til registrering via mail.	**Høy** Fungerer ved at admin inviterer forskere via mail slik at brukeren setter eget passord
3 - Innlogging	1	Logger på med e-post og passord	**Høy** Gjøres via “Log in”-tabben i navbar
4 - Endre passord	1	Gitt at bruker vet passordet sitt og får logget på så kan den endre passordet fra “Min profil” ved å fylle inn nåværende passord, nytt passord og gjentatt nytt passord	**Høy** Kan gjøres under “Account”-tabben i menyen
5 - Forsker skal kunne se oversikt over undersøkelsene sine.	1	En “admin”-side med lett tilgang til undersøkelsene, med videre navigasjon for statistikk/analyse av disse.	**Høy** Kan ses under “Surveys” i menyen som lister ut undersøkelser i tabellarisk form og har videre navigasjon
6 - Invitere dommer vha. lenke	1	Generer delbar lenke som tar dommer med til undersøkelsen	**Høy** Kan kopieres til clipboard ved å trykke på “Share”-knappen i Surveys. Til forskjell fra oppgaveteksten forsvant ønsket om å passordbeskytte lenkene, og i samtale med produkteier ble vi enige om at både lenke og PIN kode brukes (som separate måter å delta i undersøkelsen)
7 - Opprette undersøkelse	1	En forsker skal kunne opprette en undersøkelse med svaralternativer i form	**Høy** Kan gjøres under “Create

		av ren tekst.	survey”-tabben i menyen
8 - Oversikt over brukere	1	En admin skal ha oversikt over alle brukere i systemet.	**Høy** Bare tilgjengelig for admin-brukere under “Users”-tabben i menyen
9 - Slett bruker	1	Enhver bruker skal ha muligheten til å slette sin egen bruker og alle personalia-opplysninger i systemet. Admin skal også ha rettighet til å slette enhver bruker.	**Høy** Sletting av egen konto gjøres under “Account”-tabben i menyen. Admin kan slette brukere fra “Users”-tabben. Undersøkelser som slettede brukere eier blir ikke slettet når kontoen slettes av brukeren, men admin har valg for dette.
10 - Dommer skal kunne ta undersøkelse	1	Generere X antall par som skal sammenlignes av en dommer, presenter disse en etter en fram til X er unnagjort, eller brukeren forlater sesjonen. Lagre resultater underveis.	**Høy** En dommer kan ta en undersøkelse ved å motta en link eller å skrive inn en kode i landing page.
11 - Generere en lineær rangering	1	Generere en lineær rangering basert på svarene til undersøkelsen. Skal gjøres når en forsker trykker på “Run analysis”.	**Delvis/Høy** Systemet mottar en lineær rangering av estimate-modulen som presenteres for forskere under “Data”-tabben i Surveys. Analysen skjer automatisk, ikke ved trykk på en knapp. Dette er grunnet at vi ikke lagrer tidligere analyse av svarene i databasen, bare rådataene.
12 - Forsker skal kunne hente ut rådata	1	Forskere skal ha full tilgang til rådata de eier, slik at de kan gjennomføre de typene analyser de selv vil.	**Høy** Forskere kan laste ned rådata i form av en csv-fil på formen: judgeID, leftItemTag, rightItemTag, winner
13 - Rediger undersøkelse	2	La en forsker redigere undersøkelsen sin etter at den har blitt lagd, på samme måte som før	**Høy** En forsker kan redigere undersøkelsen etter den er lagd. De kan og redigere på items og expected comparisons i en aktiv undersøkelse, til tross for at dette invaliderer tidligere

			samlet data. Brukeren blir varslet og må huke av en checkbox som sier at de forstår konsekvensene før de får endre disse feltene. Dette er etter ønske fra produkteier.
14 - Slett undersøkelse	2	En admin skal kunne slette en undersøkelse, og bestemme om de bare vil slette undersøkelsen, svarene, eller begge deler.	**Høy** En admin kan slette en undersøkelse i systemet for godt. En forsker kan "slette" en undersøkelse de eier i den forstand at de melder seg av som owner av den undersøkelsen. For å faktisk slette selve undersøkelsen fra systemet må de kontakte en admin. På grunn av akseptansekrav om sletting av items er siste bit av behovet ikke valid, da det er ønsket at svar tilhørende item skal slettes, og dermed slettes alle svarene uten noe valg om å unngå dette.
15 - Endre tilgjengelighet til undersøkelse	2	En forsker skal kunne endre tilgjengeligheten på undersøkelsene sine.	**Delvis/Høy** En forsker kan bestemme i hvilken grad de vil gi med-forskere innsyn i undersøkelsen, ved hjelp av rettighetsnivåene: Manage members, Edit survey og View results. Det finnes ikke funksjonalitet for å dele en undersøkelses med alle andre forskere på plattformen.
16 - Invitere dommer vha. kode	2	Generer en kode for undersøkelsen som kan skrives inn på forsiden for å ta undersøkelsen.	**Høy** En forsker kan dele en aktiv undersøkelse sin 6-sifrede PIN-kode ved å kopiere den fra tabellen i Surveys. En dommer kan dermed skrive inn denne på forsiden og ta undersøkelsen
17 - Å gjennomføre en undersøkelse skal se bra ut på utvalgte nettbrett-resolusjoner	2	Design som ikke krever horisontal scrolling.	**Høy** Det går an å gjennomføre undersøkelser på nettbrett, men dette kan alltid forbedres.Vi har laget designet etter spesifikke ønsker fra produkteier slik at alt man trenger når man tar en

			undersøkelse på nettbrett passer på en side og ikke støtter scrolling, men har støtte for å blåse opp svaralternativer slik at disse kan scrolles.
18 - Filtrere/søke i undersøkelser	2	Søkefelt med valg på hvilke felter man søker på og søketekst.	**Delvis** Brukere kan ikke søke på undersøkelser, men de kan søke på medforskere basert på email og fullt navn. Kode for søk er implementert på serversiden, men mangler søke-komponent på frontend.
19 - Glemt passord funksjonalitet	2	En bruker som har glemt passordet kan trykke på "glemt passord" på innloggingsskjermen og få tilsendt en lenke hvor man kan sette nytt passord uten å huske det gamle.	**Høy** En bruker kan opprette et nytt passord ved hjelp av å trykke på en lenke som sender en mail til brukeren med en lenke for registrering av nytt passord
20 - La flere forskere jobbe sammen på en undersøkelse	2	La en eier av en undersøkelse legge til flere eiere når man oppretter eller redigerer undersøkelsen.	**Høy** En forsker kan legge til medforskere under Create/Edit survey
21 - Hurtigsortering av undersøkelser	2	Sortere listen over undersøkelser ved å klikke på attributten man vil sortere etter i listen over undersøkelser(default dato opprettet)	**Høy** Brukeren kan sortere undersøkelser etter tittel, opprettelses-dato, antall items, mail til eier av undersøkelsen og PIN-kode
22 - Tilby svaralternativer i flere medietyper	2	Støtte for flere medietyper (rentekst, bilder, pdf, latex), støtte for generell gradering og forskning.	**Delvis** Systemet støtter ren-tekst og PDF. Vi valgte PDF framfor bilder og latex da begge disse kan konverteres til PDF.
23 - Favoritter/hurtigaksess til undersøkelser	3		**Ikke implementert**
24 - Tilby analyse av dataen	3	Tilby så mye av funksjonaliteten fra tidligere som mulig	**Høy** Det anvendes en estimate-modul utviklet i R for å analysere svarene. Denne modulen er skrevet av

			produkteier.
25 - Presentere en reliabilitetskonstant for enhver besvart undersøkelse	3	Implementere chronbach's alpha som en indikator på om undersøkelsen trenger flere respondenter	**Høy** En reliabilitetskonstant (ikke chronsbach's alpha) oppgis under RawSurveyData-komponente n som kan nås ved å trykke på "Data"-tabben i Surveys.
26 - Oversikt over antall respondenter og antall sammenligninger	3	I oversikten over en studie skal forskeren få se enkel informasjon om hvordan det går med undersøkelsen	**Høy** Forskere kan se informasjon over antall respondenter og antall sammenligninger både i Surveys og i RawSurveyData
27 - Dele undersøkelse med alle forskere på plattformen	3	De skal kunne publisere undersøkelsen uten data til registrerte brukere på plattformen, men også kunne velge å gjøre den privat igjen.	**Ikke implementert**
28 - Brukere skal kunne lese nettsiden på norsk	4	Støtte flere språk, i første runde hovedsakelig engelsk og norsk.	**Ikke implementert** Plattformen er utviklet med engelsk som språk. Eventuell utviding krever omgjøring av alle strenger til variabler og forskjellige "dictionaries" for forskjellige språk. Spares til videreutvikling
29 - Nettsiden skal se bra ut på andre mobile enheter	4	Sømløst design for mobile resolusjoner som ikke krever horisontal scrolling.	**Lite** Det er ikke gjort spesielle tiltak for å tilrettelegge for mobile enheter utenom å svare på undersøkelser på nettbrett.
30 - Statistikk for Respondent-spesifikk informasjon angående left/right bias, median-tid på valg og infit-koeffisient.	4	Gi forskeren muligheten til å detektere og slette besvarelser som er så statistisk usannsynlige at de ikke anses som genuine svar.	**Høy** Systemet gir forsker informasjon om left/right-bias, median tid per svar, gjennomsnittstid per svar og infit/outfit under By judge-tabben i RawSurveyData
31 - Progresjons-graf for undersøkelsen	4	Tilby en graf som gir forskeren en oversikt over hvordan undersøkelsen har utviklet seg med tid. For eksempel grafe chronbachs alpha mot tid.	**Ikke implementert** Denne ble lavt prioritert, og vi klarte ikke helt å bli enige om en god måte å gjøre dette på i forhold til å lagre bare reliabiliteten (som ofte bare

			var NA), eller en hel analyse osv.
32 - Nettsiden skal se bra ut for innloggede forskere på utvalgte nettbrett-resolusjoner	4	Å hindre at forskere må scrolle horisontalt, og å sikre at alt ser bra ut.	**Delvis** På en standard PC skjerm i dag (>1080p) skal det ikke være ofte at man trenger å scrolle horisontalt, men vi har ikke spesifikt implementert løsninger for nettbrett utenom siden man svarer på undersøkelser. Produkteier har testet siden på nettbrett og sagt at det ser bra ut.

Figur 5: Brukerbehov med beskrivelse, prioritet og oppnåelsesgrad

De ikke-funksjonelle kravene har vi delvis bestemt selv, med unntak om ønsket om internasjonalisering og viktigheten av brukeropplevelsen. De resterende punktene som er lagt til anser vi som noe alle nettsider burde ha litt fokus på.

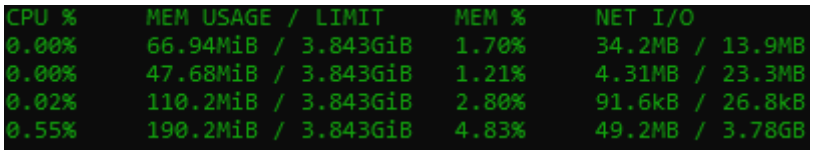
Krav	**Beskrivelse**	**Grad av oppnåelse**
God MMI	Det legges vekt på at brukeropplevelsen er i fokus. Her viktigst at det er enkelt å svare på undersøkelsen, etterfulgt av erfaringen til researcher brukerne, og til slutt administratoren. Disse kravene skal testes ved hjelp av brukertester underveis i prosjektet.	**Delvis/Høy** Vi har etter beste evne prøvd å gjøre produktet intuitivt å bruke, og å gi god tilbakemelding på input fra brukeren. Vi har gjennom dialog med produkteier, erfaring fra tidligere prosjekter og gjennom brukertester kommet fram til utseende og designet av nettsiden slik det er i dag.
Ha et universelt utformet produkt	Det er viktig at så mange som mulig skal kunne bruke produktet vårt uten å slite med brukergrensesnittet. I hovedsak skal vi fokusere på å sikre brukbarhet for fargeblinde og svaksynte, ved hjelp av blant annet lesbar skrift med god kontrast. Vi vil gjerne og at nettsiden skal være lett for skjermleser/"tekst til tale"-software å lese. Vi vil gjerne og at nettsiden skal være mulig å navigere uten bruk av mus. Dette inkluderer å kunne bruke tab for å hoppe fra element til element, og å kunne bruke enter i stedet for å trykke på en knapp.	**Delvis/Lite** Produkteier har ikke ytret noen spesifikke tanker om UU, men vi har prøvd å gjøre nettsiden brukbar med tab-navigering, og har god kontrast på tekst og elementer (målt med google chrome dev tools). Dette er definitivt et forbedringspunkt.
Flerspråklig	Nettsiden lages i første omgang med engelsk som språk, men det er ønskelig å kunne internasjonalisere.	**Ikke implementert** Vi har desverre ikke tatt internasjonalisering i betraktning, mye grunnet at å lese nettsiden på norsk fikk prioritet 4.
Sikkerhet	Systemet må støtte HTTPS, sikker lagring av brukerinformasjon (hash passord + salt)	**Høy** Vi benytter HTTPS mellom klienten og tjenestene den kan kontakte. Vi lagrer aldri klartekst passord, og bruker pbkdf2 med 1 million iterasjoner av sha256 med salting for å

		hashe passordet. Hash og salt lagres i databasen.

Figur 6: Ikke-funksjonelle krav med beskrivelse og oppnåelsesgrad

4.2.2 Ytelse og ressursbruk

Siden en av de store prioriteringene til produkteier er en god brukeropplevelse, og produkteier selv må stå for hosting av plattformen har vi hatt fokus på ytelse og ressursbruk.



Figur 7: Utklipp fra docker stats. Rekkefølgen på radene er backend, frontend, analyse og database.

Vi kan se at minnebruken for hele plattformen er under 420MB, og fra NET I/O kolonnen ser vi at det primært gjøres lese-operasjoner mot databasen. Årsaken til forskjellen på datamengden her er at database-containeren har vært oppe mye lengre enn frontend og backend, da disse har blitt startet på nytt når vi har oppdatert versjonen som kjører på serveren. Den lave minnebruken skyldes i stor grad at tre av de fire modulene er tilstandsløs. Utenom databasen endres ikke tilstanden i modulene etter en forespørsel er avsluttet. I tilfellet for backend er dette et bevisst valg, og kan ha gått på bekostning av ting som bruker-sesjoner og rate-limiting.

Vi har også gjort noen enkle benchmarks for å måle ytelsen til databasen.

Test mot User entitet, 100k dokumenter	**Tid**
Sorter på attributt “_id”, limit 5	54.90ms
Sorter på attributt “_id”, limit 10	57.49ms
Sorter på attributt “_id”, limit 25	57.00ms
Sorter på attributt “_id”, limit 50	54.86ms
Sorter på attributt “_id”, limit 100	65.84ms
Sorter på attributt “email”, limit 5	52.54ms
Sorter på attributt “email”, limit 10	61.98ms
Sorter på attributt “email”, limit 25	60.75ms
Sorter på attributt “email”, limit 50	61.62ms
Sorter på attributt “email”, limit 100	68.59ms

Søk med tekst “hotmail”, limit 5	5.558ms
Søk med tekst “gmail”, limit 5	4.319ms
Søk med tekst “Seivaag”, limit 5	43.42ms
Søk med tekst “lasse”, limit 5	48.58ms
Søk med tekst “Ca”, limit 5	4.235ms
Søk med tekst “Ma”, limit 5	3.551ms
Søk med tekst “Mo”, limit 5	5.933ms

Figur 8: Benchmark mot database med 100 000 “User” dokumenter

Disse testene ble gjort fra en utviklingsmaskin mot en annen som hostet databasen på et kablet lokalt nett, og det vil derfor være noe variasjon grunnet dette (typisk under 1ms). Tiden ble målt med *“process.hrtime()”* rundt databasekallet. Sortering og søking er de databasekallene vi anser som mest krevende, og det er derfor disse ble valgt. Vi ser at sortering i de fleste tilfellene er raskere på _id feltet enn email feltet, dette kan skyldes at email er en streng, mens _id kan sorteres numerisk. Søk krever litt mindre tid, og vi ser at det er noen avvik: “Seivaag” og “lasse”. Dette skyldes at vi begrenser søket til 5 resultater. I alle de andre tilfellene fant databasen 5 resultater før den søkte gjennom hele collection, og returnerte dermed tidlig. Det verste tilfellet er der det finnes færre matcher enn limit, grunnet at hele collection må sjekkes.

4.2.3 Overleveringsplan og opplæring

Siden plattformen skal videreutvikles av produkteier har vi laget en overleveringsplan [19] og en brukerguide [20]. Dette er i tillegg til dokumenter som systemdokumentasjon og har som hensikt å guide videre utvikling og bruk av systemet. I brukerguiden er det beskrevet rutiner for hvordan man kan endre for eksempel utvalg av spørsmål til dommere og hvor man endrer innhold i mailer som blir sendt fra systemet. Dette har dukket opp som konkrete punkter som produkteier ønsker å eksperimentere med. Det er i tillegg generelle forklaringer og kodeeksempler på hvordan teknologiene vi anvender, blant annet Svelte og Mongoose, fungerer.

4.3 Administrative resultater

4.3.1 Milepæler

I henhold til prosjektplanen [21] har vi i stor grad nådd milepælene til, eller før fristen vi planla. Vi hadde oppstartsmøte 20.01.21 hvor vi ble enige om kjerne-teknologier vi skulle bruke, dette var milepæl en. Milepæl to var å få godkjent et førsteutkast til front-end. Vi hadde et brukbart førsteutkast vi viste fram på møtet 27.01.21, og vi fikk feedback om å endre hvordan en dommer svarer på undersøkelsen. Vi mottok ingen negative kommentarer på designet av front-end og vi anser dermed milepæl to til å ha blitt gjennomført innen tidsfristen vi satte.

Vi hadde en fungerende løsning for bruker-registrering og innlogging 08.02.21, og denne milepælen var satt til 10.02.21. Denne løsningen har senere blitt endret, men vi hadde oppnådd milepæl tre.

Den neste planlagte milepælen var en ferdig MVP, denne var satt til å være fullført 24.03.21, og det kan diskuteres litt om når denne faktisk ble ferdig, basert på endringer i kravene til MVP, og litt mangelfulle møtereferater. I møtet 24.02.21 ble det sagt at vi nesten var i mål med MVP, basert på visjonsdokument og prosjektplan. Vi fikk bekreftet av produkteier at de var enige i at MVP var godkjent 07.04.21, litt etter skjemaet i henhold til planen.

Vi startet å planlegge brukertester i møtet 07.04.21, og godtok planen for gjennomføring 12.04.21. I møtet 21.04.21 gikk vi gjennom resultatene av brukertesten og bestemte noen av de siste endringene vi ville gjøre. Dette var milepæl fem, å beslutte siste systemendringer basert på gjennomførte brukertester. Vi fortsatte med “brukertilpasninger” etter dette tidspunktet, men ingen ny funksjonalitet. De to siste milepælene er innlevering av rapporten som er satt til 20.05.21, og presentasjon for produkteiere som etter planen var satt til 26.05.21, men nå er bestemt skal gjøres 20.05.21 etter innlevering av prosjektet.

Alt i alt har vi holdt oss veldig nær planen vi satte opp; men vi har gjort mye mer enn det som var planlagt i MVP. Behov med lavere prioritet ble jobbet på samtidig som vi fokuserte på MVP, da vi så at vi hadde god tid til dette. Det ble også gjort mye nytt, og mange endringer mellom milepæl for MVP og milepæl fem.

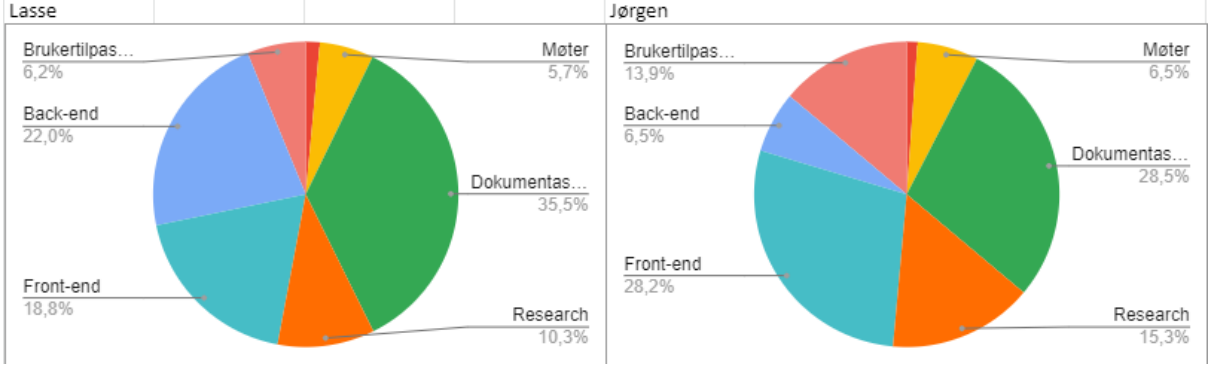
4.3.2 Aktiviteter og timeliste

Arbeidsmengden i faget TDAT3001 var anslått til 500 timer. I starten av prosjektet mottok vi ikke informasjon om hvor mye av dette kom til å bli brukt på andre deler av faget enn selve prosjektet. I prosjektplanen anslo vi dermed 435 timer per student til prosjektet. Disse timene ble fordelt på aktivitetene vi satte opp. Vi anslo 50 timer hver på research, 100 timer hver på front-end utvikling, 125 timer hver på back-end utvikling, 125 timer hver på dokumentasjon og 35 timer hver på administrasjon, herunder 25 timer hver på møter og 10 timer hver på å føre timeliste og arbeidslogg. Timelisten vi har ført har vært basert på disse planlagte aktivitetene, samt en ekstra aktivitet "Brukertilpasning" som veileder ba oss legge til mot slutten av prosjektet.

Sluttrapport total timetall summert pr deltaker pr aktivitet

Aktivitet	Lasse	Jørgen	Total gruppearbeid sum pr aktivitet
Timeliste	7,5	5,5	13
Møter	29	31,5	60,5
Dokumentasjon	179,5	138,5	318
Research	52	74,5	126,5
Front-end	95	137	232
Back-end	111,5	31,5	143
Brukertilpasning	31,5	67,5	99
Total arbeid pr student	506	486	992

Startdato	12.01.2021
Sluttdato	20.05.2021
Antall dager	128
Dagens dato	19.05.2021
Dager igjen	1



Figur 9: Timeoversikt per 19.05.2021

Ovenfor ser vi en oversikt over hvordan tiden ble brukt i prosjektet. Dette inkluderer ikke andre aktiviteter slik som workshops og forelesninger i faget TDAT3001.

4.3.3 Utviklingsprosessen

Vi har holdt oss veldig nær utviklingsprosessen vi valgte, med bi-ukentlige møter med produkteier hvor vi har vist fram produktet, fått feedback og planlagt hva som skal gjøres videre eller må endres på. Vi har også brukt oppgavetavlen på trello for å holde styr på hva hver av oss jobber med, og for å ha kontroll på hva som gjenstår.

Kapittel 5: Diskusjon

I forrige kapittel vises det at vi i høy grad har oppfylt behovene til brukerne av systemet. Alt som ikke var prioritet 1 (MVP) ble beskrevet i møter som "nice to have" og er derfor å anse som tillegg til det produkteier forventet.

5.1 Produktet

I starten av prosjektet presenterte produkteier litt ekstra informasjon om hva de ønsket utover det som sto i oppgaveteksten. De foreslo blant annet et spesifikt format på rådata, noen metoder for analyse av data og noen foreslåtte utvelgingsmetoder for sammenligninger. Vi har derfor brukt en del tid på å bygge opp listen med behov i visjonsdokumentet. Vi ville ha med så mye som mulig i denne listen, selv om vi ikke nødvendigvis kom til å rekke å implementere alt. Tanken bak dette var å kunne se for oss den idèelle versjonen av produktet, fra produkteiers side. Vi mener at å ha denne komplette listen med behov har hjulpet oss med å ikke kun fokusere på å oppnå MVP, men i stedet løfte blikket og utvikle en bredere plattform med mer funksjonalitet. Dette håper vi fører til at produkteier kan fokusere på å utvikle gode comparative judgement-funksjoner fremfor å måtte fokusere på systemutvikling.

Fra brukertestene og demoer for produkteier har vi fått positive tilbakemeldinger om hvor intuitiv og navigerbar siden er. Dette er spesielt viktig da dette har blitt gitt høyeste prioritet av produkteier. Et design vil aldri tilfredsstille alle, så brukertesten resulterte i forskjellige forslag til forbedringer. Forslagene ble vurdert i samarbeid med produkteiere, og vi har implementert de vi anså som viktigst. Allerede i møtet 10.03.21 mente produkteier at de foretrakk systemet vårt over No More Marking, som er produktet produkteierne anvendte før dette prosjektet startet. De spesifikke kravene til MVP ble strengere enn de i oppgavebeskrivelsen når vi i samarbeid med produkteierne begynte å utarbeide brukernes behov i visjonsdokumentet, men produkteierne bekreftet 07.04.21 at MVP var oppfylt.

Ut ifra utfordringene som ble satt opp i kapittel 1 har vi spesielt hatt fokus på brukeropplevelsen, altså å forbedre denne ved å minimere friksjon mellom systemet og brukeren. For eksempel kan all kjernefunksjonalitet nås med tre eller færre klikk fra startsiden. Til tross for at dette prinsippet har blitt kritisert av blant annet IEEE [22] i nyere tid, mener vi at vi har klart å overholde dette prinsippet uten at det kommer på bekostning av intuitiviteten til navigeringen. Brukertesting og kommunikasjon med produkteier har gitt oss muligheten til å lage et godt design til nettsiden. Vi har også vært heldige med valg av teknologi når det kommer til hastighet, da erfaringen vår nå viser at Svelte på frontend er ekstremt rask i forhold til tidligere erfaring med for eksempel React, og som vist i resultater-kapittelet er MongoDB også veldig raskt. Siden vi ikke har testet et tilsvarende oppsett med MySQL kan vi ikke objektivt si at MongoDB er bedre, men fra egen erfaring har MongoDB krevd mye mindre arbeid for å gjøre hurtige spørringer enn det vi tidligere har opplevd med MySQL. Ett av de konkrete tiltakene vi har gjort for å gjøre siden mer responsiv er å gjøre flere kall mot REST-APIet samtidig, altså at vi fyrer av alle kallene vi trenger, og så venter til disse er fullført. Vi har også prøvd å gi god tilbakemelding på input fra brukeren, for eksempel med input validering, pop-up meldinger og markering av hvor brukeren befinner seg i menyene. Dette kan alltids forbedres, men vi mener det er et veldig godt utgangspunkt.

Utfordringen med å velge infrastruktur løste seg i stor grad med kommunikasjon med produkteier. Produkteier har erfaring med utvikling med noen teknologier fra før, og siden de skal videreutvikle produktet var det naturlig at vi valgte de teknologiene de har erfaring med. Utover de grunnleggende teknologiene har vi valgt populære biblioteker som i stor grad er godt dokumentert. Et unntak her er UI rammeverket “Svelte Materialify”, som på tross av å være det mest populære, og den beste kandidaten på tidspunktet det ble valgt, endte opp med å annonsere at det blir utfaset mot slutten av prosjektet. Vi har snakket med produkteier om dette i møtet 05.05.2021, og de var enige i at vi kunne ikke vist at dette ville skje, og at det var det rette valget når vi valgte det.

I forhold til utfordringen med å ta hensyn til at systemet skal videreutvikles har vi prøvd å ha fokus på dette siden starten av prosjektet hvor vi har valgt teknologier vi vet produkteier har erfaring med. Vi har også prøvd å forholde oss til kjente metoder og arkitektur for systemutvikling, slik som REST-API og korresponderende service-objekter på frontend. På

grunn av modulariseringen av koden og rutene på backend kreves det ikke oversikt over hele systemet for å for eksempel endre hvilken utvelgingsmetode som benyttes når en dommer svarer på en undersøkelse. Vi har også hatt fokus på å dokumentere så mye vi kan, og spesielt det produkteier har sagt at de ønsker, slik som en god oversikt over filstruktur og instrukser for å finne fram til konkrete sub-systemer slik som utvalg av svaralternativer.

5.2 Problemer og mangler

Et problem som oppstod når vi endret fra at administrator-bruker oppretter nye brukerkontoer direkte, til at de inviterer nye brukere på mail var at vi ikke fikk til å sende mail uten å oppgi brukernavn og passord til våre personlige NTNU-brukere. Det var ikke et problem å kunne sende mail med brukernavn og passord fra utviklingsmiljøet eller produksjonsmiljøet, men verken vi eller produkteier ønsket å måtte oppgi kredensialer for å sende mail fra tjenesten. Dette var et problem med SMTP-konfigurasjonen vår, og ble løst ved at NTNU Hjelp ga oss korrekte innstillinger slik at vi slipper å inkludere brukernavn og passord.

Det er fremdeles områder med forbedringspotensiale. Et av disse er å støtte transaksjoner som vil forbedre integriteten av dataen og minske friksjon for brukerne. Per nå er det en mulighet for at en undersøkelse blir opprettet uten alle filvedleggene brukeren ønsket, da dette krever flere databasekall mot forskjellige collections. Vi har ikke opplevd at dette har skjedd selv, men det er et mulig problem. Et annet punkt er at vi ikke har funksjonalitet for å lage en kopi av en eksisterende undersøkelse. Dette var en av akseptansekravene for “Opprette undersøkelse” i kravdokumentasjonen [2]. Selv om dette var et akseptansekrav så var det ikke høyt prioritert. Funksjonaliteten ble nevnt som en mulig løsning på problematikken rundt å opprette “test” undersøkelser for tilbakemelding fra medforskere, men siden vi støtter å gjøre undersøkelsene aktiv eller inaktiv, kan gi direkte tilgang til medforskere i undersøkelsen og har mulighet til å sette “purpose” til “testing” anser vi ikke dette som en stor mangel.

Det er implementert en rute for å søke på undersøkelser og brukere, men det er ikke opprettet en frontend-komponent som benytter seg av disse rutene. Dette ble rett og slett prioritert bort mot slutten av prosjektet, delvis grunnet at vi allerede hadde implementert sortering. Det finnes en frontend-komponent som brukes når man oppretter undersøkelser. Denne søker etter brukere, men de komponentene som mangler tenkes spesifikt brukt i den tabbelariske oversikten over brukere og undersøkelser. Dette vil bli nevnt som forslag til videre arbeid.

Vi har ikke implementert rate-limiting på REST-APIet. Dette skyldes at vi ønsket å beholde tilstandløsheten, og fordi vi ikke kunne bestemme hvor mange requests en normal bruker kan finne på å sende mot de spesifikke rutene. Vi vil anbefale å spesielt implementere dette på rutene for opprettelse av konto, innlogging og operasjoner som søk og sortering da disse er ressurskrevende eller kan misbrukes for å finne innloggingskredensialer.

Vi har ikke tatt høyde for vilkår for bruk av tjenesten, samt overensstemmelse med gjeldende lovverk i forhold til bruk av informasjonskapsler og lagring av persondata og brukergenerert data. Dette føler vi krever input fra noen med mer ekspertise som kan fastslå hva som kreves for å være innenfor loven. Det vi har gjort er å ikke lagre informasjon om sesjoner, bare bruke informasjonskapsler til autentisering, og det lagres ingen informasjon som kan knyttes til dommere.

Den største mangelen i systemet i henhold til oppgavebeskrivelsen er adaptiviteten. Ved overlevering tilbyr systemet ingen form for intelligent utvalg av hvilke alternativer som burde sammenlignes for å maksimere reliabiliteten. Utvalget sikrer at brukeren ikke blir stilt samme spørsmål, for eksempel A vs B og B vs A i samme undersøkelse, men utover det er utvalget tilfeldig. Dette var resultatet av en bevisst prioritering som ble gjort sammen med produkteierne. Grunnet at vi bare var to stykker på teamet, er det naturlig at noe funksjonalitet må ofres. Siden utvalget av alternativer er en relativt isolert modul i systemet, krever det ikke at man setter seg inn i hvordan hele systemet fungerer før man kan begynne å eksperimentere med nye måter å velge ut alternativer på. Dette kombinert med produkteiers egenskaper som matematikk-lektorer gjorde det til en naturlig del av systemet å prioritere bort. Til tross for at vi står for at dette var en god prioritering, er utvelgingsalgoritmen en sentral del av ethvert system som anvender adaptive comparative judgement. Det kan spare både forskere og dommere tid og ressurser, og det burde implementeres en bedre algoritme i videreutviklingen.

Produkteier uttrykte et ønske om å støtte flere medietyper, spesifikt ren tekst, PDF, bilder og Word-dokumenter, i oppgaveteksten. LaTeX ble også senere nevnt som et ønsket filformat. Av disse har vi bare implementert ren tekst og PDF. Denne prioriteringen skjedde på basis av tiden vi hadde igjen og at både Word-, bilde- og LaTeX-dokumenter ganske enkelt kan eksporteres eller konverteres til PDF.

5.3 Milepæler og timeliste

Vi kunne allerede i planleggingsfasen ha tatt høyde for videre arbeid etter MVP ved å føre på dette som milepæler. Dette ble ikke gjort da vi fremdeles var usikre på hvor mye vi kom til å få implementert, gitt mengden nye teknologier vi valgte å anvende. Det viste seg i ettertid at vi fikk gjort betydelig mer enn å bare fullføre MVP, og det var mye tid satt av til dette mellom milepæl for godkjent MVP og å vedta siste systemendringer etter endt brukertesting.

I resultatene for aktiviteter og timeliste nevnte vi at vi satte av 435 av de 500 timene i faget per student til prosjektet. Dette viste seg å være et veldig konservativt estimat. Totalt ble det bare brukt ca. 10 timer per person på aktiviteter i faget utenom prosjektet. Disse aktivitetene var hovedsakelig forelesninger og workshops, og ble ikke ført i timelista. Fra kapittel 4.3.2 så vi at vi kom til å ende med 500 +/- 25 timer totalt. Vi ser også at fordelingen av timer på aktiviteter ikke stemmer overens med planen. Spesielt timebruk på backend, frontend (med brukertilpasninger) og dokumentasjon avviker fra det som var planlagt. Vi brukte betydelig mindre tid på backend enn planlagt. Dette skyldes til dels at produkteier uoppfordret utviklet analyse-modulen, men også rett og slett en overestimering av tiden vi kom til å bruke på denne aktiviteten. De resterende aktivitetene brukte vi mer tid på enn planlagt. Spesielt frontend og brukertilpasning tok mye mer tid grunnet en omskrivning av frontend, endringer ønsket etter brukertester og spesifikke ønsker fra produkteier.

5.4 Refleksjon rundt gruppearbeid

Vi har jobbet godt sammen på tross av at vi nesten utelukkende har jobbet digitalt grunnet pandemien. Det har vært perioder hvor vi har jobbet hver for oss, men ved hjelp av oppgaveoversikten på Trello og digital kommunikasjon hadde vi fremdeles god oversikt over hva motparten jobbet med. Det har ikke oppstått problemer hvor vi har gjort overlappende arbeid. Vi har vært flinke til å lufte idèer oss i mellom, og å diskutere problemer og løsninger underveis i prosjektet. For det meste av utviklingsfasen brukte vi kommunikasjonsplattformen Discord oss imellom for å sikre en konstant kommunikasjonsflyt.

Kapittel 6: Konklusjon og videre arbeid

6.1 Svar til utfordringen

Hovedutfordringen i prosjektet var å produsere en fungerende web-plattform for å gjennomføre comparative judgement-studier som ikke lider av de samme problemene som eksisterende løsninger, og dermed kunne oppfylle effektmålene til produkteier. Løsningen vår tilbyr forskere undersøkelsenes rådata, støtte for flere medietyper som svaralternativ og et simplistisk og intuitivt design. Produkteier kan bistå brukere ved hjelp av en admin-bruker, og har mulighet til å videreutvikle systemet etter deres egne preferanser. Siden produkteier drifter hele systemet selv har de full kontroll over forsknings- og bruker-data, og enhver mulighet til å overholde gjeldende lovverk om personvern. Basert på tilbakemeldinger fra produkteier som tilsier at de foretrekker vår løsning framfor dagens løsning er denne utfordringen i sin helhet å anses som overkommet.

Prosjektet har vært preget av kontinuerlig kommunikasjon med produkteier for å sikre at produktet vi utvikler er i henhold til deres visjon. I tillegg har vi gjennomført brukertester for å sikre oss at systemet tilfredsstiller eksterne aktører.

Vi har forsøkt å legge til rette for videreutvikling ved hjelp av dekkende dokumentasjon i form av systemdokumentasjon [13], samt en brukerguide [20] som beskriver hvordan overtakerne av systemet burde gå frem for å implementere funksjonalitet som produkteier har vist interesse for. Brukerguiden inkluderer og en innføring i de viktigste bibliotekene og rammeverkene som er anvendt i prosjektet.

6.2 Økt kompetanse og tips til lignende prosjekter

Gjennom dette prosjektet har vi blitt kjent med flere nye teknologier slik som Svelte og MongoDB. Ett prosjekt er ikke nok til å mestre en ny teknologi, men vi har fått en god forståelse for disse teknologiene. Det har vært en god opplevelse å utvikle en nettside på en helt ny måte, og å oppdage andre måter å gjøre spørringer mot en database på enn SQL. Vi har også fått mer erfaring med å planlegge og utføre prosjekter.

For de som skal gjennomføre lignende prosjekter i fremtiden ønsker vi å meddele vår erfaring fra dette prosjektet. Vi anbefaler å legge stor vekt på hyppig kommunikasjon mellom partene. Dette øker sannsynligheten for at man utvikler rett produkt, med de funksjonene og det designet som er ønsket, og kan føre til mindre bortkastede ressurser ved å ikke implementere unødvendig funksjonalitet. Det er viktig med en felles forståelse mellom partene i prosjektet, og det vil gjøre det lettere å utvikle produktet. Det anbefales spesielt å vise fram produktet til produkteier for å få tilbakemeldinger, samt å huske å dokumentere beslutninger som tas underveis.

6.3 Forslag til videre arbeid

I kapittel 5.2 diskuterte vi problemer og mangler vi er klar over med produktet. Ut ifra disse har vi laget en liste med forslag til videre arbeid:

- En bedre utvelgingsalgoritme.

  Dette mangler for at systemet skal være “adaptivt”. Det kan redusere mengden dommere man trenger, eller antall sammenligninger hver dommer trenger å gjennomføre for å oppnå høyere reliabilitet.
- Støtte for flere medietyper.

  Dette var et behov for brukerne av systemet, og gir muligheten til å vise fram svaralternativer på en mer intuitiv måte basert på hva som sammenlignes (For eksempel LaTeX-støtte for matematiske formler)
- Endre databasetoppsettet til å bruke replica-set, selv om det bare er en node i settet.

  Dette vil forbedre integriteten av dataen og minske problemer hvor flere databasekall avhenger av hverandre, eller må utføres sammen for å unngå referanser til ikke-eksisterende dokumenter. Dette kommer med en kostnad til ytelse og ressursbruk.
- Implementere rate-limiting på rutene til REST-API

  Dette vil redusere muligheten for brute-force angrep mot brukernavn/passord kombinasjoner og kan redusere ressursbruken. Dette kommer med et overhead på ressursbruk per bruker.
- Opprett “vilkår for bruk” som kreves for å lage brukerkonto, og rådfør eksperter for å sikre overensstemmelse med gjeldende lovverk for lagring og bruk av informasjon
- Implementere søke-komponent for tabellene for brukere og undersøkelser

REST-APIet har ruter for søk, men det mangler komponenter på frontend som snakker med disse. (Søk på brukere blir brukt i “opprett undersøkelse”, men det burde lages en egen komponent som brukes for å fylle tabellen i oversikten)

Referanser

[1]. Visjonsdokument. Se kapittel vedlegg eller 064_vedlegg.zip

[2]. Kravdokumentasjon. Se kapittel vedlegg eller 064_vedlegg.zip

[3]. Jones, Ian. Alcock, Lara. Peer assessment without assessment criteria. Studies in Higher Education, volum 39. 2014. Tilgjengelig fra: https://www.tandfonline.com/doi/full/10.1080/03075079.2013.821974?scroll=top&needAccess=true

[4]. Melnikov, Vitalik. Hüllermeier, Eyke. Kaimann, Daniel. Frick, Bernd. Gupta, Pritha. Pairwise versus Pointwise Ranking: A Case Study. Schedae Informaticae, volum 25. 2016. s.77. Tilgjengelig fra: https://www.ejournals.eu/Schedae-Informaticae/2016/Volume-25/art/9011/

[5] Beck, Kent, et al. The Agile Manifesto. Agile Alliance. 2001. [Hentet 15. mai 2021]. Tilgjengelig fra: http://agilemanifesto.org/

[6]. Red Hat. What is a REST API? [Internett]. [Hentet 15. mai 2021]. Tilgjengelig fra: https://www.redhat.com/en/topics/api/what-is-a-rest-api

[7]. DB-Engines. DB-Engines Ranking - popularity ranking of database management systems [Internett]. [Oppdatert mai 2021; Hentet 15. mai 2021]. Tilgjengelig fra: https://db-engines.com/en/ranking

[8]. Li, Lorraine. Database Normalization Explained [Internett]. Towards Data Science. [Oppdatert 2. juli 2019; Hentet 17. mai 2021]. Tilgjengelig fra: https://towardsdatascience.com/database-normalization-explained-53e60a494495

[9]. Gerend, Jason. Containers vs. virtual machines [Internett]. Microsoft. [Oppdatert 21 oktober 2019; Hentet 15. mai 2021]. Tilgjengelig fra: https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm

[10]. Bartnes, Maria. HTTPS. Store Norske Leksikon. 2019. [Oppdatert 25. november 2019; Hentet 15. mai 2021]. Tilgjengelig fra: https://snl.no/HTTPS

[11]. OWASP. A7:2017-Cross-Site Scripting (XSS) [Internett]. 2017. [Hentet 26. april 2021]. Tilgjengelig fra: https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS)

[12]. OWASP. A1:2017-Injection [Internett]. 2017. [Hentet 26. april 2021]. Tilgjengelig fra:

https://owasp.org/www-project-top-ten/2017/A1_2017-Injection

[13]. Systemdokumentasjon. Se kapittel vedlegg eller 064_vedlegg.zip

[14]. Prosjekthåndbok. Se 064_vedlegg.zip

[15]. State of JS. Front-end frameworks [Internett]. 2020. [Hentet 17. mai 2021]. Tilgjengelig fra: https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/

[16]. Harris, Rich. Rethinking reactivity [Internett]. You Gotta Love Frontend 2019. [Hentet 2. mai 2021]. Slides 24-25. Tilgjengelig fra: https://rethinking-reactivity.surge.sh/#slide=24

[17]. Schwaber, Ken. Sutherland, Jeff. The Scrum Guide 2017 [Internett]. [Oppdatert november 2017; Hentet 15. mai 2021]. Tilgjengelig fra: https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

[18]. Analyse av brukertester. Se 064_vedlegg.zip

[19]. Overleveringsplan. Se 064_vedlegg.zip

[20]. Brukerguide. Se 064_vedlegg.zip

[21]. Prosjektplan. Se 064_vedlegg.zip

[22]. IEEE (Institute of Electrical and Electronics Engineers). The 3-click rule: myth or fact? [Internett]. [Hentet 16. mai 2021]. Tilgjengelig fra: https://brand-experience.ieee.org/the-3-click-rule-myth-or-fact/

Vedlegg

Web-plattform for Adaptive Comparative Judgement
Visjonsdokument

Versjon 1.3

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
27/01/21	1.0	Beskrevet visjonen i samarbeid med oppdragsgiver. Fylt ut deler av funksjonskravene, men oppdragsgiver var ikke helt klar på hva som krevdes, så dette må suppleres senere.	Lasse Seivaag, Jørgen Steig
18/02/21	1.1	Implementert endringer som ble klargjort etter demonstrasjon av produktet for oppdragsgivere 10/02/21. Fylt ut funksjoner og egenskaper/krav	Lasse Seivaag, Jørgen Steig
25/02/21	1.2	Utdypt funksjonelle krav og sortert etter prioritet. Endret prioriteringsskalaen fra 1-10 til 1-4.	Lasse Seivaag, Jørgen Steig
04/05/21	1.3	Klargjort for siste versjon	Lasse Seivaag, Jørgen Steig

Innholdsfortegnelse

1. Innledning

Hensikten med oppgaven er å utvikle en web-plattform der forskere ved institutt for lærerutdanning (ILU) ved NTNU kan samle inn data for forskning. Prosjektet vil i hovedsak anvende Adaptive Comparative Judgement for å finne en lineær rangering av objekter, i henhold til deltakerens preferanser. Rangeringen vil være basert på en rekke med spørsmål der deltakeren blir presentert for to og to objekter, og må velge den de foretrekker. Dette prosjektet skal gi forskerne muligheten til å opprette undersøkelser basert på denne teknikken, og gi tilgang til rådata og analyser av undersøkelsene. Senere vil det og være aktuelt å videreutvikle plattformen og gjøre den tilgjengelig for lærere slik at de kan anvende den til vurdering.

Visjonsdokumentet eksisterer for å dokumentere hvilke overordnede mål oppdragsgiver og oppdragstaker har blitt enige om for prosjektet. Ved å få begge parter til å godkjenne dokumentet sikrer vi oss at vi har en felles forståelse av hva som forventes av et ferdig produkt. Partene kan senere referere tilbake til dokumentet for å se hva de har blitt enige om. Det vil i hovedsak anvendes som et beslutningsgrunnlag for videre arbeid.

2. Sammendrag problem og produkt

2.1 Problemsammendrag

Problem med	Lokasjon og tilgjengelighet på data (f.eks. personopplysninger), samt dårlig fleksibilitet i eksisterende løsninger
berører	produkteier
som resultatet av dette	Kan de risikere bøter, færre testsubjekter og vanskeligheter med å lage gode undersøkelser
en vellykket løsning vil	En egenutviklet backend som kan kjøres på servere til produkteier hvor de kan kontrollere hva som blir lagret hvor.

2.2 Produktsammendrag

For	Instituttet for lærerutdanning
som	Trenger en web-plattform som de kan anvende til forskning, der subjektene velger den beste av to alternativer, og systemet produserer en lineær rangering. Plattformen skal kutte ned på antall spørsmål som må bli stilt (n*(n-1)) ved hjelp av transitivitet, statistiske metoder m.m. Være adaptiv.
produktet navngitt	Web-plattform for Adaptive Comparative Judgement.

som	Tilbyr tilgang til rådata, er i henhold til GDPR og kan tilpasses kravene til forskningen. Viktig å kunne presentere alternativer som ren-tekst, bilder eller PDF.
I motsetning til	No More Marking (?), som ikke gir tilgang til rådata, krever PDF, og sender data utenfor EU (Singapore)
Er vårt produkt	Utviklet internt, med muligheter for videreutvikling i fremtiden for å møte nye krav.

3. Overordnet beskrivelse av interessenter og brukere

3.1 Oppsummering interessenter

Navn	Utdypende beskrivelse	Rolle under utviklingen
Produkteiere (Tore & Hermund)	Representerer forskere som ønsker å gjennomføre studier. Mer spesifikt representerer de institutt for lærerutdanning ved NTNU.	Løpende feedback til demoer og generell beslutningstakning. Vil også bistå i å gjennomføre brukertester.

3.2 Oppsummering brukere

Navn	Utdypende beskrivelse	Rolle under utviklingen	Representert av
Forskere	Forskere som ønsker å gjennomføre undersøkelsene og kunne analysere svarene.	Testing av systemet	Produkteiere, evt. med medarbeidere
Dommere	Hvilken som helst person. Trenger ingen ekspertise.	Testing av systemet	Produkteiere med medarbeidere og eventuelle testsubjekter
Administrator	Administrator brukeren skal fungere som forsker, men også ha komplett tilgang til alle entiteter. Dette innebærer opprettelse, sletting, redigering og utlisting.	Testing av systemet	Produkteiere

3.3 Brukermiljøet

Tar utgangspunkt i at produktet skal fungere godt i Chrome/Firefox på PC. Velkjente oppløsninger (responsiv nettside), skal kunne besvares på vanlige iPad / andre kjente nettbrett. Spesifikt “Samsung F6 light” landscape.

3.4 Sammendrag av brukernes behov

Behovene er prioritert fra 1-4, der prioritet 1 er krav for Minimum Viable Product. Prioritet 2 representerer behov som helst burde være oppfylt ved innlevering, men som teknisk sett ikke faller under MVP. Prioritet 3 er behov som det hadde vært fint om vi klarte å oppfylle, men som det ikke egentlig er forventet at vi skal få gjennomført. Vi har ambisjoner om å klare så mange av disse som mulig. Dersom et behov er rangert som 4 anses de som lite viktige for plattformens funksjonalitet, og kommer deretter til å bli veldig nedprioritert. Vi har ikke ambisjoner om å få gjennomført disse, men det er greie ideer som kan jobbes videre på av de neste systemansvarlige.

Behov	**Prioritet (1-4)**	**Påvirker**	**Dagens løsning**	**Foreslått løsning**
1 - Registrer admin-bruker	1	Brukersystem	Ingen	En admin-bruker blir opprettet manuelt ved oppsett av produktet. Deretter kan denne brukeren benyttes til å lage nye admin-brukere
2 - Registrer forskere	1	Brukersystem	Registrere egen bruker på ekstern CJ tjeneste	Tillate admin-bruker å opprette nye forsker brukere, evt. invitere til registrering via mail.
3 - Innlogging	1	Brukersystem	Logger på med e-post og passord	Logger på med e-post og passord
4 - Endre passord	1	Brukersystem	?	Gitt at bruker vet passordet sitt og får logget på så kan den endre passordet fra “Min profil” ved å fylle inn nåværende passord, nytt passord og gjentatt nytt passord
5 - Forsker skal kunne se oversikt over undersøkelsene sine.	1	Brukersystem	Får opp en tabell med undersøkelsene sine.	En “admin”-side med lett tilgang til undersøkelsene, med videre navigasjon for statistikk/analyse av disse.
6 - Invitere dommer vha. lenke	1	Studiesystem	Generer delbar lenke med forventede antall dommere og antall sammenligninger	Generer delbar lenke som tar dommer med til undersøkelsen
7 - Opprette undersøkelse	1	Studiesystem	Tungvindt oppsett som er designet for å gradere elevbesvarelser, mye	En forsker skal kunne opprette en undersøkelse med svaralternativer i form

			manuelt arbeid, krever import av .csv filer og pdf filer	av ren tekst.
8 - Oversikt over brukere	1	Brukersystem	Har ikke tilgang til admin-brukere, dermed ingen løsning.	En admin skal ha oversikt over alle brukere i systemet.
9 - Slett bruker	1	Brukersystem	Ingen	Enhver bruker skal ha muligheten til å slette sin egen bruker og alle personalia-opplysninger i systemet. Admin skal også ha rettighet til å slette enhver bruker.
10 - Dommer skal kunne ta undersøkelse	1	Studiesystem	Dommer får tilsendt lenke på epost. Deretter kan de ta undersøkelsen. Dommeren sammenlikner 2 pdf-er og velger ved å trykke på knapper øverst på siden. Enten venstre eller høyre.	Generere X antall par som skal sammenlignes av en dommer, presenter disse en etter en fram til X er unnagjort, eller brukeren forlater sesjonen. Lagre resultater underveis.
11 - Generere en lineær rangering	1	Studiesystem	Generer lineær rangering basert på svar på undersøkelsen. Skjer når man trykker på “Refresh task”	Generere en lineær rangering basert på svarene til undersøkelsen. Skal gjøres når en forsker trykker på “Run analysis”.
12 - Forsker skal kunne hente ut rådata	1	Studiesystem	Kan ikke hente ut rådata. Får bare analyser basert på dataen.	Forskere skal ha full tilgang til rådata de eier, slik at de kan gjennomføre de typene analyser de selv vil.
13 - Rediger undersøkelse	2	Studiesystem	Kan redigere undersøkelsen ved å navigere til undersøkelsen og klikke på den	La en forsker redigere undersøkelsen sin etter at den har blitt lagd, på samme måte som før
14 - Slett undersøkelse	2	Studiesystem	Kan slette undersøkelsen, men valget er ganske gjemt og man får ikke valg om å ta vare på spesifikke data, alt forsvinner.	En admin skal kunne slette en undersøkelse, og bestemme om de bare vil slette undersøkelsen, svarene, eller begge deler.
15 - Endre tilgjengelighet til undersøkelse	2	Studiesystem	Kan kun legge til flere eiere av undersøkelsen via epost-adresse	En forsker skal kunne endre tilgjengeligheten på undersøkelsene sine.
16 - Invitere dommer vha. kode	2		Ingen, bare delbar leneke	Generer en kode for undersøkelsen som kan skrives inn på forsiden for å ta undersøkelsen.

17 - Å gjennomføre en undersøkelse skal se bra ut på utvalgte nettbrett-resolusjoner	2		Må scrolle i begge dimensjoner for å se hele siden.	Design som ikke krever horisontal scrolling.
18 - Filtrere/søke i undersøkelser	2		Ingen	Søkefelt med valg på hvilke felter man søker på og søketekst.
19 - Glemt passord funksjonalitet	2	Brukersystem	"Glemt passord" knapp på innloggingsbildet, sender en mail man kan trykke på for å sette nytt passord.	En bruker som har glemt passordet kan trykke på "glemt passord" på innloggingsskjermen og få tilsendt en lenke hvor man kan sette nytt passord uten å huske det gamle.
20 - La flere forskere jobbe sammen på en undersøkelse	2	Brukersystem/Studiesystem	Kan invitere flere eiere av undersøkelsen via epost	La en eier av en undersøkelse legge til flere eiere når man oppretter eller redigerer undersøkelsen.
21 - Hurtigsortering av undersøkelser	2	Studiesystem	Kan sortere listen over undersøkelser ved å klikke på attributten man vil sortere etter i listen over undersøkelser	Sortere listen over undersøkelser ved å klikke på attributten man vil sortere etter i listen over undersøkelser(default dato opprettet)
22 - Tilby svaralternativet i flere typer media	2	Studiesystem	Bare støtte for PDF	Støtte for flere medietyper (rentekst, bilder, pdf, latex), støtte for generell gradering og forskning.
23 - Favoritter/hurtigaksess til undersøkelser	3	Brukersystem	Ingen	
24 - Tilby analyse av dataen	3	Studiesystem	Har live oppdatering av hvor mange som har svart, hvor mange spørsmåk hver av disse har svart på, reliabiliteten av modellen, den lineære rangeringen, venstre/høyre bias mm.	Tilby så mye av funksjonaliteten fra tidligere som mulig
25 - Presentere en reliabilitetskonstant for enhver besvart undersøkelse	3	Studiesystem	Implementer i form av reliabilitet-parameter i undersøkelsen	Implementere chronbach's alpha som en indikator på om undersøkelsen trenger flere respondenter
26- Oversikt over antall respondenter og antall sammenligninger	3	Studiesystem	Kan se antall sammenligninger og antall respondenter	I oversikten over en studie skal forskeren få se enkel informasjon om hvordan det

				går med undersøkelsen
27 - Dele undersøkelse med alle forskere på plattformen	3	Studiesystem	Ingen	De skal kunne publisere undersøkelsen uten data til registrerte brukere på plattformen, men også kunne velge å gjøre den privat igjen.
28 - Brukere skal kunne lese nettsiden på norsk	4	Nettsiden	Nettsiden er bare tilgjengelig på engelsk	Støtte flere språk, i første runde hovedsakelig engelsk og norsk.
29 - Nettsiden skal se bra ut på andre mobile enheter	4		Dårlig tilpasset mobile resolusjoner. Må scrolle i begge dimensjoner for å få med seg hele nettsiden.	Sømløst design for mobile resolusjoner som ikke krever horisontal scrolling.
30 - Statistikk for Respondent-spesifikk informasjon angående left/right bias, median-tid på valg og infit-koeffisient.	4	Studiesystem	Alle disse funksjonene tilbys m.m.	Gi forskeren muligheten til å detektere og slette besvarelser som er så statistisk usannsynlige at de ikke anses som genuine svar.
31 - Progresjons-graf for undersøkelsen	4	Studiesystem	Finnes på forsiden av	Tilby en graf som gi forskeren en oversikt over hvordan undersøkelsen har utviklet seg med tid. For eksempel grafe chronbachs alpha mot tid.
32 - Nettsiden skal se bra ut for innloggede forskere på utvalgte nettbrett-resolusjoner	4	Nettsiden	Brukbart, men en del horisontal scrolling	Å hindre at forskere må scrolle horisontalt, og å sikre at alt ser bra ut.

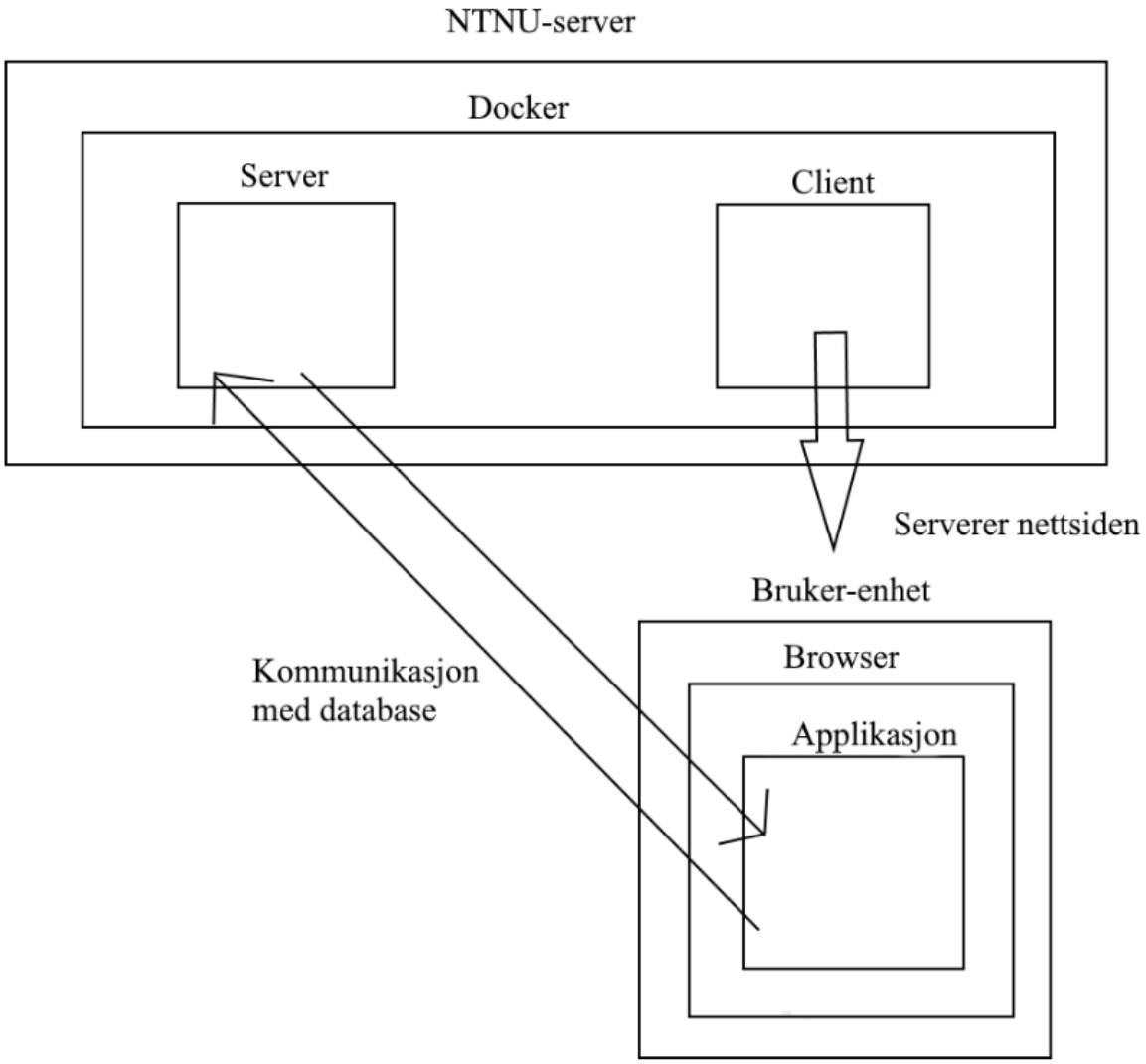
3.5 Alternativer til vårt produkt

"No More Marking" er den største konkurrenten til vårt produkt. Det er en web-plattform som anvender comparative judgement til å utføre studier på lik måte som vårt produkt. Produktet blir hovedsakelig brukt til retting av prøver i skolen, men mangler ifølge oppdragsgiver fleksibilitet når det kommer til opplastning av forskjellige typer svaralternativer (pdf, LATEX, bilder, ren tekst etc). Oppdragsgiver har ikke fått inngått databehandleravtale som gir de kontroll over dataene i undersøkelsen, eller svarene til undersøkelsen, som fører til at de ikke kan overholde GDPR. De får heller ikke tilgang til rådata fra undersøkelsen, så de har begrensede muligheter til å gjennomføre de analysene de

vil.

4. Produktoversikt

4.1 Produktets rolle i brukermiljøet



4.2 Forutsetninger og avhengigheter

Vi forutsetter at brukerne av systemet har korrekt maskinvare og programvare til å benytte produktet slik som beskrevet i 3.3 Brukermiljøet.

Vi er avhengige av at maskinvare som kan fasilitere systemet er tilgjengelig.

At brukernes krav forblir noenlunde konstante er en viktig forutsetning for dokumentet.

5. Produktets funksjonelle egenskaper

Funksjon	Beskrivelse	Brukerkrav
Registrere brukere	Det finnes to typer brukere, admin og researcher. Admin-brukere skal kunne opprette nye admin og- research-brukere. I første omgang kan admin-bruker manuelt opprette kontoen, men det er ønskelig om det er tid å endre dette til å invitere en bruker via epost med en lenke der brukeren selv kan registrere seg. Krever epost-adresse, passord og gjentatt passord. Svake passordkrav for researcher og sterke passordkrav for admin.	1, 2
Logge inn som bruker	Innlogging skjer ved å trykke på “Login” i navigasjonsmenyen, og man skriver inn epost-adresse og sitt passord.	3
Endre passord for bruker	En registrert bruker må ha mulighet til å endre sitt passord. Dette krever at de vet sitt gamle passord, og at de fyller inn nytt passord to ganger.	4
Glemt passord for bruker	En registrert bruker som har glemt sitt passord må ha en måte å resette dette. Dette gjøres ved å sende en mail til brukerens epost adresse, som kan bruker for å sette nytt passord på denne brukeren.	19
Slette bruker	Admin-bruker må ha mulighet til å slette brukere. Dette kan gjøres ved å finne brukeren å trykke på et søppelkasse-ikon, og bekrefte handlingen.	9
Vis eksisterende brukere	Admin-bruker må ha muligheten til å liste opp eksisterende brukere, for oversikt og mulighet til å slette brukere.	8
Opprette undersøkelse	Både admin og researcher brukere skal ha tilgang til å opprette en undersøkelse. Dette innebærer å fylle ut flere felter, slik som tittel, deskripsjon, hvem som skal ha tilgang, hvilken type data som skal sammenlignes og å legge til alle valgene som skal sammenlignes. Det støttes i det aller minste spørsmål i form av renteks, men det er ønskelig å støtte flere medietypert.	7, 22
Rediger undersøkelse	Både admin og researcher brukere skal kunne redigere eksisterende undersøkelser som de har rettigheter til (er eiere av)	13
Legge til medforskere på en undersøkelse	En forsker som lager eller redigerer en undersøkelse må ha mulighet til å legge til medforskere. Dette gjøres ved å søke opp brukeren til medforskeren med navn eller epostadresse.	20
Slette undersøkelse	Både admin og researcher brukere skal kunne slette undersøkelser som de har rettigheter til.	14
Endre tilgjengelighet til undersøkelse	Både admin og researcher brukere skal kunne endre tilgjengeligheten til en undersøkelse (fra privat til åpen f.eks.). En åpen undersøkelse deler ikke data fra svarene som kommer inn, men man får lesetilgang til spørsmålene.	15. 27
Se statistikk for undersøkelse	Både admin og researcher brukere skal kunne se statistikk for undersøkelser som de har rettigheter til (eller som er åpne). F.eks. antall svar, antall dommere og lignende.	26, 30
Hente ut rådata	Både admin og researcher brukere skal kunne hente ut rådata	12

	(svarene) for en undersøkelse som de har rettigheter til. Dette kan lastes ned som .CSV eller .JSON.	
Få analyse av undersøkelse	Både admin og researcher brukere skal kunne be om analyse av resultatene fra undersøkelsen og få vist fram dette. F.eks. reliabilitet, lineær rangering, forholdet mellom objektene ol.	11, 24, 25, 31
Vis eksisterende undersøkelser	Admin må kunne se alle undersøkelser for å kunne slette. Researcher skal bare få opp undersøkelsene de har rettigheter til. Det er mulig å filtrere/søke og sortere listen.	5, 18, 21, 23, 26
Generere delbar link og pin-kode for undersøkelse	Eieren av en undersøkelse må kunne hente ut en lenke som kan deles med dommerne slik at de kan svare på undersøkelsen. Her tenkes det to forskjellige typer lenker, en der man kommer rett til undersøkelsen, og en som i tillegg krever en aksesskode	6, 16
Svare på undersøkelse	Dommerne må kunne utføre undersøkelsen. Dette innebærer å gi dommeren to og to valg, med muligheten til å velge en av disse. (Knapp, piltast eller lignende). Dette skjer fram til bestemt antall sammenligninger er utført, eller dommeren lukker undersøkelsen.	10, 17

6. Ikke-funksjonelle egenskaper og andre krav

Krav	**Beskrivelse**
God MMI	Det legges vekt på at brukeropplevelsen er i fokus. Her viktigst at det er enkelt å svare på undersøkelsen, etterfulgt av erfaringen til researcher brukerne, og til slutt administratoren. Disse kravene skal testes ved hjelp av brukertester underveis i prosjektet.
Ha et universelt utformet produkt	Det er viktig at så mange som mulig skal kunne bruke produktet vårt uten å slite med brukergrensesnittet. I hovedsak skal vi fokusere på å sikre brukbarhet for fargeblinde og svaksynte, ved hjelp av blant annet lesbar skrift med god kontrast. Vi vil gjerne og at nettsiden skal være lett for skjermleser/“tekst til tale”-software å lese. Vi vil gjerne og at nettsiden skal være mulig å navigere uten bruk av mus. Dette inkluderer å kunne bruke tab for å hoppe fra element til element, og å kunne bruker enter i stedet for å trykke på en knapp.
Flerspråklig	Nettsiden lages i første omgang med engelsk som språk, men det er ønskelig å kunne internasjonalisere.
Sikkerhet	Systemet må støtte HTTPS, sikker lagring av brukerinformasjon (hash passord + salt)

Web-plattform for Adaptive Comparative Judgement

Kravdokumentasjon

Versjon 1.1

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
19/02/21	1.0	Lagt til user stories	Lasse Seivaag
23/04/21	1.1	Gjort endringer i user stories basert på visjonsdokumentet og tidligere møter med produkteier. Fått aksept av Hermund på teams.	Lasse Seivaag, Jørgen Steig

Innholdsfortegnelse

1. Introduksjon

Dette dokumentet er skrevet for å eksemplifisere behovene og kravene for funksjonalitet fra visjonsdokumentet for bacheloroppgave 064 Web-plattform for Adaptive Comparative Judgement. Vi har laget user stories for å vise behovene og for å sette akseptansekrav tilknyttet disse.

2. User Stories

Det er laget user stories for alle funksjonene i visjonsdokumentet, men det betyr ikke at alle disse må være oppfylt for at produktet skal anses som en suksess. Se visjonsdokument kapittel 3.4 og kapittel 5 for en prioritert liste med ønsket funksjonalitet. Kravene med prioritet 1 må være oppfylt for at produktet skal anses som en suksess sett i forhold til oppdragsbeskrivelsen. Dette kapitlet er benyttet for å sette behovet for funksjonalitet i perspektiv til en bruker, og får å få mer konkrete krav for aksept av funksjonaliteten.

2.1 Registrere bruker

Som administrator

Ønsker jeg å kunne registrere brukere

Slik at forskere kan bruke systemet

Akseptansekriterier:

- Det finnes en administrator bruker til å begynne med
- Som administrator har jeg mulighet til å se og navigere til “opprett bruker” siden
- Jeg får mulighet til å fylle inn epost-adresse og type bruker
- Det blir validert at epost-adressen er på rett format
- Det er klar feedback om prosessen gikk bra eller dårlig
- Mail med lenke for å opprette bruker sendes til den angitte epostadressen
- En forsker bruker skal ikke kunne registrere brukere

2.2 Logge inn

Som bruker

Ønsker jeg å kunne logge inn på tjenesten

For å kunne få oversikt over mine undersøkelser, opprette nye undersøkelser og analysere undersøkelsene.

Akseptansekriterier:

- Det er et tydelig menyvalg på siden som kan navigere meg til siden hvor jeg logger inn
- Jeg kan skrive inn epost-adresse og passord
- Klar feedback på at man blir logget inn, eller om noe gikk galt.

2.3 Endre passord

Som bruker

Ønsker jeg å kunne endre passord

Slik at jeg kan opprettholde sikkerheten på brukeren min

Akseptansekriterier:

- Gitt at jeg kan logge inn
- Det skal være et tydelig menyvalg jeg kan navigere til
- Jeg kan fylle inn nåværende passord, nytt passord og gjentatt nytt passord
- Input valideres
- Tydelig feedback etter validering, og ved endring av passord om dette gikk bra

2.4 Glemt passord

Som bruker

Ønsker jeg å kunne få nytt passord uten å huske det gamle

For å få tilgang til kontoen min selv om jeg har glemt passordet mitt

Akseptansekriterier:

- Det skal være et tydelig hvor jeg skal navigere (Log in -> Forgotten Password)
- Jeg kan fylle inn eposten jeg har registrert bruker på
- Jeg blir tilsendt en mail med lenke hvor jeg kan sette nytt passord
- Jeg fyller inn eposten min, nytt passord og repetert passord
- Tydelig feedback etter validering, og ved endring av passord om dette gikk bra

2.5 Slette bruker

Som bruker av systemet

Ønsker jeg å kunne slette min brukerkonto

Slik at jeg ikke lengre er registrert på tjenesten

Akseptansekrav:

- Jeg kan navigere til en side hvor jeg kan slette kontoen min (Account->Delete Account)
- Jeg må bekrefte at jeg ønsker å slette kontoen for å unngå sletting ved uhell
- Tilhørende data (undersøkelser) slettes ikke når brukeren slettes

2.6 List ut brukere

Som administrator

Ønsker jeg å kunne se en liste over brukere

Slik at jeg enkelt kan få oversikt over hvem som er i systemet, og har en plass hvor jeg kan administrere (slette) brukerne

Akseptansekrav:

- Som administrator skal det være mulig å navigere til en side der en liste over brukere vises
- Listen må inneholde funksjonalitet for å slette en bruker
- Når en bruker slettes fra listen må den oppdateres automatisk
- Det finnes en sjekkboks for om man vil slette tilhørende data (undersøkelser med svar)
- Bare administrator skal ha tilgang til denne listen og funksjonaliteten

2.7 Opprette undersøkelse

Som bruker av systemet

Ønsker jeg å kunne opprette undersøkelser

Slik at jeg kan få svar på spørsmålene jeg utforsker

Akseptansekrav:

- Både administratorer og forsker brukere må kunne opprette undersøkelser
- Jeg kan navigere til "opprett undersøkelse"
- Jeg blir bedt om å fylle inn tittel, intern beskrivelse, dommer veiledning, antall sammenligninger en dommer kan forvente å svare på, og å legge til alle svaralternativene jeg ønsker
- Når jeg legger til svaralternativer må jeg ha mulighet til å velge hvilken mediatype alternativet er (ren tekst, pdf etc.)
- Jeg får mulighet til å legge til medforskere som skal ha tilgang til undersøkelsen
- Jeg får satt rettigheter på medforskere jeg legger til
- Jeg kan huke av for om undersøkelsen er aktiv eller inaktiv. En inaktiv undersøkelse kan ikke besvares.
- Input valideres og gir klar feedback om noe ikke stemmer
- Når undersøkelsen er validert og lagret blir jeg tatt med til oversikt over mine undersøkelser
- Tittel er ikke synlig til dommer
- Survey question er synlig for dommer
- Dommerveiledning vises på førstesiden når man starter testen
- Det skal være mulig å lage en kopi av undersøkelsen

2.8 Redigere undersøkelse

Som bruker av systemet

Ønsker jeg å kunne redigere mine undersøkelser

Slik at jeg kan legge til eller endre spørsmål, rette opp i feil eller gi medforskere tilgang til

den.

Akseptansekrav:

- Både administrator og forsker/medforsker med korrekt rettighet i undersøkelsen skal kunne redigere den
- Alt skal kunne redigeres, svaralternativer, tittel, spørsmål, dommer-veiledning, intern beskrivelse og antall sammenligninger per dommer
- Om man endrer noe som kan invalidere resultatene i undersøkelsen skal dette gis advarsel om, men det er til syvende og sist forskerens ansvar om dette skjer
- Svar som allerede har blitt gitt som inkluderer et slettet svaralternativ skal slettes automatisk
- Input valideres og det gis klart feedback om noe ikke stemmer

2.9 Slette Undersøkelse

Som administrator

Ønsker jeg å kunne slette undersøkelser

For å fjerne uegnet innhold eller rydde opp i dataen som er lagret i tjenesten

Akseptansekrav:

- Admin kan slette alle undersøkelser
- En forsker kan forlate sine egne undersøkelser, men ikke faktisk slette de
- Sletter man undersøkelsen så slettes også svarene på undersøkelsen

2.10 Legge til medforskere på undersøkelser

Som bruker av systemet

Ønsker jeg å kunne legge til medforskere på undersøkelser

Slik at vi kan samarbeide eller validere hverandres arbeid

Akseptansekrav:

- Når man oppretter eller redigerer en undersøkelse skal det være en klart markert seksjon hvor man kan legge til medforskere

- Medforkserne kan bli gitt forskjellige rettigheter (implisitt se undersøkelsen), redigere undersøkelsen, se resultater, og administrere medforskere. (administrer medforskere gir brukeren tilsvarende "eier" rettigheter, og man kan fjerne originale eier)
- Man kan søke opp brukere med navn eller epost
- Det går an å fjerne medforskere om man har rettighet til det (administrer medforskere)

2.11 Endre tilgjengelighet av undersøkelse

Som bruker av systemet

Ønsker jeg å kunne endre tilgjengeligheten av en undersøkelse

Slik at andre registrerte brukere kan invitere dommere til å svare på undersøkelsen

Akseptansekrav:

- Det må komme klart fram at om man gjør undersøkelsen public så vil alle registrerte brukere kunne se undersøkelsen, og invitere dommere til den.
- Det er fremdeles bare eierne av undersøkelsen som får tilgang til statistikk, analyse og rådata for undersøkelsen

2.12 Se statistikk for undersøkelse

Som bruker av systemet

Ønsker jeg å kunne se statistikk for undersøkelsene mine

Slik at jeg får bedre innsikt i framdriften av undersøkelsen, og lettere kan velge om jeg har fått nok besvarelser, om en dommers svar anses å være brukbar og om jeg har noen spørsmål som kunne vært fjernet

Akseptansekrav:

- Det skal kreve få klikk for å navigere til statistikken
- Det skal være mulig å endre visningen av statistikken i forhold til alle svar fra en dommer, alle svar (på spørsmål), og totalt sett for undersøkelsen

2.13 Hente ut rådata

Som bruker av systemet

Ønsker jeg å kunne hente ut rådata (svar på undersøkelsene)

For å kunne bruke dataen i eksterne systemer, eller for å ta backup.

Akseptansekrav:

- Jeg må kunne navigere til en side hvor dataen blir presentert (Liste over surveys -> View -> raw data)
- Det må være en knapp for å laste ned dataen i form av .CSV eller .JSON filer
- Ønskelig format er type: surveyId, judgeId, leftItem, rightItem, winner

2.14 Få analyse av undersøkelse

Som forsker i systemet

Ønsker jeg å kunne få analyse av svarene på undersøkelsen min

Slik at jeg kan få oversikt over hvordan de forskjellige svaralternativene gjør det, og i hvilken grad jeg kan stole på resultatene.

Akseptansekrav:

- En forsker skal kunne navigere til en side hvor analysen blir presentert (Liste over surveys -> View -> Linear ranking/By judge/By item
- Analysen må innebære et theta-parameter som angir rangeringen av svaralternativene
- Analysen må innebære en reliabilitetskonstant som angir i hvilken grad man kan stole på rangeringen.

2.15 List ut undersøkelser

Som forsker/admin i systemet

Ønsker jeg å kunne få oversikt over undersøkelsene mine

Slik at jeg enkelt kan se på statistikk, samt endre, slette og dele undersøkelsen.

Akseptansekrav:

- Admin bruker skal lett kunne søke/filtrere etter undersøkelser uten eiere
- Fra listen skal man kunne hente kode for å la dommere delta i undersøkelsen, laste ned rådata, slette undersøkelsen, redigere undersøkelsen og trykke seg inn på undersøkelsen for å få tilgang til statistikk og analyse

2.16 Dele undersøkelse med dommere

Som bruker av systemet

Ønsker jeg å kunne enkelt dele en undersøkelse med andre

Slik at det blir lettest mulig for dommere å begynne å ta undersøkelsen, så jeg får så mange respondenter som mulig.

Akseptansekrav:

- En forsker skal enkelt få tilgang til en fungerende lenke til undersøkelsen
- Det genereres en unik PIN-kode som en dommer kan bruke på forsiden for å delta i undersøkelsen

2.17 Svar på undersøkelse

Som deltaker i undersøkelsen

Ønsker jeg å enkelt kunne svare på en undersøkelse

Slik at jeg kan bidra i forskningen så fort og effektivt som mulig

Akseptansekrav:

- Får tilgang til å svare på undersøkelsen enten ved direkte lenke, eller med pinkode på forsiden
- Det er mulig å besvare undersøkelsen på bestemte nettbrett og desktop/laptop

Web-plattform for Adaptive Comparative Judgement

Systemdokumentasjon

Versjon 1.0

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
13.05.2021	1.0	Ferdigstilt systemdokumentasjon etter mal	Lasse Seivaag, Jørgen Steig

Innholdsfortegnelse

Introduksjon

Dette dokumentet er skrevet for å dokumentere systemet som ble opprettet som følge av Bacheloroppgave 064 i faget TDAT3001 ved NTNU. Dokumentets hensikt er å gi en overordnet oversikt over alle delene av systemet som ble utviklet, slik at det blir lett for produkteier å drifte og videreutvikle produktet i etterkant.

1. Arkitektur

Plattformen består av fire deler:

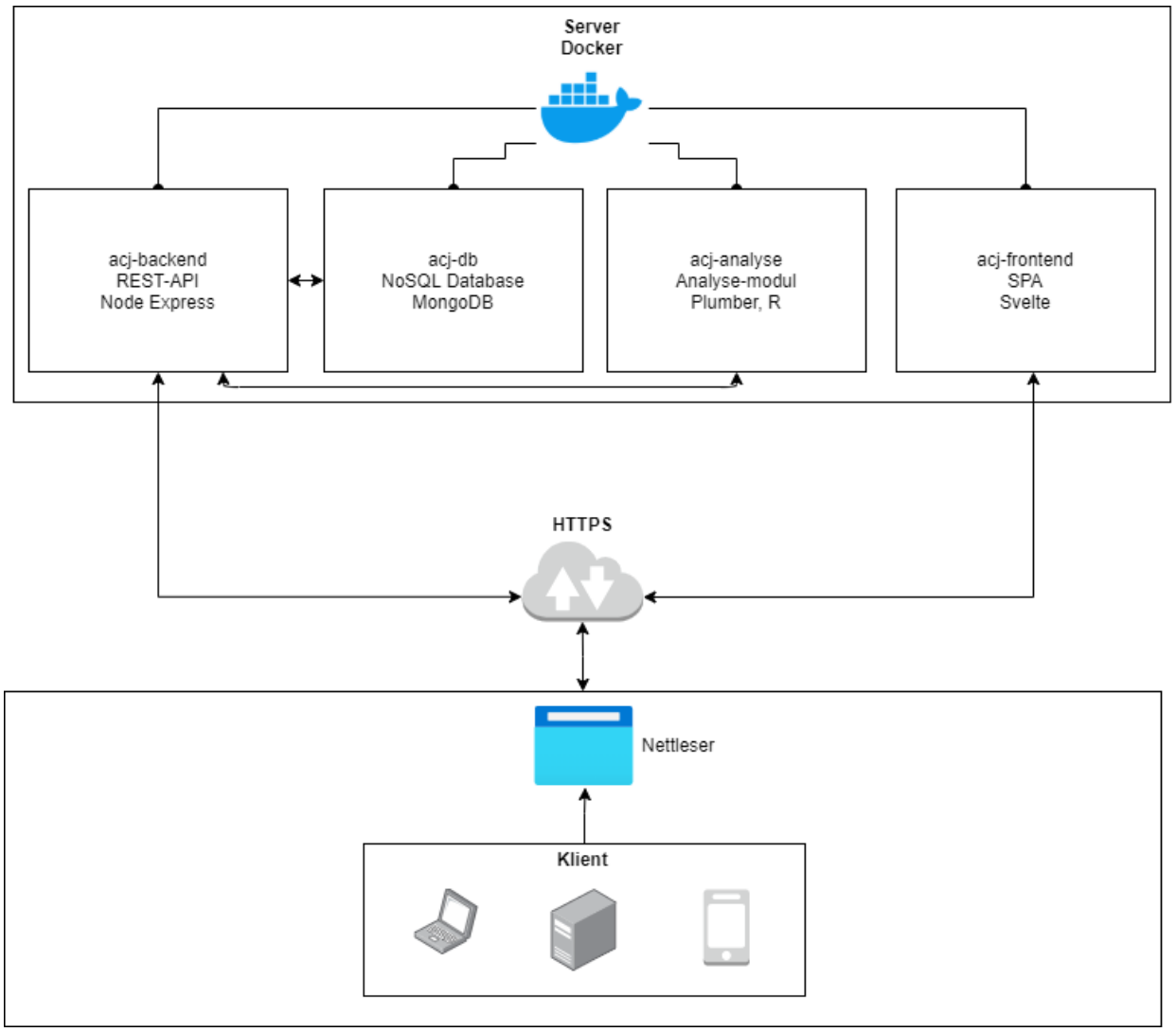
- Frontend (nettside)
- Backend (REST-API)
- Database
- Analyse-modul

Frontend er utviklet med JavaScript, CSS, HTML5, og Svelte. Det benyttes også diverse npm pakker.

Backend er utviklet med JavaScript, NodeJS, Express.js med diverse middleware og Mongoose som databasedriver.

Databasesystemet som blir brukt er MongoDB, som er en dokument-basert NoSQL-database.

Analyse-modulen er laget av Tore Alexander Forbregd og er skrevet i R.



I figuren over ser vi hvordan systemet er tenkt brukt og henger sammen. Klienten snakker med acj-frontend og blir servert en nettside. Nettsiden er bygd for å snakke med acj-backend, som igjen snakker med acj-db og acj-analyse på vegne av klienten. Det brukes HTTPS mellom frontend, backend og klient. Alle fire tjenestene som systemet består av kjøres i separate containere i et Docker miljø på serveren.

1.1 Porter

Portene som er brukt uten modifikasjoner i konfigurasjonen:

acj-backend: 5000

acj-db: 27017

acj-analyse: 1030

acj-frontend: 443

Det er bare acj-backend og acj-frontend som er åpen for trafikk utenfor Docker miljøet.

2. Prosjektstruktur

Filstrukturen er bygd opp i moduler. På toppnivå har vi to frittstående filer "**docker-compose.yml**" som brukes for å konfigurere containeriseringen av modulene, og "**build.sh**" som abstraherer docker-compose kommandoer for å bygge og starte containerne. Videre har vi egne mapper for modulene:

/AnalyseModule

/client

/mongodb

/server

Felles for alle disse er at de har en "**Dockerfile**" fil for å bygge containere.

2.1 AnalyseModule

Består bare av "**estimate.r**" som er skrevet av produkteier Tore Alexander Forbregd.

2.2 Client

Denne modulen er en webserver som serverer en Single-Page-Application nettside til klienter. På toppnivå har vi konfigurasjonsfiler:

/package.json → Viktigste konfigurasjon her er "prod" verdi, denne må inkludere sti til gyldig sertifikat og nøkkel for å kunne bruke HTTPS.

/rollup.config.js → Viktigste konfigurasjon her er "apiBasePath" verdi som brukes i alle REST-API kall.

/public/ → Denne mappen blir servert til klienter. Legg statisk innhold til nettsiden her. For eksempel bilder i /public/img/. Det kompilerte svelte prosjektet blir også plassert her, under **public/build/bundle.js, bundle.js.map og bundle.css**.

/src/ → Kildekode for nettsiden. Inneholder sub-mapper som logisk splitter opp nettsiden

/src/Components/ → Gjenbrukbare komponenter slik som Navbar og Table

/src/Components/Menu.svelte → Menu brukes på alle sub-sider av admin_board

/src/Components/Navbar.svelte → Navigasjonsbar brukes på alle sider

/src/Components/Table.svelte → Brukes for å framvise data tabellarisk (Surveys, Users, RawSurveyData for eksempel)

/src/Components/TableFilter.svelte → Filter/Sorter komponent som brukes sammen med table.svelte
/src/Components/SurveyComponents/ → Komponenter spesifikt knyttet til undersøkelser
/src/Components/SurveyComponents/PDFItem.svelte → Brukes for å lage et Item med type PDF i en undersøkelse
/src/Components/SurveyComponents/PDFView.svelte → Brukes for å vise fram en PDF
/src/Components/SurveyComponents/TextItem.svelte → Brukes for å lage et Item med type plain (rå tekst)
/src/Components/SurveyComponents/TextView.svelte → Brukes for å vise fram plain-text
/src/Pages/ → "Sider" eller "Views" av nettsiden. Landing Page, LoginPage og lignende.
/src/Pages/AboutProject.svelte → "About" siden, forklarer bakgrunn for prosjektet og CJ
/src/Pages/AdminBoard.svelte → Hovedsiden for en innlogget bruker, tilbyr en ekstra meny med undersider som brukere kan benytte seg av
/src/Pages/App.svelte → Fungerer som "index.html", entrypoint for nettsiden. Toppnivået i web-appen
/src/Pages/CreateSurvey.svelte → Side for å opprette eller redigere en undersøkelse
/src/Pages/ForgottenPassword.svelte → Side hvor brukere kan få en lenke for å resette passordet sitt, brukes også for å sette nytt passord ved bruk av lenken
/src/Pages/IntroductionToSurvey.svelte → Brukes som overlay i Survey.svelte, informasjonsside hvor spørsmålet i undersøkelsen vises, instruksjoner fra forsker og generell info om hvordan man tar undersøkelsen.
/src/Pages/InviteResearcher.svelte → Side som bare administrator har tilgang til, bruker for å invitere nye brukere
/src/Pages/Landing_page.svelte → Forsiden på nettsiden. En dommer kan skrive inn PIN kode her for å delta i en undersøkelse.
/src/Pages/LoginPage.svelte → Side for innlogging
/src/Pages/Profile.svelte → En side hvor en bruker kan se hvor mange undersøkelser de har tilgang til, funksjonalitet for å endre passord, og slette sin konto
/src/Pages/RawSurveyData.svelte → Side for å se statistikk og analyse, samt laste ned disse - for en undersøkelse
/src/Pages/RegisterAccount.svelte → Side hvor en invitert bruker kan opprette sin konto
/src/Pages/Researchers.svelte → Bare tilgjengelig for admin, lister opp registrerte brukere med funksjonalitet for å slette disse (med eller uten tilhørende undersøkelsesdata)
/src/Pages/Survey.svelte → Siden hvor dommere svarer på en undersøkelse

/src/Pages/Surveys.svelte → Side for å liste ut undersøkelser man har tilgang til. Har lenker og funksjonalitet for å se statistikk/analyse, redigere, dele lenke til dommere og slette.
/src/Services/ → Inneholder filer for funksjonalitet knyttet til kommunikasjon mot REST-API,
/src/Services/UserService.js → Funksjoner som gjør HTTP kall mot User ressursen
/src/Services/SurveyService.js → Funksjoner som gjør HTTP kall mot Survey ressursen
/src/Services/SurveyAnswerService.js → Funksjoner som gjør HTTP kall mot SurveyAnswer ressursen
/src/Services/SurveyItemFileService.js → Funksjoner som gjør HTTP kall mot SurveyItemFile ressursen
/src/Utility/ → Inneholder filer for gjenbrukbar logikk
/src/Utility/dateFromObjectId.js → Funksjon for å hente Date ut fra en ObjectId
/src/Utility/nodeBufferToBlobURL.js → Funksjoner for å gå fra en Node Buffer type til BLOB URL eller et File objekt
/src/Utility/pdf-js/ → Mozilla pdf-js, 3. parts bibliotek for å vise fram PDF filer

2.3 Mongodb

/default.env → brukes som mal for å konfigurere databasen under installasjon.

2.4 Server

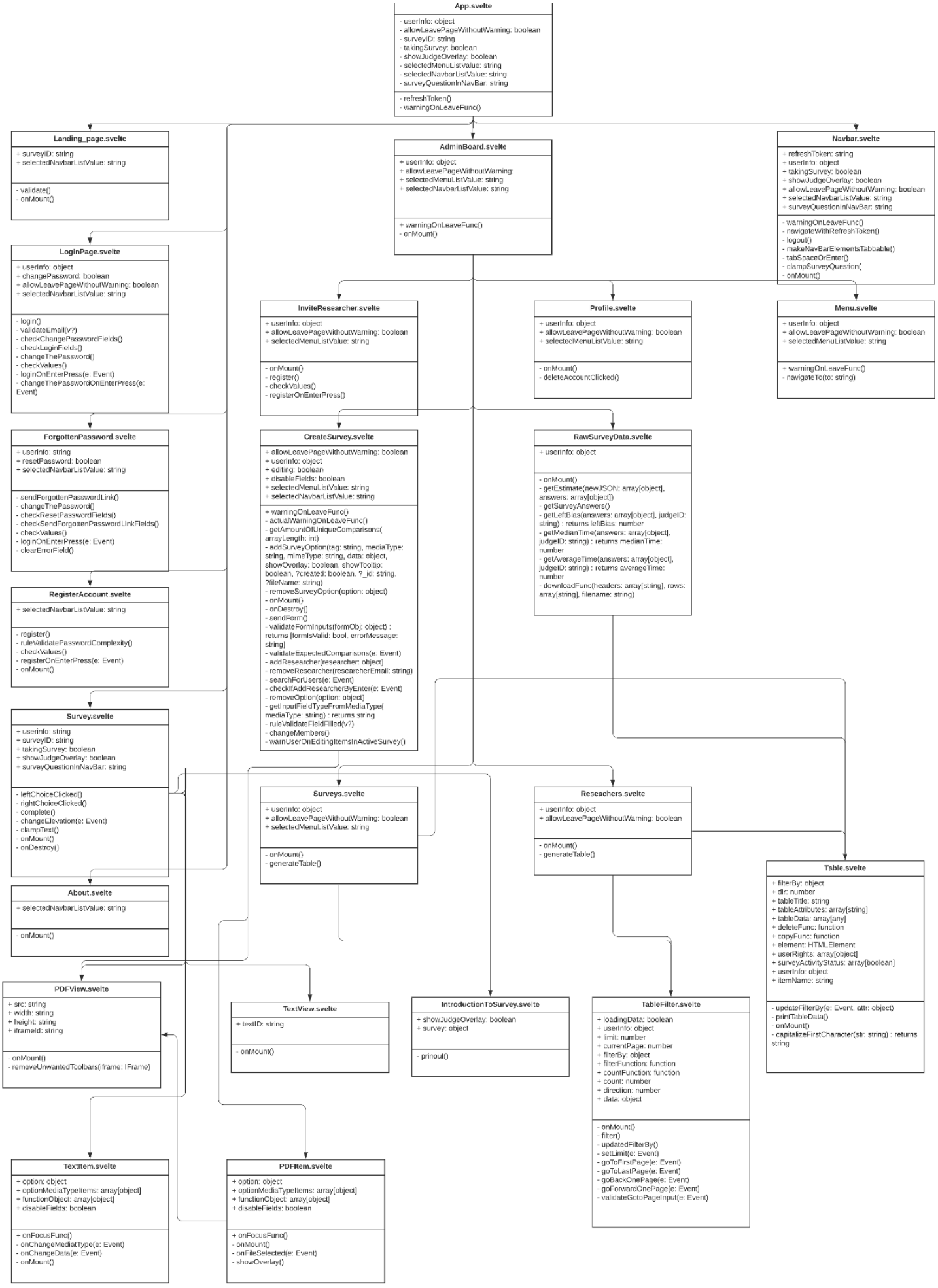
Server modulen er et REST-API og er knutepunktet for analyse-modulen, databasen og klienten.
På toppnivået har vi konfigurasjonsfilene og entrypoint for serveren:
/apidoc.json → Innstillinger for apidoc (generering av dokumentasjon for REST-API)
/apidoc_globals.js → Inneholder kommentar-blokker som brukes flere plasser i kildekoden
/default.env → En miljøvariabl fil man bruker som mal ved installasjon
/package.json → Konfigurasjonsfil for node og npm, burde ikke trenge å gjøre manuelle endringer her
/server.js → Entrypoint for serveren, inneholder noen ting som konfigureres: maks filstørrelse ved filopplastning og godkjente domener for CORS
/Utility/ → Inneholder filer med gjenbrukbar logikk
/Utility/hashing.js → Funksjoner for å hashe, og sammenligne hasher

/Utility/mail.js → Funksjon for å sende mail
/Utility/passwordRequirement.js → Brukes for å sjekke passordstyrken, må samsvare med client sin versjon
/models/ → Inneholder modeller/schemaer for database entiteter
/models/Survey.js → Schema og instans av modell for Survey entiteten
/models/SurveyAnswer.js → Schema og instans av modell for SurveyAnswer entiteten
/models/SurveyFileItem.js → Schema og instans av modell for SurveyFileItem entiteten (fildata og metadata for items fra survey)
/models/User.js → Schema og instans av modell for User entiteten
/routes/ → Inneholder ruter for REST-APIet
/routes/authentication.js → Brukes for login, logout og håndtering av access/judge tokens. Inneholder også en middleware funksjon "auth"
/routes/survey_answer_route.js → Ruter for å manipulere SurveyAnswer ressursen
/routes/survey_item_file_route.js → Ruter for å manipulere SurveyItemFile ressursen
/routes/survey_route.js → Ruter for å manipulere Survey ressursen
/routes/user_route.js → Ruter for å manipulere User ressursen

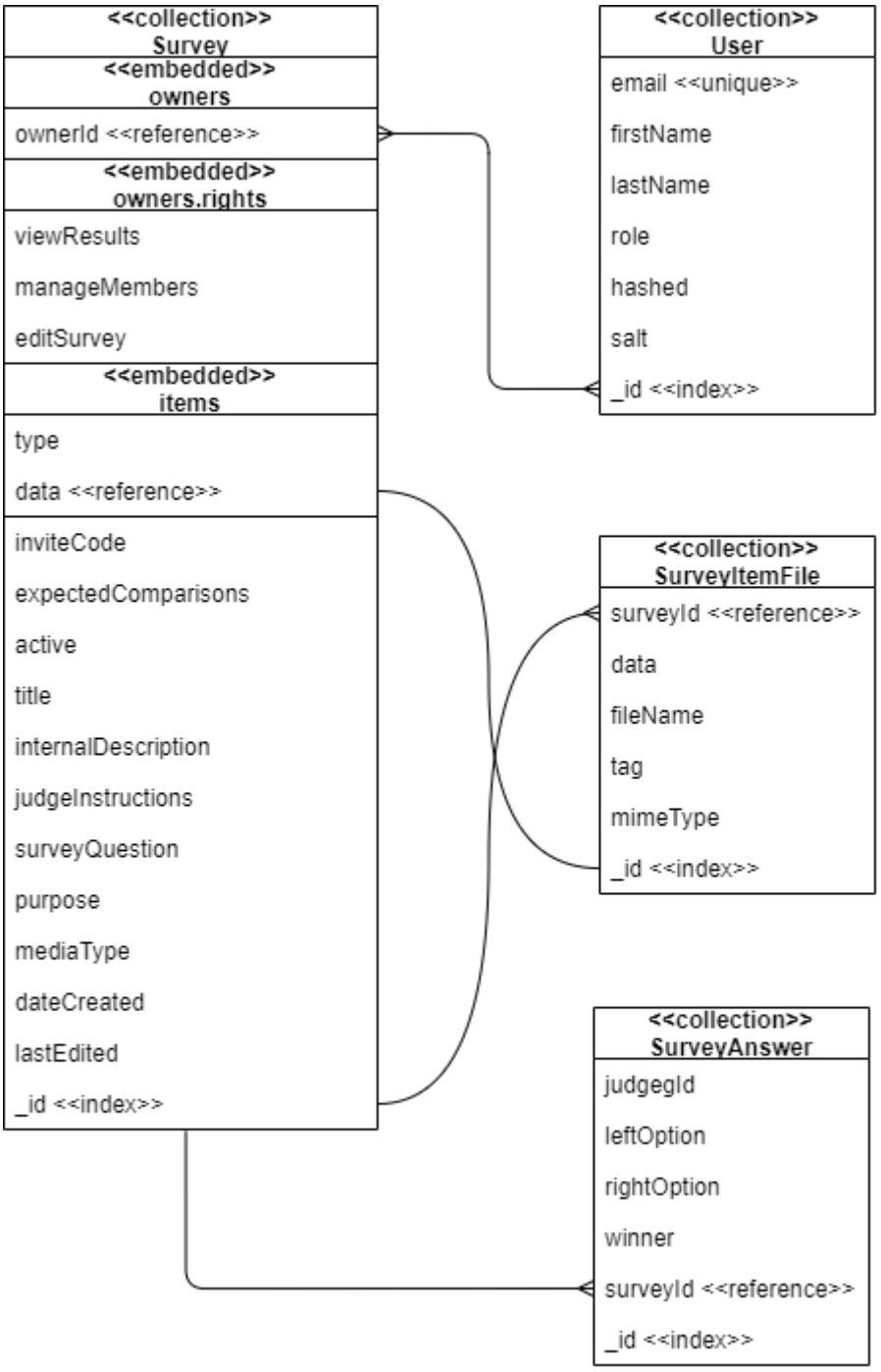
3. Komponentdiagram

Komponentdiagrammet visualiserer komponentene som bygger opp frontend i prosjektet. Tittelen på hver boks er filnavnet til komponenten. Den øverste raden viser alle variabler som kan bestemmes av foreldre til komponenten, samt deres datatyper. Disse er alle deklarert med syntaksen "export let variabelnavn" i komponenten. Den nederste raden inneholder alle de navngitte funksjonene i komponenten, samt eventuelle parametere og deres datatyper. Dersom funksjonen returnerer noe vil datatypen til det som blir returnert også være inkludert.

Relasjonspilene mellom komponentene representer forelder-barn-forholdet deres. Pilene vil peke fra forelder og ned til barn. Med barn menes det her at komponenten er innrammet i en annen komponent. En forelder kan eksistere (vises på nettsiden) uten et barn, men et barn kan ikke eksistere (vises på nettsiden) uten at forelderen også vises. Pilene går bare mellom direkte forelder-barn forhold, så dersom en forelder har et barn som igjen har flere barn under seg, vil det ikke gå en pil mellom forelder og "barnebarn".


App.svelte
- userInfo: object
- allowLeavePageWithoutWarning: boolean
- surveyID: string
- takingSurvey: boolean
- showJudgeOverlay: boolean
- selectedMenuListValue: string
- selectedNavbarListValue: string
- surveyQuestionInNavBar: string
- refreshToken()
- warningOnLeaveFunc()
Landing_page.svelte
+ surveyID: string
+ selectedNavbarListValue: string
- validate()
- onMount()
AdminBoard.svelte
+ userInfo: object
+ allowLeavePageWithoutWarning:
+ selectedMenuListValue: string
+ selectedNavbarListValue: string
+ warningOnLeaveFunc()
- onMount()
Navbar.svelte
+ refreshToken: string
+ userInfo: object
+ takingSurvey: boolean
+ showJudgeOverlay: boolean
+ allowLeavePageWithoutWarning: boolean
+ selectedNavbarListValue: string
+ surveyQuestionInNavBar: string
- warningOnLeaveFunc()
- navigateWithRefreshToken()
- logout()
- makeNavBarElementsTabbable()
- tabSpaceOrEnter()
- clampSurveyQuestion(
- onMount()
LoginPage.svelte
+ userInfo: object
+ changePassword: boolean
+ allowLeavePageWithoutWarning: boolean
+ selectedNavbarListValue: string
- login()
- validateEmail(v?)
- checkChangePasswordFields()
- checkLoginFields()
- changeThePassword()
- checkValues()
- loginOnEnterPress(e: Event)
- changeThePasswordOnEnterPress(e: Event)
InviteResearcher.svelte
+ userInfo: object
+ allowLeavePageWithoutWarning: boolean
+ selectedMenuListValue: string
- onMount()
- register()
- checkValues()
- registerOnEnterPress()
Profile.svelte
+ userInfo: object
+ allowLeavePageWithoutWarning: boolean
+ selectedMenuListValue: string
- onMount()
- deleteAccountClicked()
Menu.svelte
+ userInfo: object
+ allowLeavePageWithoutWarning: boolean
+ selectedMenuListValue: string
+ warningOnLeaveFunc()
- navigateTo(to: string)
ForgottenPassword.svelte
+ userinfo: string
+ resetPassword: boolean
+ selectedNavbarListValue: string
- sendForgottenPasswordLink()
- changeThePassword()
- checkResetPasswordFields()
- checkSendForgottenPasswordLinkFields()
- checkValues()
- loginOnEnterPress(e: Event)
- clearErrorField()
CreateSurvey.svelte
+ allowLeavePageWithoutWarning: boolean
+ userInfo: object
+ editing: boolean
+ disableFields: boolean
+ selectedMenuListValue: string
+ selectedNavbarListValue: string
+ warningOnLeaveFunc()
- actualWarningOnLeaveFunc()
- getAmountOfUniqueComparisons(arrayLength: int)
- addSurveyOption(tag: string, mediaType: string, mimeType: string, data: object, showOverlay: boolean, showTooltip: boolean, ?created: boolean, ?_id: string, ?fileName: string)
- removeSurveyOption(option: object)
- onMount()
- onDestroy()
- sendForm()
- validateFormInputs(formObj: object) : returns [formIsValid: bool, errorMessage: string]
- validateExpectedComparisons(e: Event)
- addResearcher(researcher: object)
- removeResearcher(researcherEmail: string)
- searchForUsers(e: Event)
- checkIfAddResearcherByEnter(e: Event)
- removeOption(option: object)
- getInputFieldTypeFromMediaType(mediaType: string) : returns string
- ruleValidateFieldFilled(v?)
- changeMembers()
- warnUserOnEditingItemsInActiveSurvey()
RawSurveyData.svelte
+ userInfo: object
- onMount()
- getEstimate(newJSON: array[object], answers: array[object])
- getSurveyAnswers()
- getLeftBias(answers: array[object], judgeID: string) : returns leftBias: number
- getMedianTime(answers: array[object], judgeID: string) : returns medianTime: number
- getAverageTime(answers: array[object], judgeID: string) : returns averageTime: number
- downloadFunc(headers: array[string], rows: array[string], filename: string)
RegisterAccount.svelte
+ selectedNavbarListValue: string
- register()
- ruleValidatePasswordComplexity()
- checkValues()
- registerOnEnterPress(e: Event)
- onMount()
Survey.svelte
+ userinfo: string
+ surveyID: string
+ takingSurvey: boolean
+ showJudgeOverlay: boolean
+ surveyQuestionInNavBar: string
- leftChoiceClicked()
- rightChoiceClicked()
- complete()
- changeElevation(e: Event)
- clampText()
- onMount()
- onDestroy()
Surveys.svelte
+ userInfo: object
+ allowLeavePageWithoutWarning: boolean
+ selectedMenuListValue: string
- onMount()
- generateTable()
Reseachers.svelte
+ userInfo: object
+ allowLeavePageWithoutWarning: boolean
- onMount()
- generateTable()
Table.svelte
+ filterBy: object
+ dir: number
+ tableTitle: string
+ tableAttributes: array[string]
+ tableData: array[any]
+ deleteFunc: function
+ copyFunc: function
+ element: HTMLElement
+ userRights: array[object]
+ surveyActivityStatus: array[boolean]
+ userInfo: object
+ itemName: string
- updateFilterBy(e: Event, attr: object)
- printTableData()
- onMount()
- capitalizeFirstCharacter(str: string) : returns string
About.svelte
+ selectedNavbarListValue: string
- onMount()
PDFView.svelte
+ src: string
+ width: string
+ height: string
+ iframeId: string
- onMount()
- removeUnwantedToolbars(iframe: IFrame)
TextView.svelte
+ textID: string
- onMount()
IntroductionToSurvey.svelte
+ showJudgeOverlay: boolean
+ survey: object
- printout()
TableFilter.svelte
+ loadingData: boolean
+ userInfo: object
+ limit: number
+ currentPage: number
+ filterBy: object
+ filterFunction: function
+ countFunction: function
+ count: number
+ direction: number
+ data: object
- onMount()
- filter()
- updatedFilterBy()
- setLimit(e: Event)
- goToFirstPage(e: Event)
- goToLastPage(e: Event)
- goBackOnePage(e: Event)
- goForwardOnePage(e: Event)
- validateGotoPageInput(e: Event)
TextItem.svelte
+ option: object
+ optionMediaTypeItems: array[object]
+ functionObject: array[object]
+ disableFields: boolean
+ onFocusFunc()
- onChangeMediaType(e: Event)
- onChangeData(e: Event)
- onMount()
PDFItem.svelte
+ option: object
+ optionMediaTypeItems: array[object]
+ functionObject: array[object]
+ disableFields: boolean
+ onFocusFunc()
- onMount()
- onFileSelected(e: Event)
- showOverlay()

4. Databasemodell



Databasen er designet med fire entiteter: User, Survey, SurveyAnswer og SurveyItemFile. En undersøkelse kan ha flere brukere med tilgang, og en bruker kan ha tilgang til flere undersøkelser. En undersøkelse kan ha flere svar, men et svar (SurveyAnswer) hører bare til en undersøkelse. En undersøkelse kan også ha flere filer som tilhører svaralternativene; men ett “items” dokument samsvarer bare med et “SurveyItemFile” dokument.

5. Server-tjenester

Det er en tjeneste som er nåbar fra internett utenfor webtjeneren som serverer nettsiden. Dette er et REST-API som også håndterer kommunikasjon med databasen og analyse-modulen på vegne av klienten.

5.1 REST-API

API-et serveres med node express og har fire ressurser, samt løsning for autorisasjon og autentisering. Fire av de fem ressursene samsvarer med database-entitene. Alle rutene er sikret med JWT basert autentisering som krever autorisering, med unntak av noen ruter for "dommere" som er anonyme brukere som svarer på undersøkelsene. Rutene filtrerer også resultatene fra databasen slik at f.eks. "User" hash og salt ikke sendes til klienten.

5.1.1 Survey

Denne ressursen brukes for å håndtere "Survey" entiteten og kontaktes på base-url/api/survey. Vi støtter CRUD operasjoner, og noen spesielle ruter som f.eks. /function/estimate som gjør videre kall mot analyse-modulen og returnerer svar til klienten, /function/sort og /function/search som gir klienten mulighet til å sortere og søke med diverse parametere f.eks. felt for sortering, retning for sortering og antall dokumenter man vil hente ut.

5.1.2 SurveyItemFile

Denne ressursen brukes for å håndtere "SurveyItemFile" entiteten og kontaktes på base-url/api/surveyitemfile.
Vi støtter CRUD operasjoner, og en rute for å hente informasjon utenom selve dataen /:id/view

5.1.3 SurveyAnswer

Denne ressursen brukes for å håndtere "SurveyAnswer" entiteten og kontaktes på base-url/api/surveyanswer.
Vi Støtter CRUD operasjoner samt /function/count/answers/:surveyId og /function/count/judges/:surveyId som svarer med antall svar og dommere som tilhører en gitt undersøkelse, respektivt.

5.1.4 User

Denne ressursen brukes for å håndtere "User" entiteten og kontakes på base-rul/api/user. Vi støtter CRUD operasjoner samt søk, sortering og "count" på respektivt /search/:term, /function/sort og /function/count. Det å opprette og endre en bruker (passord spesifikt) har flere steg og flere ruter. En administrator kan kalle /invite_link for å invitere en bruker, og den brukeren må da kalle post / for å faktisk opprette brukeren sin. For endring av passord har vi både /change_password, og /forgotten_password. Før man kaller /forgotten_password må man ha kalt /forgotten_my_password som sender en mail til brukerens registerte epost adresse med en lenke som inneholder en token som gir de rettighet til å sette nytt passord.

5.1.5 Auth

Dette er ruter som brukes til innlogging og oppdatering av JWT. Rutene finnes på endepunkter med start URL /auth. Eksmepelvis /auth/login og /auth/refresh-token.

6. Sikkerhet

6.1 Generell sikkerhet

Vi benytter oss av "codeql" på GitHub som analyserer kildekoden vår for å identifisere mulige sikkerhetsproblemer. Denne har funnet og rapportert problemer som polynomiske regex uttrykk som ikke hadde maks lengde på input (muliggjør ReDoS), databasekall med usanitert brukerinput (NoSQL Injection) og lignende.

Sensitiv informasjon ligger ikke i kildekoden, men blir satt med miljøvariabler når man installerer og kjører systemet.

6.2 Sanitering av brukerinput

På rutene i REST-APIet bruker vi node pakkene "mongo-escape" og "escape-string-regex" som bytter ut eller escaper input slik at input fra bruker ikke tolkes som gyldig query kode eller regex. Det er noen steder hvor vi istedenfor har brukt "$eq" operatoren og manuell fjerning av "$" etter tips fra codeql, og hvor vi trenger å la brukeren bruke "." i inputten sin, f.eks. i søk og filtrering av undersøkelser. Dette stopper brukeren fra å kunne gjøre ting som "$gt > 0" eller "$exists: true" for å få ut informasjon, men vi kan fremdeles inkludere

sub-dokumenter i søk via "." operatoren.

Oppå dette brukes "Mongoose" som er en ODM wrapper for MongoDB driveren, og vi har her satt spesifikke datatyper på schemaene. Det vil si at input av feil type stopper databasekallet.

Vi har også prøvd å være sparsom på bruken av "de farlige" MongoDB query typene som "$where" og "$mapReduce" som tar inn og kjører javascript i database-instansen.

6.3 Kryptert kommunikasjon

Systemet anvender et TLS-sertifikat for å sikre krypterte kommunikasjonskanaler på webtjener og REST-API. Det er en fallback mode hvor backend og frontend kjører uten HTTPS, og denne er bare tenkt brukt når man videreutvikler produktet. Fallback skjer bare når miljøvariablene for filstien til sertifikatet og nøkkelen mangler.

6.4 Hashing

Det anvendes hashing med salt for å håndtere sikker lagring av brukernes passord. Det innebygde node biblioteket "crypto" anvendes for både generering av saltet og for hashing med pbkdf2 algorithmen. Både generering av salt og hashingen skjer på server-siden. Saltet genereres med crypto.randomBytes(256) og pbkdf2 algoritmen bruker 1 million iterasjoner med sha256 som resulterer i sirka 250 millisekunder per hash på en moderne CPU, noe som drastisk reduserer muligheten for brute-force angrep, selv med GPU-er eller ASIC-er. Passordet manipuleres ikke på noen annen måte før det sendes fra klienten til serveren over HTTPS. Klartekst passord lagres ikke. Det hashede passordet og salt lagres i User entiteten i databasen med hex encoding.

6.5 Autentisering og Autorisering

Systemet bruker JSON Web Token (JWT) som settes med HTTPOnly informasjonskapsler for å gi brukere tilgang til ressurser på REST-APIet. Siden vi bruker HTTPOnly kan ikke kapslene hentes ut via javascript, og en angriper må dermed ha tilgang til enheten man bruker, eller analysere nettverkstrafikken (som bruker HTTPS). Vi bruker også JWT for

registrering av brukere, og for å la en bruker resette passord om brukeren har glemt passordet sitt.

Systemet har følgende tokens:

access-token:

- For registrerte brukere, settes ved innlogging
- 30 minutter ekspirasjonstid
- Kan brukes for å få en ny token (refresh)
- Gir full tilgang til det den registrerte brukeren har tilgang til

judge-token:

- For dommere, settes når man ber om å svare på en aktiv undersøkelse
- 30 minutters ekspirasjonstid
- Kan brukes for å få en ny token (refresh)
- Gir bare delvis tilgang til undersøkelser som er aktive

RegistrerBruker-token:

- Krever admin bruker for å opprette
- 24 timers ekspirasjonstid
- Bruksområdet tar høyde for om tokenet har blitt brukt (sjekker om bruker med korrekt epost-adresse eksisterer)
- Gir bare tilgang til å registrere en bruker med korrekt (invitert) epost-adresse
- Kan ikke brukes for å få ny token (no-refresh)

GlemtPassord-token:

- Kan opprettes av hvem som helst og sendes til en brukers registrerte epost-adresse.
- 1 time ekspirasjonstid
- Bruksområdet tar per i dag ikke høyde for om tokenet har blitt brukt
- Gir bare tilgang til å endre passord for den registrerte brukeren man etterspurte token for
- Kan ikke brukes for å få ny token (no-refresh)

Det er bare access-token og judge-token som settes via informasjonskapsler, da Registrer bruker og Glemt passord sendes på mail.

7. Installasjon og kjøring

7.1 Eksterne avhengigheter

- Det forventes et linux basert operativsystem
- Docker og docker-compose
    - Systemet er laget for å kjøre i et docker miljø
    - Installeres fra https://www.docker.com/
- Node.js og npm
    - Både frontend og backend bruker npm for å installere 3. parts biblioteker
    - Installeres fra https://nodejs.org/
- TLS sertifikat for frontend og backend
    - Kreves for bruk av HTTPS
    - Kan for eksempel få gratis sertifikat fra https://letsencrypt.org
    - Se eget brukerguide [1] for eksempel på framgangsmåte for dette

7.2 Programvarebiblioteker

Alle programvarebibliotekene vi benytter er inkludert i prosjektet eller installeres via npm.

7.2.1 Frontend

@mdi/js [2] er en samling av *material design* ikoner vi benytter for å forbedre UI/UX og gjør det lettere å bruke siden da vi bruker veletablerte ikoner til knapper.
@zerodevx/svelte-toast [3] gir oss muligheten til å vise en "toast" på nettsiden. Dette er en kort melding som kan lukkes manuelt eller lukkes automatisk via en timer.
axios [4] er en wrapper for "fetch" og benyttes for å kommunisere med backend.
clamp-js [5] brukes for å gjemme tekst som ellers ikke ville fått plass i en kontainer.
file-saver [6] brukes for å laste ned forskningsdata fra undersøkelser i CSV format.
query-string [7] håndterer parsing av URL-er og gir oss en enkle måte å hente ut parametere.
svelte-materialify [8] er et UI bibliotek med ferdige svelte komponenter som kort, knapper input-felter med mer.
svelte-routing [9] lar oss servere siden som en Single Page Application ved å endre innholdet som vises basert på URL, uten at det krever refresh av siden.

sweetalert [10] er et bibliotek som lar oss vise et overlay med meldinger og knapper. Er mye brukt for å gi tilbakemelding til brukerens handlinger, for eksempel om det gikk bra eller oppstod problemer med sletting av bruker.
uuid [11] lar oss generere unike ID-er.
Mozilla pdf-js [12] lar oss vise fram PDF-er i nettleseren, både på dekstop og mobil/tablet. Dette biblioteket ligger direkte i client/src/utility og kopieres over til client/public mappen ved kompilering av prosjektet.

7.2.2 Backend

apidoc [13] benyttes for å kunne generere dokumentasjon for REST-APIet.
axios [4] benyttes for å sende HTTP forespørsler.
express [14] er et web utviklings rammeverk for Node.js og benyttes for REST-APIet. Express håndterer routing og har ekstremt bred adopsjon og støtte for en mengde forskjellige middlewares som minker mengden boilerplate kode som kreves.
body-parser [15] er et middleware til express som håndterer data som sendes med HTTP og gjør disse disse tilgjengelig via request.body.
cookie-parser [16] er et middleware til express som gjør håndtering av informasjonskapsler enklere. Informasjonskapslene som følger med en forespørsel kan leses fra request.cookies.
cors [17] er et middleware til express som håndterer Cross-Origin-Resource-Sharing og benyttes for å la nettsiden vår snakke med REST-APIet. Dette kreves da nettsiden og APIet kjøres på forskjellige porter og er derfor distinkte origins.
dotenv [18] er et bibliotek som leser miljøvariabler fra en .env fil og benyttes for å enklere kunne konfigurere systemet, og å holde sensitiv informasjon slik som passord utenfor kildekoden.
escape-string-regexp [19] er et bibliotek vi brukes for å sanitere input som brukes i regex uttrykk, den sørger for at input ikke behandles som en del av regex-en.
express-fileupload [20] er et middleware som håndterer filopplastning.
jsonwebtoken [21] brukes for autentitisering.
mongo-escape [22] saniterer input som brukes i databasekall mot MongoDB slik at input ikke behandles som en del av databasekallet.
mongoose [23] er en ODM wrapper for MongoDB driveren og lar oss sette restriksjoner på database-entitetene for å øke sikkerheten og integriteten av dataen.
nodemailer [24] brukes for å kunne sende mail fra systemet.

crypto [25] er innebygd i Node og brukes for å genere salt og hashe passord med pbkdf2.
sjcl [26] benyttes for hashing med SHA256 for epost-adresser som benyttes i JWT.

7.3 Miljøvariabler

7.3.1 acj-server:

MongoDBConnectionString: skal være på formen:
"mongodb://brukernavn:passord@acj-db".
ExpressServerPort: porten REST-APIet blir servert fra.
estimateServicePath: påkrevd, full URL til estimate tjenesten, typisk:
"http://acj-analyse:1030/estimate".
JWTSecret: brukes for å signere access-token. Anbefales en lengre streng, for eksempel:
"TAxvUzIvAE9FKVBI8wAhbfVAaatDEwwujF1aSUIC"
JWTJudgeSecret: brukes for å signere judge-token. Samme anbefaling som *JWTSecret*, men skal være forskjellig.
JWTRegisterUserSecret: Brukes for å signere RegisterUserToken. Samme anbefaling som *JWTSecret*, men skal være forskjellig.
JWTForgottenPasswordSecret: Brukes for å signere ForgottenPasswordToken. Samme anbefaling som *JWTSecret*, men skal være forskjellig.
AdminUsername: Brukes for å opprette administrator-brukeren for systemet. Anbefalt "admin@dittdomene"
AdminPassword: Brukes for å opprette administrator-brukeren for systemet. Minimum 12 bokstaver, anbefaler å lage et sterkt passord.
MAIL_USER: Valgfri, brukernavn for mail-tjener, ikke bruk denne om du ikke trenger å autentisere deg for mail-tjener.
MAIL_PW: Valgfri, passord for mail-tjener, ikke bruk denne om du ikke trenger å autentisere deg for mail-tjener.
MAIL_FROM_STRING: epost-adressen du bruker som avsender adresse.
MAIL_SERVER: URL til mail-tjeneren du bruker
MAIL_PORT: Porten mail-tjeneren kjører på
CLIENT_BASE_URL: URL til frontend. Brukes for å generere korrekt lenke når man inviterer brukere eller når en bruker har glemt sitt passord.

7.3.2 acj-db:

MONGO_INITDB_ROOT_USERNAME: Brukernavn til admin bruker for databasen.
MONGO_INITDB_ROOT_PASSWORD: Passord til admin bruker for databasen.

7.4 Installasjonsveiledning

1. Klon prosjektet: `git clone https://github.com/jorgstei/Web-Platform-for-Adaptive-Comparative-Judgement.git`
2. Gå til mappen: `cd Web-Platform-for-Adaptive-Comparative-Judgement`
3. Gjør nødvendige endringer i docker-compose.yml (porter og volumes for tjenestene)
4. Lag en .env fil for mongodb og REST-API server: `cp ./mongodb/default.env ./mongodb/.env; cp ./server/default.env ./server/.env`
5. Fyll inn /server/.env og /mongodb/.env
6. Endre /client/rollup.config.js slik at apiBasePath blir satt til rett URL (backend/server adresse)
7. Endre /client/package.json "scripts.prod" til å peke på rett sertifikat og nøkkel
8. Legg til domenet til klienten i listen over godkjente origins for CORS i /server/server.js
9. Gjør ./build.sh kjørbar: `chmod +x ./build.sh`
10. Kjør `./build.sh`
11. Sjekk at alle tjenestene er oppe ved å kjøre `docker ps`

8. Dokumentasjon av kildekode

Full dokumentasjon av REST-api finnes i kildekoden til modulen og det kan genereres en web-versjon ved å kjøre "npm run docs" i "server" mappen. Denne kommandoen vil generere HTML og Javascript som blir plassert i mappen "/server/apidoc". Dokumentasjonen kan åpnes lokalt fra "/server/apidoc/index.html" eller legges på en webserver om det ønskelig.

På frontend er det skrevet dokumentasjon inline med kildekoden der det er hensiktsmessig.

9. Referanser

[1]. Brukerguide.pdf, se 064_vedlegg.zip

[2]. https://www.npmjs.com/package/@mdi/js [Besøkt 13.05.2021]

[3]. https://www.npmjs.com/package/@zerodevx/svelte-toast [Besøkt 13.05.2021]

[4]. https://www.npmjs.com/package/axios [Besøkt 13.05.2021]

[5]. https://www.npmjs.com/package/clamp-js [Besøkt 13.05.2021]

[6]. https://www.npmjs.com/package/file-saver [Besøkt 13.05.2021]

[7]. https://www.npmjs.com/package/query-string [Besøkt 13.05.2021]

[8]. https://www.npmjs.com/package/svelte-materialify [Besøkt 13.05.2021]

[9]. https://www.npmjs.com/package/svelte-routing [Besøkt 13.05.2021]

[10]. https://www.npmjs.com/package/sweetalert [Besøkt 13.05.2021]

[11]. https://www.npmjs.com/package/uuid [Besøkt 13.05.2021]

[12]. https://mozilla.github.io/pdf.js/ [Besøkt 13.05.2021]

[13]. https://www.npmjs.com/package/apidoc [Besøkt 13.05.2021]

[14]. https://www.npmjs.com/package/express [Besøkt 13.05.2021]

[15]. https://www.npmjs.com/package/body-parser [Besøkt 13.05.2021]

[16]. https://www.npmjs.com/package/cookie-parser [Besøkt 13.05.2021]

[17]. https://www.npmjs.com/package/cors [Besøkt 13.05.2021]

[18]. https://www.npmjs.com/package/dotenv [Besøkt 13.05.2021]

[19]. https://www.npmjs.com/package/escape-string-regexp [Besøkt 13.05.2021]

[20]. https://www.npmjs.com/package/express-fileupload [Besøkt 13.05.2021]

[21]. https://www.npmjs.com/package/jsonwebtoken [Besøkt 13.05.2021]

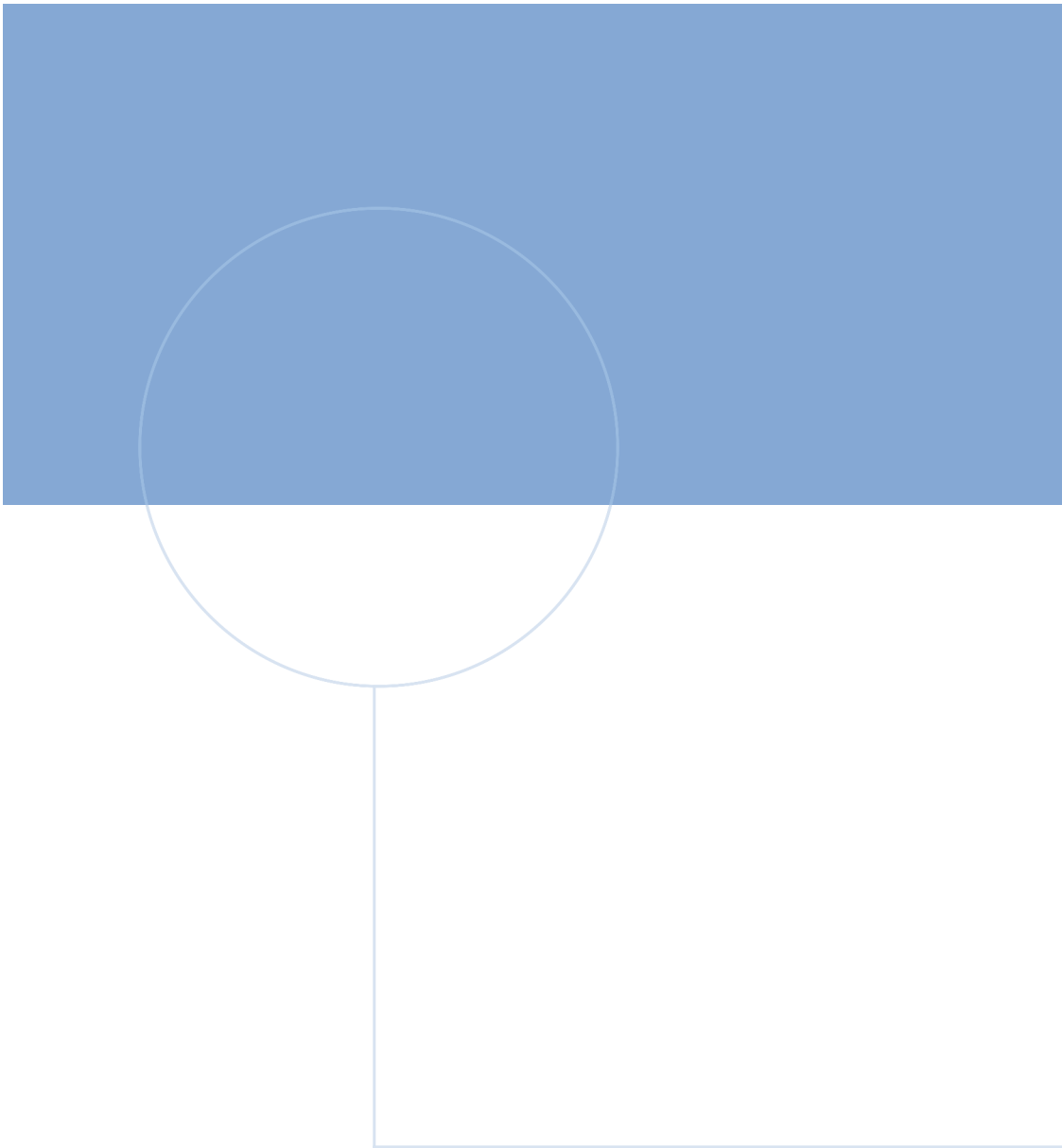
[22]. https://www.npmjs.com/package/mongo-escape [Besøkt 13.05.2021]

[23]. https://www.npmjs.com/package/mongoose [Besøkt 13.05.2021]

[24]. https://www.npmjs.com/package/nodemailer [Besøkt 13.05.2021]

[25]. https://nodejs.org/api/crypto.html [Besøkt 13.05.2021]

[26]. https://www.npmjs.com/package/sjcl [Besøkt 13.05.2021]



NTNU

Norwegian University of
Science and Technology