Mikkel Ofrim
Oskar Remvang

NTNUI NDA-System

Bacheloroppgave i Dataingeniør
Veileder: Surya Kathayat
Mai 2023

Bacheloroppgave

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

NTNU
Kunnskap for en bedre verden

Mikkel Ofrim
Oskar Remvang

NTNUI NDA-System

Bacheloroppgave i Dataingeniør
Veileder: Surya Kathayat
Mai 2023

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

NTNU
Kunnskap for en bedre verden

Sammendrag

NTNUI er Norges største idrettsorganisasjon med et bredt spekter av idretter og et stort antall medlemmer. Som en følge av sin størrelse og medlemsmasse sitter organisasjonen på enorme mengder personlig informasjon om mange mennesker. Med et ønske om å prioritere personvern og sikkerhet innad i organisasjonen, ville NTNUI senke terskelen for å få sine tillitsvalgte til å signere taushetserklæringer. Håpet er at dette vil fremme at personvern blir tatt på alvor.

For å løse dette problemet utviklet vi NDA-systemet, som er et avansert system for generering, signering, utsending og lagring av taushetserklæringer. Systemet ble utviklet ved bruk av moderne teknologier som React, Node.js og MongoDB, og det ble integrert med NTNUI sine allerede eksisterende interne systemer. Gjennom dette systemet kan brukere enkelt generere og signere taushetserklæringer, sende dem til riktig person og ha tilgang til de signerte kontraktene.

NDA-systemet har effektivt transformert en tidkrevende og manuell prosess til å bli raskere, digitalisert og mer automatisert. Ved å automatisere genereringen og utsendingen av taushetserklæringene, har systemet gjort det enklere for NTNUI sine tillitsvalgte å oppfylle sine forpliktelser rundt taushetsplikt. Produktansvarlig er svært fornøyd med systemets funksjonalitet og brukervennlighet, og det skal rulles ut i produksjon.

Gjennom samtaler med fremtidige brukere av systemet ble de største utfordringene relatert til håndtering av taushetserklæringer kartlagt. Brukertester viste senere at den implementerte digitale løsningen var langt å foretrekke overfor fysisk signering.

Abstract

NTNUI is Norway's largest sports organization with a wide range of sports and a large number of members. Due to its size and scope, the organization holds vast amounts of personal information about many of its members. With a desire to prioritize privacy and security within the organization, NTNUI wanted to lower the threshold for its elected representatives to sign non-disclosure agreements (NDAs) to ensure that this issue was taken seriously.

To solve this problem, we developed the NDA system, which is an advanced system for generating, sending, and storing NDAs. The system was developed using modern technologies such as React, Node.js, and MongoDB, and it was integrated with NTNUI's existing internal systems. Through this system, users can easily and automatically generate NDAs, send them to the appropriate individuals, and have access to the signed contracts.

The NDA system has efficiently transformed a time-consuming and manual process into becoming faster, digitalized, and partially automated. By automating the generation and sending of NDAs, the system has made it easier for NTNUI's elected representatives to fulfill their obligations regarding confidentiality. The productowner is very pleased with the functionality and user-friendliness of the system, and it will be rolled out into production.

The main challenges related to processing NDAs were uncovered through conversations with future users of the system. User tests later revealed that the digital solution is preferable to physical signing.

Forord

Denne rapporten omhandler utviklingen av et taushetserklæringssystem for NTNUI. Rapporten inneholder informasjon om selve oppgaven, hva som har blitt gjort og relevant teori.

Det er flere grunner til av teamet valgte denne oppgaven:

- Teamet ønsket å jobbe med et fullstack webutviklingsprosjekt.
- Teamet ønsket å bygge noe fra bunnen av, heller enn å bare videreutvikle eksisterende systemer.
- Teamet ville lage noe som ville ha en praktisk anvendelse og faktisk bli tatt i bruk for å gjøre arbeidet lettere for NTNUIs tillitsvalgte.
- Teamet har en forkjærlighet for NTNUI. NTNUI er kun driftet på frivilligehet og teamet hadde lyst til å gjøre noe som kunne hjelpe driftingen av idrettslaget videre.

Teamet vil gjerne takke Ole Marius Inderhaug, Axel Legallais-Korsbakken og Mats Finsås fra NTNUI Sprint for en flott oppgave og bra oppfølgning under prosessen. Teamet ønsker også å takke Surya Kathayat, Førsteamanuensis ved Institutt for datateknologi og informatikk på NTNU, for god veiledning underveis. Til slutt vil vi gjerne takke alle som deltok i brukertestene våre.

Trondheim, 20.05.2023

Sted, dato

Ofrim, Mikkel

Trondheim, 20.05.2023

Sted, dato

Remvang, Oskar

Revisjonshistorikk

Dato	Versjon	Beskrivelse	Forfatter
20.01.2023	0.1	Oppsett av mal	Mikkel Ofrim, Oskar Remvang
15.05.2023	1.0	Førsteutkast	Mikkel Ofrim, Oskar Remvang
17.05.2023	1.1	Revidering før innlevering	Mikkel Ofrim, Oskar Remvang
20.05.2023	2.0	Gjort klar for innlevering	Mikkel Ofrim, Oskar Remvang

Oppgavetekst

Hensikten av oppgaven er å lage et system der NTNUIs tillitsvalgte kan signere digitale taushetserklæringer. I dag gjøres dette med penn og papir. Med over 600 tillitsvalgte totalt, der 300 skiftes hvert år, så medfører det en stor manuell jobb for oppfølging, arkivering og overvåkning. Kontroll av taushetserklæringer er derfor noe som blir nedprioritert av organisasjonens Hovedstyre i dag, siden organisasjonen krever ressurser på andre områder som gir mer umiddelbar verdi.

NTNUI tar personvern på alvor, og ønsker derfor å forenkle prosessen ved å signere taushetserklæringer, slik at personvernet for idrettslaget og dets 14 000 medlemmer overholdes på en tilfredsstillende måte.

NTNUI Taushetserklæring er en utvidelse av NTNUIs nyeste satsing på mikrotjenestearkitektur. Det skal lages en webapplikasjon som integrerer mot NTNUIs sentrale API (api.ntnui.no). Når en tillitsvalgt i NTNUI skal signere en taushetserklæring, må vedkommende autorisere seg via det sentrale API-et, slik at NTNUI kan ha oversikt over hvem som har signert.

I tillegg er det ønskelig at applikasjonen utvides med et administrasjonspanel der Hovedstyret kan logge inn og se status over hvem som har signert.

Oppgaveomfanget er å lage en mikrotjeneste. Mikrotjenesten inneholder en frontend, en backend og en egen, isolert database.

NTNUI ønsker at prosjektet benytter seg av samme teknologier som andre tjenester i NTNUIs økosystem. Det er derfor ønskelig at frontenden skrives i TypeScript med enten React eller Vue som rammeverk, og at backenden enten bruker Express med TypeScript eller Django REST.

Dersom studentene ønsker å jobbe med andre teknologier vil det selvfølgelig respekteres.

Studentene må belage seg på at applikasjonen skal produksjonssettes i løpet av noen måneder etter leveranse. Dette var tilfellet for et bachelor-prosjekt NTNUI hadde ansvar for våren 2022. Det stilles derfor krav til at studentene lager en enkel CI/CD-pipelines.

NTNUI ønsker å forvalte programvaren etter at studentene er ferdig. Det er derfor ønskelig at studentene har tatt stilling til hvordan testing skal gjennomføres og hvilke rammeverk som skal brukes. I tillegg er det ønskelig at det skal ha blitt laget noen tester, både på frontend og backend.

Endringer fra den originale oppgaveteksten

Selv om oppgaveteksten i stor grad er den samme, har det vært noen tillegg av ekstra funksjonalitet og spesifikasjoner i etterkant. Disse beskrives nærmere i senere kapitler, samt i Vedlegg A - Visjonsdokument. Dette er kun utvidelser av den originale teksten, så den gjelder fortsatt. Det eneste som har endret seg fra den originale teksten er at Hovedstyret nå har tilgang til ekstra funksjonalitet på hjemmesiden i stedet for et eget administrasjonspanel.

Innhold

Figurer

Chapter 1

Introduksjon og relevans

1.1 Bakgrunn

Med over 14 000 medlemmer, spredt over et titalls idretter og administrative utvalg, er NTNUI Norges største idrettsforening. Som følge av dette har tillitsvalgte i idrettsforeningen tilgang til store mengder personlig og sensitiv informasjon som ikke burde deles med parter som ikke trenger den. Ett av tiltakene NTNUI har satt inn for å hindre deling av personlig informasjon er å kreve at alle tillitsvalgte med tilgang til slik informasjon må signere en taushetserklæring.

Det nåværende systemet for kontrakthåndtering innebærer manuell opprettelse og utskrift av kontrakter, signering med penn og papir av begge parter, og lagring av kontrakten i et fysisk lagerrom. Dette systemet har en rekke utfordringer, spesielt i en organisasjon som NTNUI, som har over 600 tillitsvalgte og bytter ut over 300 av disse hvert år. Den største utfordringen er tidsbruken som kreves for å administrere kontraktene på denne måten. Med så mange tillitsvalgte som må håndtere kontrakter, kan det ta mye tid å lage, signere og lagre alle kontraktene. I tillegg er det ingen effektiv måte å spore hvem som har signert og hvor de signerte kontraktene er lagret. På grunn av disse utfordringene blir kontrakthåndtering ofte nedprioritert av organisasjonens hovedstyre, da organisasjonen krever mye ressurser på andre områder som gir en mer umiddelbar verdi.

NTNUI ønsket derfor et brukervennlig system som forenkler og effektiviserer denne prosessen så mye at det ikke lenger nedprioriteres.

1.2 Problemstilling

Oppgavens hovedfokus ligger på å utvikle et internt system for håndtering av taushetserklæringer for NTNUI. I tråd med utviklingen vil gruppen undersøke følgende problemstilling:

Hvilke utfordringer oppstår i forbindelse med håndtering av taushetserklæringer, og hvordan kan disse utfordringene adresseres gjennom digitalisering?

1.3 Hovedrapportens struktur

Kapittel 1 - Introduksjon og relevans

Dette kapitlet gir en grundig beskrivelse av oppgavens innhold og formål. Her finner du informasjon om oppgavens bakgrunn, selve oppgaveteksten, problemstillingen, samt annen relevant informasjon.

Kapittel 2 - Teori og relevant litteratur

I dette kapittelet blir diverse teoretiske konsepter som brukes senere i teksten definert og forklart.

Kapittel 3 - Metode

Her blir de forskjellige teknologiene og utviklingsmetodene teamet brukte forklart og begrunnet.

Kapittel 4 - Resultater

Dette kapittelet inneholder en grundig gjennomgang av systemet som har blitt utviklet, samt en beskrivelse av de endelige resultatene knyttet til utforskning av problemstillingen og tidsbruk i løpet av prosjektperioden.

Kapittel 5 - Diskusjon

I dette kapittelet blir resultatene fra kapittel 4 drøftet.

Kapittel 6 - Konklusjon og videre arbeid

Her finnes konklusjonen, samt mulig videre arbeid som kan gjøres på prosjektet.

Samfunnspåvirkning

Dette kapittelet vil ha en kort refleksjon rundt samfunnspåvirkningen av systemet som ble utviklet.

1.4 Akronymer og forkortelser

API Application Programming Interface

JSON JavaScript Object Notation

NDA Non Disclosure Agreement, engelsk for taushetserklæring

CI Continuous Integration

URL Uniform Resource Locator

URI Uniform Resource Identifier

DOM Domain Object Model

JSX JavaScript XML

CSS Cascading StyleSheet

HTTP HyperText Transfer Protocol

HTTPS HyperText Transfer Protocol Secure

HTML HyperText Markup Language

JWT JSON Web Token

1.5 Ordliste

1.5.1 NTNUI

Her er forklaringer på flere begreper og referanser som er knyttet til NTNUI sin interne struktur, og som vil bli benyttet i hovedrapporten uten ytterligere forklaring.

Gruppe En gruppe i NTNUI er en spesifikk idrett. Alle grupper har sin egen leder, nestleder, kasserer og styre, samt medlemmer. For å bli medlem i en gruppe må du først være medlem i NTNUI som helhet. Eksempler på grupper er NTNUI Svømming eller NTNUI Volleyball.

Utvalg Et utvalg ligner veldig på en gruppe, men det er ikke en idrett. Et utvalg har den samme strukturen som en gruppe, men det er et administrativt utvalg som har en rolle som påvirker alle gruppene. Eksempler på utvalg er NTNUI Sprint som er teknisk ansvarlige for hele NTNUI og NTNUI Event som er ansvarlige for å gjennomføre arrangementer for hele NTNUI.

Hovedstyret Hovedstyret er foreningens høyeste myndighet. Det er de som styrer selve NTNUI.

Gruppeleder En gruppeleder er personen som er leder for en spesifikk gruppe. De er ansvarlige for driften av sin gruppe.

Styremedlem Et styremedlem er i styret på en gruppe. Denne gruppen dekker alle som er i styret til en gruppe som ikke er leder.

Medlemssystemet Medlemssystemet er hvordan NTNUI har oversikt over styrene og medlemmene i alle grupper og utvalg. Dette systemet utvikles og driftes av NTNUI Sprint.

1.5.2 Diverse

NDA-database NDA-databasen er databasen til NDA-systemet, altså systemet som denne rapporten omhandler.

Chapter 2

Teori og relevant litteratur

2.1 Taushetserklæring

En taushetserklæring er en kontrakt som brukes til å sikre konfidensialitet og beskyttelse av sensitive opplysninger[24]. Det brukes vanligvis i profesjonelle eller forretningsmessige sammenhenger der en person eller organisasjon ønsker å sikre at visse opplysninger ikke blir videreformidlet eller brukt uten tillatelse. Det er vanlig at en taushetserklæring inneholder hvilken informasjon som omfattes av avtalen, varighet av avtalen og hvilken forpliktelser de forskjellige partene har.

2.2 Mikrotjeneste

En mikrotjeneste er en liten, selvstendig tjeneste i en applikasjon som utfører en spesifikk oppgave. De samarbeider ved hjelp av definerte grensesnitt og kan utvikles og deployes uavhengig av hverandre [1]. Mikrotjenester fremmer fleksibilitet, skalerbarhet og kontinuerlig leveranse i programvareutvikling.

2.3 Pagination

Pagination omfatter prosessen ved å splitte opp resultater i flere sider for å redusere trafikkbelastningen til et API. Det implementeres i tjenerkoden og fungerer typisk ved at klient sender et sidetall med hver ressursforespørsel som indikerer hvilke resultater tjener skal returnere[29].

2.4 Utviklingsmetodikk

2.4.1 Versjonskontroll

Versjonskontroll er en måte å spore endringer i filer over tid. Det tillater flere personer å jobbe på det samme prosjektet, spore forskjellige versjoner av filer og gjenopprette tidligere versjoner ved behov. Den vanligste typen versjonskontroll er noe som kalles distribuerte versjonskontrollsystemer. I disse versjonskontrollsystemene har hver bruker en komplett kopi av hele repositoriumet, noe som inkluderer hele endringshistorikken. Dette lar brukere jobbe offline og gjøre endringer lokalt, før det på et senere tidspunkt blir slått sammen med den originale kodebasen[6].

Det er flere fordeler ved å bruke versjonskontroll. Disse inkluderer:

- **Sammarbeid:** Versjonskontroll tilrettelegger for at flere personer kan jobbe på det samme prosjektet og at endringer enkelt kan slås sammen.
- **Versjonshistorikk:** Når man bruker et versjonskontrollsystem vil man ha tilgang til historikken til spesifikke filer. Man kan se hvilke endringer som er gjort, når disse endringene ble gjort og hvem som gjorde endringene. Dette er nyttig hvis man ønsker å se på endringshistorikken til en fil, eller om man trenger å gå tilbake til en gammel versjon.

- **Grener:** Grener (branches) er en måte å lage en "kopi" av prosjektet som man kan gjøre endringer på, uten at det påvirker selve kodebasen. Når man lager en gren vil det være en isolert kopi av kodebasen man kan bruke til å eksperimentere eller gjøre endringer på. Etter at endringene man ønsker å gjennomføre er ferdige kan en gren slås sammen med kodebasen igjen, og endringene som er gjort i grenen vil bli reflektert der.

2.4.2 Pull requests

Pull requests er tett knyttet opp mot versjonskontroll. Når en person ønsker å slå sammen en gren med en annen gren kan de lage en pull request. Dette handler i hovedsak om å legge til endringer fra en gren som inneholder utviklerens arbeid inn i hovedgrenen (selve kodebasen)[9].

Kontinuerlig integrasjon

Kontinuerlig integrasjon (Continuous integration/CI) er en praksis som automatisk bygger og tester koden i en gren hver gang det åpnes en pull request eller gjøres endringer i en eksisterende pull request. Dette kan identifisere feil tidlig i utviklingsprosessen og passer på at feilaktig kode ikke kommer inn i kodebasen[30].

Kodegjennomgang

Kodegjennomgang (Code reviews) er en praksis der en person fra teamet går gjennom endringene i koden som er foreslått i en pull request før de blir integrert i kodebasen. Dette er en viktig sikkerhetsmekanisme som sikrer høy kodekvalitet, samtidig som det gir en plattform for diskusjon og tilbakemelding på endringene. Det finnes ulike tilnærminger til kodegjennomgang, men en vanlig fremgangsmåte er at endringene ikke kan integreres i kodebasen før en annen person i teamet har gjennomgått og godkjent dem i pull requesten.

Squashing

En pull request kan ofte gjennomgå flere iterasjoner med endringer. Dette kan skyldes ulike årsaker, for eksempel feilfunksjon, kodeforbedringer eller endringer basert på tilbakemeldinger fra kodegjennomgang. Squashing er en prosess som slår sammen alle endringscommitene i en gren til én enkelt commit før den legges til i kodebasen. Dette gjøres for å gjøre endringene i kodebasen mer oversiktlige, siden det kun vil være én commit for hver pull request som blir integrert[31].

2.4.3 Wireframes

Wireframes er en illustrasjon av en nettside, applikasjon eller annet grensesnitt. Dette er noe som ofte lages tidlig i prosessen for å planlegge strukturen på det som skal lages. En wireframe bør være relativt simpel og skal ikke ta lang tid å lage[5]. Formålet er å ha noe konkret å se på, så alle er enige om hvordan grensesnittet skal se ut før man starte på utviklingen.

2.4.4 Issueboard

Et issueboard er en oversikt over gjøremål. Disse skal vises på en strukturert måte, gjerne med prioritet og andre merkelapper som f.eks. "refaktorering" eller "bug". Når man starter på en oppgave fra issueboardet skal man vise at man har startet på den oppgaven for å forsikre seg om at ingen andre starter å jobbe med det samme og så andre har oversikt over hvem som jobber med hva. Når oppgaven er ferdig skal den markeres som ferdig, og den vil da ikke være på listen over gjøremål lenger. Dette er et nyttig verktøy som bidrar til å koordinere og gi en oversikt over arbeidet i prosjektet.

2.4.5 Brukertester

Brukertester handler om å la ekte sluttbrukere teste et produkt, tjeneste eller prototype. Målet med dette er å få reell tilbakemelding og mulighet til å tilpasse produktet basert på disse tilbakemeldingene. Innenfor systemutvikling kjøres det gjerne flere runder med brukertester på forskjellige stadier av utviklingen for å kunne tilpasse utviklingen før for store endringer er gjort[25].

2.4.6 Testing

Unit tester

Unit testing er en testingsteknikk der individuelle deler av kildekoden testes isolert for å sikre at de fungerer som forventet. Dette innebærer å lage tester som sammenligner den faktiske oppførselen med den forventede oppførselen. Unit testing oppdager feil tidlig, hjelper med å opprettholde koden og gir utviklere tillit i koden de skriver[33].

End-to-end tester

End-to-end testing er en testingsteknikk der hele systemet blir testet fra start til slutt for å sikre at det fungerer som forventet. Det simulerer virkelige brukerscenarier og evaluerer samspillet mellom alle systemets komponenter. Målet er å identifisere problemer i systemet når det brukes som en helhet og sikre at det oppfyller kravene[32].

Chapter 3

Metode

3.1 Forskningsmetode

3.1.1 Brukertester

Teamet anså det å gjennomføre brukertester som essensielt for å nå målet om å utvikle et intuitivt og brukervennlig system. Brukertestene ble hovedsakelig gjennomført med tillitsvalgte i NTNUI, da det er de som er sluttbrukere av systemet. Måten vi fikk tak i disse personene var gjennom bekjente, samt en melding i en felles Slack-kanal for alle tillitsvalgte i organisasjonen. I denne Slack-kanelen er det flere hundre tillitsvalgte og flere av disse hadde mulighet til å stille til brukertester.

Selve gjennomføringen av brukertestene ble gjort med en kombinasjon av et fungerende system og mocking. Det så ut som et fungerende system for brukeren som gjennomførte testen. Her ble brukeren guidet gjennom en omfattende user story og oppfordret til å snakke høyt underveis. Med brukeren sitt samtykke ble det gjort lyd- og skjermopptak så teamet kunne gå gjennom brukertestene i etterkant for å hente ut den mest relevante informasjonen. Disse opptakene ble slettet etter at teamet hadde gått gjennom dem. Etter at systemgjennomgangen var fullførst ble brukeren spurt en rekke spørsmål angående opplevelsen. User story og spørsmålene kan finnes i Vedlegg E - Brukertestinstrukser.

3.1.2 Brukersamtaler

Teamet ønsket å få bedre forståelse for og identifisere de største utfordringene som oppstår i forbindelse med håndtering av taushetserklæringer. For å få dette hadde teamet samtaler med personer som er eller har vært involvert i denne prosessen. Disse samtalene ble hovedsakelig gjennomført med nåværende gruppeledere og medlemmer av Hovedstyret, da det i hovedsak er disse som er i målgruppen. Disse samtalene ble gjennomført tidlig i prosessen, så teamet fikk muligheten til å bruke disse tilbakemeldingene i utformingen av systemet.

Teamet fikk tak i personer å ha disse samtalene med gjennom bekjente og ved å spørre relevante folk på NTNUI klubbhuset om de kunne være med på en kort og uformell samtale. I disse samtalene ble personene spurt om hva de syntes om det nåværende systemet, om noe kunne forbedres og om de hadde noen forslag til hvordan det kan forbedres.

3.1.3 Andre signeringsimplementasjoner

Teamet gjennomførte en felles gjennomgang av andre signeringssystemer, f.eks. kontraktsignering gjennom Posten. Dette er noe teamet gjorde ettersom disse aktørene alt har velfungerende og intuitive systemer som kunne brukes til inspirasjon.

3.2 Valg av teknologier

I NTNUI er alle utviklere frivillige studenter ved NTNU. Disse studentene er ansvarlige for vedlikehold og videreutvikling av systemet i fremtiden. Derfor var det viktig å velge teknologi som gjør det enkelt for dem å utføre disse oppgavene. Teamet kom på tre punkter som vi syntes var spesielt viktige å vurdere i denne sammenhengen, og som vi baserte alle valgene våre på.

- Det må være en populær og vel etablert teknologi som det finnes mye ressurser på nettet om.
- Det må være en teknologi som aktivt brukes blant studenter ved NTNU.
- Det må være en teknologi som har blitt brukt i andre systemer i NTNUI sitt teknologiske økosystem.

3.2.1 Server

Express.js/Node.js

Node.js er en open-source, server-side, JavaScript runtime som lar utviklere kjøre JavaScript-kode utenfor en nettleser. Det ble først utgitt i 2009 og har siden blitt en populær plattform for utvikling av skalerbare, høy-ytelses applikasjoner. Express.js er et minimalistisk web-rammeverk for Node.js som gjør det enkelt å bygge web-applikasjoner og API-er[8][3].

Produkteier hadde et ønske om at vi skulle bruke enten Django eller Express.js som server-side teknologi for systemet. De fleste andre mikrotjenester som er utviklet og vedlikeholdet av NTNUI bruker Express.js, det er kun medlemssystemet som bruker Django. Det er også enklere å bruke det samme språket i både server og klient. Teamet bestemte seg dermed for å velge Express.js.

TypeScript

TypeScript utvider funksjonaliteten til JavaScript ved å gi utviklere muligheten til å angi typer på variabler. Dette er en funksjon som ikke er tilgjengelig i JavaScript, og gir flere fordeler, inkludert evnen til å oppdage feil i koden før den kjøres, økt kode robusthet og en mer lesbar og selvdokumentert kodebase. TypeScript-kode oversettes til JavaScript-kode, noe som betyr at det ikke er noen forskjell i det endelige resultatet mellom de to språkene[12].

Teamet valgte å bruke TypeScript hovedsakelig fordi det var et eksplisitt ønske fra produkteier om at vi skulle bruke det, men også på grunn av fordelene nevnt tidligere. En annen faktor var at ingen av oss hadde erfaring med TypeScript fra før, og vi var begge svært motiverte for å lære oss det. TypeScript ble brukt både i serveren og klienten.

3.2.2 Klient

React.js

React.js er et populært JavaScript rammeverk. Det gjør det enkelt å bygge interaktive og responsive brukergrensesnitt ved hjelp av gjenbrukbare komponenter[19].

Produkteier hadde et ønske om at systemet skulle utvikles med Vue eller React. Vi valgte å bruke React da det kun er medlemssystemet som er utviklet med hjelp av Vue. Resten av mikrotjenestene i NTNUI er laget med React.

Tailwind CSS

Tailwind CSS er et CSS rammeverk for frontend-utvikling som lar utviklere bygge og tilpasse et responsivt design raskt ved å bruke forhåndsdefinerte CSS-klasser for vanlige oppgaver. Det kan redusere utviklingstiden og øke produktiviteten ved å unngå å måtte skrive tilpasset CSS fra bunnen av, samtidig som det er relativt intuitivt å bruke[23].

Teamet valgte å bruke Tailwind da ett av medlemmene hadde brukt det før og anbefalte det. Etter å ha gitt det en sjanse var begge medlemmer av teamet enige om at det ville kutte betydelig ned på utviklingstiden til systemet.

3.2.3 Database

MongoDB

MongoDB er en dokument-orientert database. MongoDB bruker JSON-lignende dokumenter med dynamiske skjemaer, som gjør det enkelt å representere og lagre data på en fleksibel måte. Dokumentene kan inneholde komplekse hierarkiske strukturer, og kan enkelt utvides med nye felt og verdier etter behov uten at dette påvirker resten av dokumentet eller databasestrukturen[2].

Produkteier hadde ingen preferanse når det kom til hvilken database vi brukte. Det endte med at vi tok en avgjørelse etter å ha lest oss opp på forskjellige databaseløsninger. Denne avgjørsen var begrunnet med et par ulike grunner. Blant annet at produkteier nevnte at det er ønskelig å ha muligheten til å utvide systemet til å behandle andre type kontrakter i fremtiden, uten at det skal påvirke dataen som alt er lagret. Andre grunner inkluderer at MongoDB er godt dokumentert, det blir alt brukt i NTNUI økosystemet, og det er en simpelt og effektivt å jobbe med.

3.2.4 Prosess

GitHub

GitHub er en web-basert plattform som brukes for å administrere og dele programvareprosjekter. Det lar utviklere samarbeide om kode, diskutere prosjekter og administrere versjonskontroll av kode[10].

Teamet hadde ikke noe annet alternativ enn å velge GitHub som samarbeidsplattform og for versjonskontroll, da NTNUI utelukkende bruker GitHub til dette formålet. Avgjørelsen var imidlertid godt mottatt av teamet, siden GitHub tilbød alle funksjonene de trengte fra en slik plattform. Teamet benyttet seg av mange av de tilgjengelige funksjonene på GitHub, inkludert GitHub Actions for kontinuerlig integrasjon, Pull Requests for å sikre kvaliteten på koden, og GitHub Issue Board for å ha oversikt over oppgaver og ansvarsfordeling.

Dev-miljøet til NTNUI APIet

NTNUI APIet er det vi må bruke for å hente personer, roller, grupper og annen relevant informasjon vi trenger for at systemet vi utvikler skal fungere som det skal. NTNUI har utviklet en utviklerversjon av dette APIet (dev.api.ntnui.no) som fungerer på samme vis, bare at data som hentes eller opprettes her er fiktiv. Dette har latt oss teste at alt fungerer med APIet, uten å påvirke noen ekte data. Fleksibiliteten det har gitt oss ved å kunne opprette fiktiv data, endre på personopplysninger og slette data har vært uvurderlig under utviklingen.

3.2.5 Testing

Jest

Jest er et svært populært rammeverk for testing av JavaScript applikasjoner. Det kommer med et bredt utvalg av metoder som hovedsakelig brukes til å lage Unit-tester. Rammeverket kommer blant annet med enkel mocking av API-kall, autogenerering av testdekning og paralell testkjøring[13].

Rammeverket er mye brukt og veldokumentert. Dette passer godt for videreutviklere som skal sette seg inn i systemet.

Cypress

Cypress er et testing rammeverk for end-to-end og komponent testing av web-applikasjoner. Test-språket er enkelt oppbygget med fokus på intuitiv syntaks basert på engelsk semantikk. Når Cypress kjører tester kjøres klienten opp i det medfølgende brukergrensesnittet. Klienten gjør API-kall som ved vanlig kjøring. API-kallene kan også enkelt mockes dersom man utelukkende ønsker å teste klienten. Cypress muliggjør også å "single-steppe" gjennom testene samtidig som man ser klienten ved hvert steg i brukergrensesnittet. Dersom en test feiler vil Cypress automatisk lagre et bilde eller en video av klienten under feiling. Det er også mulig å kjøre Cypress uten brukergrensesnittet for å korte ned kjøretidene av testene[11].

Muligheten til å skrive reelle end-to-end tester, hvor både klient og tjener testes, trakk teamet mot å bruke Cypress. Det enkle brukergrensesnittet og den intuitive syntaksen gjør det også lettere for videreutviklere av systemet

å sette seg inn i testbasen. Ettersom det også er mulig å kjøre testene uten brukergrensesnittet, kolliderer heller ikke bruken av rammeverket med produkteiers krav om oppsett av CI Pipeline.

3.2.6 Diverse

Figma

Figma er en skybasert designplattform for å opprette og samarbeide på digitale designprosjekter[28]. Teamet skulle lage en mikrotjeneste helt fra bunnen av, og ønsket å visualisere det endelige produktet før selve utviklingen startet. Teamet valgte å bruke Figma for å lage wireframes på grunn av plattformens skybaserte funksjoner som gjorde det mulig for alle medlemmene i teamet å samarbeide på samme prosjekt, uavhengig av sted. Wireframes opprettet i Figma ser også realistiske ut og begge teammedlemmene hadde tidligere erfaring med plattformen, noe som gjorde lærekurven mindre bratt.

Diverse NPM-pakker

NPM er en forkortelse for "Node Package Manager", og fungerer som en pakkehåndterer for prosjekter skrevet i JavaScript og TypeScript. Ved hjelp av NPM har man tilgang til millioner av offentlige pakker og moduler som kan brukes fritt[4]. Selv om noen av de viktigste NPM-pakkene som ble brukt allerede er nevnt i detalj, har det også blitt brukt flere andre NPM-pakker for å håndtere ulike deler av prosjektet. Her er noen av disse pakkene, sammen med en kort forklaring av deres funksjoner.

- **Axios.** Axios er en populær JavaScript-basert HTTP-klient som brukes til å gjøre nettverksforespørsler fra en webapplikasjon[7].
- **Nodemailer.** Nodemailer er en Node.js-pakke som brukes til å sende e-post fra en server[14].
- **Ntnui-tools.** Ntnui-tools er en pakke som er utviklet av NTNUI og kan brukes for enkel samhandling med NTNUI APIet. Det inneholder for eksempel funksjoner for å hente informasjon om et NTNUI medlem, gitt en unik identifikasjon (NtnuiToken) og henting av NtnuiToken gitt login informasjon for en bruker[17].
- **Uuid.** Uuid er en pakke som generer unike identifikatorer som følger UUID-standarden. Identifikatorene som blir generert består av 32 heksadesimale sifre, som brukes til å identifisere data[26].
- **Pdf-lib.** Pdf-lib er et JavaScript-bibliotek som brukes til å manipulere PDF-dokumenter ved å legge til, fjerne eller endre eksisterende sider og innhold[18].
- **Nodemon.** Nodemon er en Node.js-pakke som brukes til automatisk å overvåke og restarte serveren hver gang det blir gjort endringer i koden[15].
- **Vite.** Vite er et moderne JavaScript byggeverktøy for utviklingsmiljøer. Det utnytter moderne nettleserteknologier for å korte ned byggetiden. Dette muliggjør svært korte byggetider, både ved første bygging, og ved oppdateringer[27].
- **React-signature-canvas.** React-signature-canvas er en React-komponent som gir muligheten til å lage signaturfelt på webapplikasjoner og lar brukere signere ved hjelp av musen eller fingeren på en touchskjerm[21].
- **React Router.** React Router er et populært JavaScript-basert rutingbibliotek for React som gir mulighet for klient-side navigasjon og dynamisk routing i en webapplikasjon[20].

Chapter 4

Resultater

4.1 Vitenskapelige resultater

4.1.1 Brukertester

Brukertestene ble gjennomført som planlagt. Gjennom disse brukertestene kom det frem både positive og negative bemerkninger, samt ønsket funksjonalitet, som teamet tok med seg videre i utviklingen. Alle tilbakemeldingene som kom fra brukertestene kan leses om i Vedlegg F - Brukertestresultater, men her er noen av de vanligste tilbakemeldingene:

- Den mest gjentagende tilbakemeldingen teamet mottok fra brukertestene var at systemet hadde samme generelle utseende som annen intern programvare i NTNUI, noe folk satt pris på.
- En annen gjentagende tilbakemelding var at systemer var intuitivt og enkelt å bruke.
- Det var også et ønske fra mange om at signaturen burde dukke opp i pdfen med en gang man signerer. Dette ble endret på i etterkant av brukertestene.
- Det å måtte trykke på en bruker for å få opp genereringsknappen er heller ikke intuitivt. Dette ble også endret på i etterkant av brukertestene.

4.1.2 Brukersamtaler

Gjennom samtalene som ble omtalt i kapittel 3.1.2 var det et par utfordringer som ble gjentatt flere ganger.

- Den mest nevnte utfordringen var at generering av taushetserklæringer er meget tidkrevende. Det er heller ikke alle som har kompetansen til å redigere PDFer, da det ikke er en standardisert måte å gjøre dette på.
- En annen utfordring var at det ikke var noe system for å vite om en person hadde signert en taushetserklæring fra før eller ikke. Da mange personer kan ha flere roller i organisasjonen, skal de egentlig bare trenge å signere en taushetserklæring. Med systemet som er på plass er det ingen effektiv måte å sjekke om en person har signert, og det var vanlig for personer å måtte signere mer enn en taushetserklæring gjennom sin tid i organisasjonen.
- Tett knyttet opp mot punktet over så er det ikke noe ordentlig system på plass for lagring av disse taushetserklæringene. De er lagret i permer i et lagerrom og det er ingen ordentlig struktur eller oversikt.
- Det å distribuere taushetserklæringer er også en flaskehals i prosessen, da signering må gjøres med penn og papir. Dette krever at de printes ut etter at de er laget, og at alle som skal signere dem må samles fysisk for at dette skal gjøres.

Når det kom til hvilke tiltak som var ønsket for å løse disse problemene var det også et par svar som gjentok seg.

- Det vanligste var et ønske om å kunne generere kontraktene automatisk så man skulle slippe å gjøre dette selv. Her kom det et par forskjellige tilnærminger på hvordan dette kunne løses, men fellestrekket var automatisk utfylling av navn, gruppe og roller.

- Et annet ønske som gjentok seg var muligheten til å kunne sende kontrakten digitalt til personer, da dette ville gjøre det lettere å få distribuert kontraktene uten å måtte møtes fysisk.
- Det ble også nevnt digital lagring med mulighet til å søke etter enkeltpersoner for å raskt kunne finne tilbake til en spesifikk kontrakt.

4.1.3 Andre signeringsimplementasjoner

Teamet gikk gjennom et par andre signeringsprosesser for inspirasjon, spesifikt hos Posten og hos SiT bolig. Det er vanskelig å si hvor mye som direkte ble brukt fra disse gjennomgangene, men det er trygt å si at de ga en pekepinn for hvordan en slik prosess burde gjennomføres.

4.2 Ingeniørfaglige resultater

Dette kapittelet tar for seg de ingeniørfaglige resultatene. Dette inkluderer en gjennomgang av systemet og en gjennomgang av de funksjonelle kravene som nevnt i Vedlegg A - Visjonsdokument.

4.2.1 Innlogging

Brukere må logge inn med samme konto som de bruker på medlemssystemet til NTNUI. Imidlertid er det kun gruppeledere og medlemmer av Hovedstyret som har tillatelse til å logge inn. Hvis en person forsøker å logge på med en gyldig NTNUI-konto som ikke er leder for en gruppe eller et utvalg, og heller ikke er medlem av Hovedstyret, vil vedkommende bli nektet tilgang.

Figure 4.1: Loginsiden

4.2.2 Oversikt

Etter at en person i Hovedstyret eller en gruppeleder logger inn vil de bli sendt til hjemskjermen. Her har de en oversikt over personene de skal ha tilgang til. Gruppeleder skal kun ha tilgang til det nåværende styret i sin gruppe, mens Hovedstyret skal ha tilgang til alle nåværende styrer i alle grupper, samt alle tidligere styremedlemmer i alle grupper som har signert en taushetserklæring.

På hjemskjermen vil brukeren også ha en del mulige handlinger. Disse inkludere filtrering, generering av taushetserklæringer, visning av signerte taushetserklæringer, nedlasting av signerte kontrakter og en profilmeny.

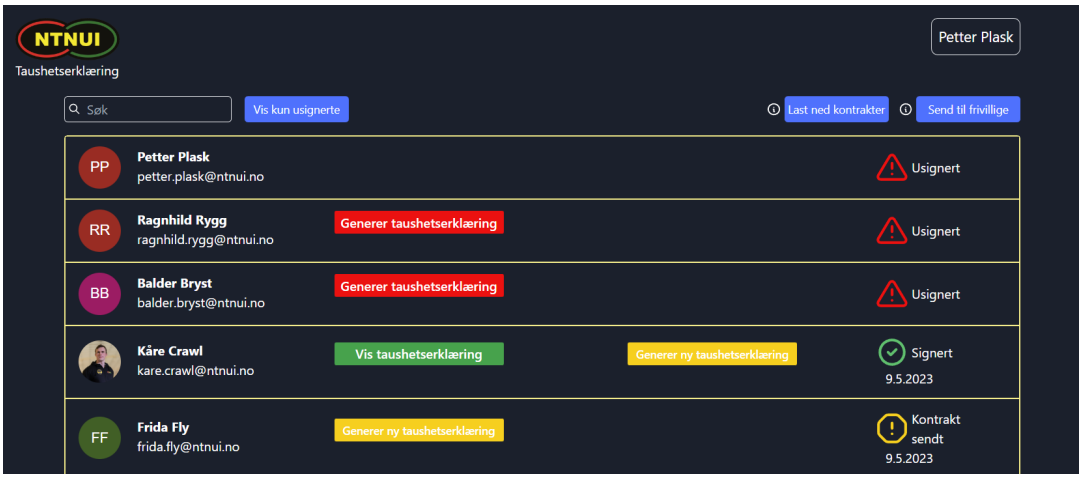


Figure 4.2: Hjemskjermen til Petter Plask, leder for NTNUI Svømming. Her har kun Kåre Crawl og Frida Fly mottatt en kontrakt, men det er kun Kåre Crawl som har signert den.

Hvis et medlem av Hovedstyret logger inn vil de, som nevnt tidligere, ha tilgang til alle grupper, samt personer som har signert tidligere men ikke lenger er i et gruppestyre eller utvalg. Mengden personer her kan fort bli stor, så de vil ha ekstra funksjonalitet for filtrering. De har tilgang til en gruppevelger, der de får en søkbar dropdown meny for å finne gruppen de leter etter. De har også et årstall-filter som filtrerer så kun personer som har signert kontrakten i et gitt år vises.

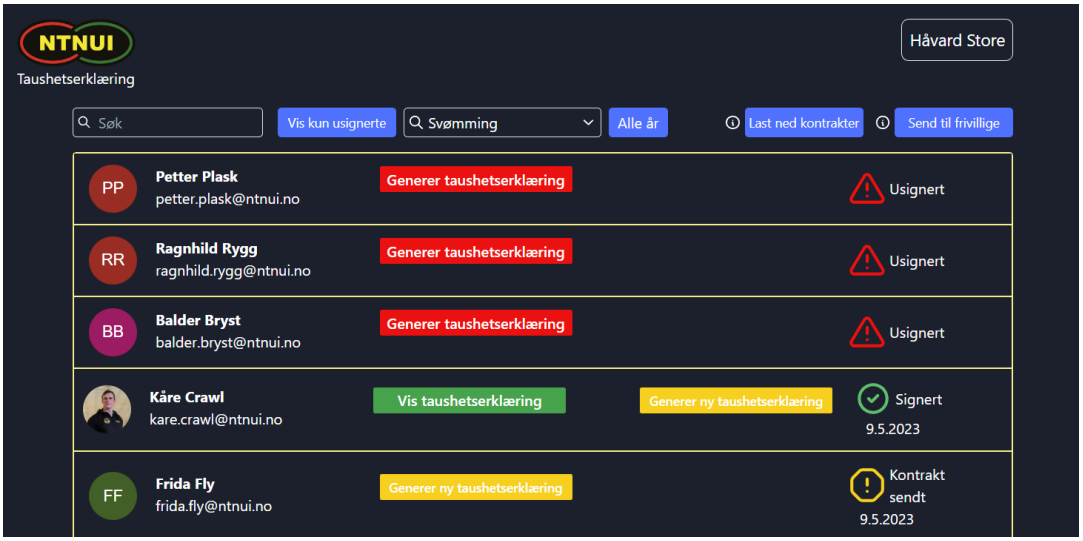


Figure 4.3: Hjemskjermen til Håvard Store, medlem av Hovedstyret. Her kan du se at han har en dropdown-meny for å velge mellom alle tilgjengelige grupper, samt et alternativ for å filtrere på hvilke år taushetskontrakten ble signert

Nedlasting av alle signerte kontrakter

Brukere kan laste ned alle signerte kontrakter i en spesifikk gruppe eller utvalg. Dette kan gjøres ved å trykke på "Last ned kontrakter" i høyre hjørne. Dette vil laste ned alle signerte kontrakter for den gitte gruppen som en zip-fil. Hvis du er logget inn som Hovedstyret vil du laste ned alle kontrakter som er signert for den gruppen, mens hvis du er en gruppeleder vil du kun laste ned signerte kontrakter for det nåværende styret.

4.2.3 Profilmeny

Ved å trykke på navnet sitt får brukere opp en meny som viser grunnleggende informasjon om innlogget bruker. Her har man også muligheten til å logge ut, samt til å laste opp og administrere en signatur.

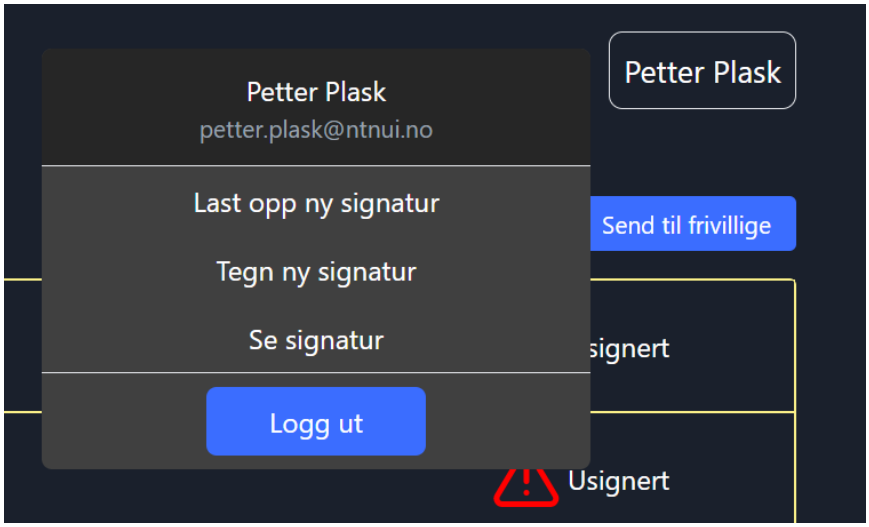


Figure 4.4: Brukermenyen til Petter Plask

Lagre signatur på profilen

I brukermenyen kan brukere laste opp og administrere en signatur som er knyttet til kontoen sin. Her kan brukeren velge å laste opp en signatur som eksisterer på maskinen fra før, eller om man ønsker å tegne en ny signatur med musen eller på touchskjerm. Etter en signatur enten er lastet opp eller tegnet har man mulighet til å se den, eller erstatte den ved å laste opp eller tegne en ny signatur. Signaturen som er lagret på en konto blir automatisk fylt inn på alle kontrakter som blir generert av den kontoen for å effektivisere prosessen for personer som må generere og sende ut mange taushetserklæringer.

4.2.4 Generering av kontrakter

Kontrakter kan genereres på to måter.

Generering for personer som er i et gruppestyre eller utvalg

For å generere en kontrakt for en person som er i et gruppestyre eller utvalg så må personen velges fra oversikten på hjemskjermen. Ved å trykke på "Generer taushetserklæring" så vil en kontrakt med de riktige navnene, rollene og gruppene opprettes.

Generering for personer som ikke er i et gruppestyre eller utvalg

For personer som ikke er i et gruppestyre eller utvalg må du bruke "Send til frivillige"-knappen i høyre hjørne på hjemskjermen. Her vil brukeren ha muligheten til å skrive inn all relevant informasjon om personen som skal motta kontrakten. Kontrakten vil dermed bli generert med den rette informasjonen for begge parter.

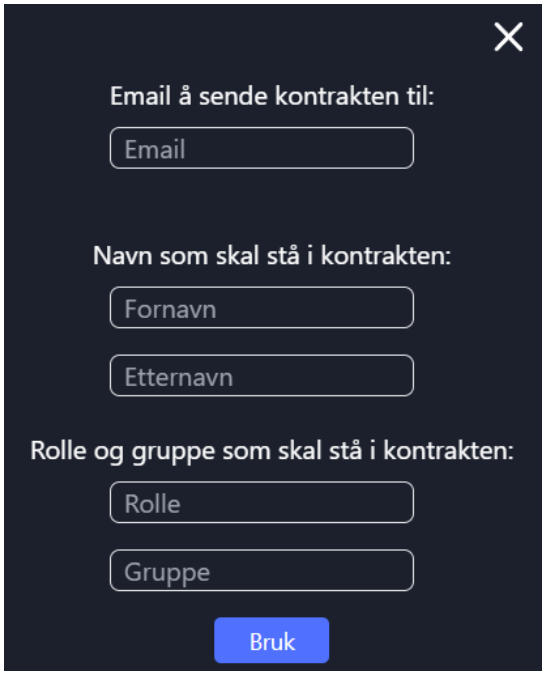


Figure 4.5: Popup der brukeren kan skrive inn informasjon om personen som kontrakten skal genereres for

Regenerering av kontrakt for personer som alt har signert

Brukere har mulighet til å generere og sende ut en ny kontrakt til personer som har mottatt, men ikke signert kontrakten sin enda. Dette er også mulig hvis noe går galt under signerings-prosessen, eller man av andre grunner har lyst til å erstatte den signerte kontrakten. For å gjøre det trykker brukeren på "Generer ny taushetserklæring" på den respektive personen det er ønsket å sende en ny kontrakt til.

4.2.5 Signering av kontrakter

Det er to forskjellige tilfeller hvor signering foregår, det er signering når man sender ut kontrakten og signering av en kontrakt man mottar på mail.

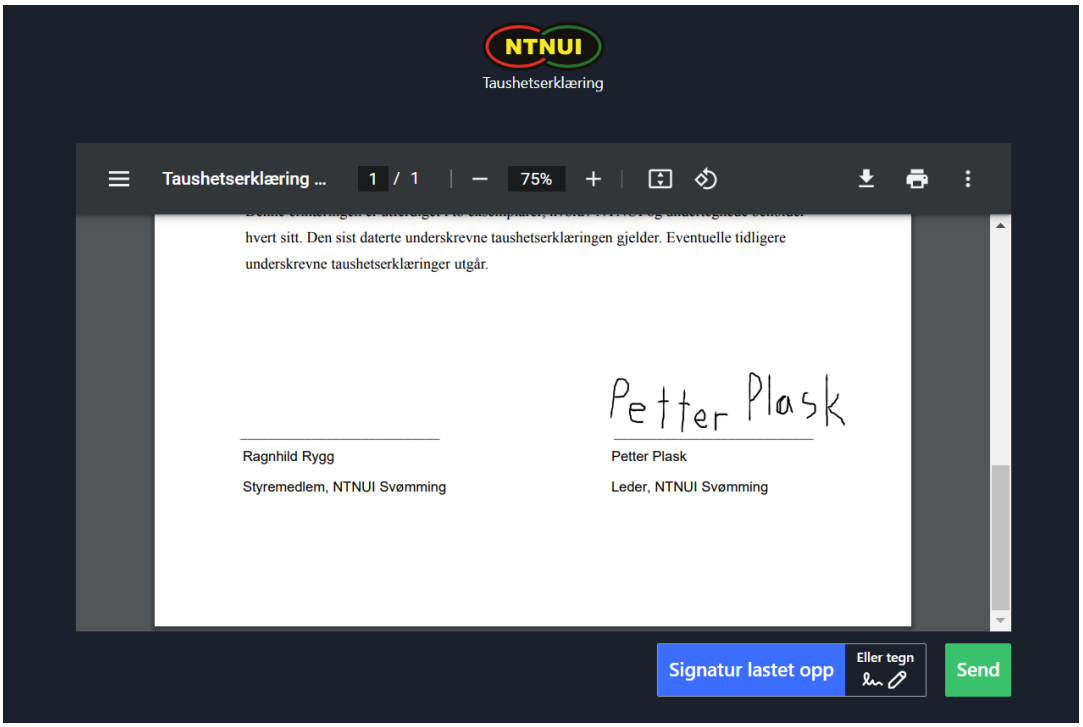


Figure 4.6: Signatursiden for avsender

Signering for sender

For sender så vil signatursiden starte med kontrakten som nettopp ble generert for personen. Kontrakten kan signeres på tre forskjellige måter.

1. Brukere kan signere ved å tegne en signatur. Dette gjør de ved å trykke på tegne-knappen under kontrakten. Da vil de ha muligheten til å tegne en signatur ved hjelp av musen sin eller ved hjelp av touchskjerm.
2. Brukere kan signere ved å laste opp en signatur. Dette gjør de ved å trykke på opplastingsknappen. Her kan de laste opp en signatur de har lagret på maskinen sin fra før.
3. Hvis en bruker har en signatur lagret på profilen sin fra før så vil den signaturen automatisk bli puttet inn i kontrakten og brukeren vil ikke trenger å signere kontrakten på nytt.

Etter at kontrakten er signert kan den bli sendt ut til mottaker. Den vil automatisk bli sendt til riktig mail-adresse når man trykker på Send-knappen.

Figure 4.7: Bekreftelse etter at kontrakt er sendt til Ragnhild Rygg

Signering for mottaker

Mottaker sin signaturside vil være veldig lik avsender sin. Den største forskjellen er at mottaker ikke har muligheten til å ha en signatur lastet opp på profilen sin, så de har kun valget mellom å tegne en signatur og å laste opp en signatur. Etter at mottakeren har signert kontrakten, vil de motta en kopi av den ferdig signerte kontrakten via e-post. Avsender vil kunne finne den ferdig kontrakten gjennom oversikten på hjemmesiden.

4.3 Administrative resultater

4.3.1 Fremdriftsplan

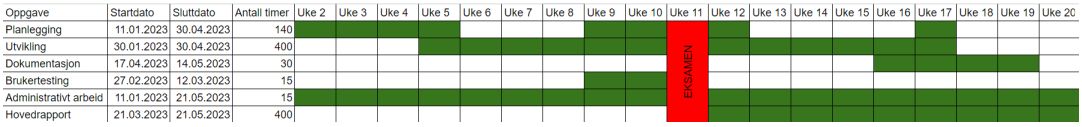


Figure 4.8: Fremdriftsplan

4.3.2 Tidsforbruk

	Planlegging	Administrativt arbeid	Dokumentasjon	Brukertesting	Hovedrapport	Utvikling	Sum
Mikkel	87,5	18,5	62,5	8,0	91,5	240,5	508,5
Oskar	112,5	18,0	66,5	6,5	125,5	175,0	504,0
Sum	200,0	36,5	129,0	14,5	217,0	415,5	1012,5
Planlagt	140	15	30	15	400	400	1000
Forskjell	60,0	21,5	99,0	−0,5	−183,0	15,5	12,5

Figure 4.9: Oversikt over tidsforbruk

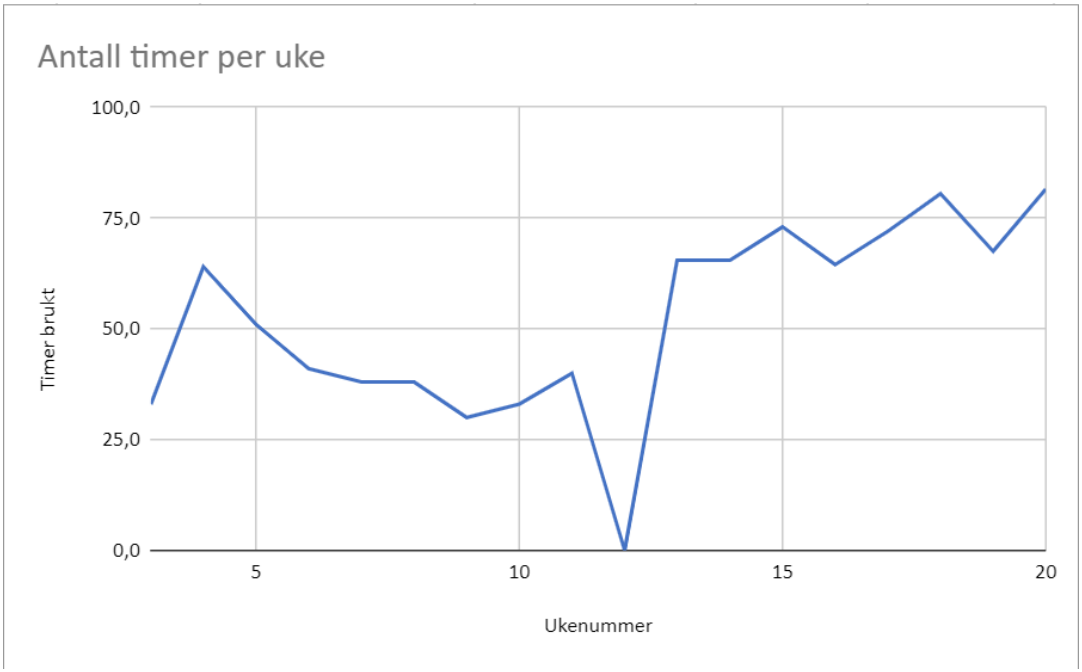


Figure 4.10: Antall timer per uke

Chapter 5

Diskusjon

5.1 Ingeniørfaglige resultater

5.1.1 Funksjonelle krav

5.1.1.1 Innlogging

Kravene rundt innlogging er oppfylt. Personer logger på med samme konto de bruker på medlemssystemet, og det er kun gruppeledere og medlemmer av Hovedstyret som kan logge på systemet.

5.1.1.2 Ulik funksjonalitet basert på rolle

Kravet om ulik funksjonalitet avhengig av rolle er oppfylt. En gruppeleder har begrenset tilgang til visning og administrering av kontrakter. De har kun tilgang til å se og administrere det nåværende styret i de gruppene de leder. Medlemmer av Hovedstyret har tilgang til å se og administrere kontrakter for både det nåværende styret i alle gruppene og for tidligere styrer i alle gruppene. De har også ekstra filtreringsmuligheter da mengden personer kan bli stor.

5.1.1.3 Generering av kontrakt

Kravet om generering av nye kontrakter er oppfylt. Gruppelederen og medlemmene av Hovedstyret har muligheten til å lage kontrakter som automatisk blir utfylt med oppdatert informasjon om personens navn, gruppe og rolle i den aktuelle gruppen. Denne informasjonen blir hentet fra medlemssystemet ved hjelp av NTNUI API-et.

5.1.1.4 Signering med digital signatur

Kravet om signering med digital signatur er oppfylt. Det var ønskelig at signering skulle skje med BankID, men etter å ha sett på alternativer og diskusjoner med produkteier ble det klart at dette ville være kostbart og oversteg det nåværende budsjettet. Som et alternativ tilbyr vi nå muligheten til å signere ved opplasting av signatur, eller ved å tegne en signatur med musen eller touchskjerm.

5.1.1.5 Utsending av generert kontrakt

Dette kravet har blitt oppfylt. Etter at en kontrakt har blitt signert kan den enkelt sendes. Når personen som genererte kontrakten sender den, vil den bli sendt til mottaker på mailen de er registrert med i medlemssystemet. Personen vil deretter bli opprettet i NDA-databasen, med en status som indikerer at de har mottatt, men ikke enda signert kontrakten.

Når mottaker sender inn kontrakten etter å ha signert, vil status for personen i NDA-databasen oppdateres til signert.

5.1.1.6 Oversikt over nåværende styremedlemmer i gruppe

Når man er logget inn som gruppeleder får man oversikt over alle nåværende styremedlemmer i gruppen(e) man er leder for. Oppdatert informasjon blir hentet fra medlemssystemet ved hjelp at NTNUI APIet. Dette kravet er dermed oppfylt.

5.1.1.7 Oversikt over signaturstatus

Alle personer som vises vil ha en signaturstatus ved siden av navnet sitt. Alle nåværende styremedlemmer i gruppen blir hentet fra medlemssystemet. Deretter varierer prosessen noe avhengig av om man er en gruppeleder eller om man er i Hovedstyret.

Hvis brukeren er en gruppeleder, blir det gjort et kall til NDA-databasen for å hente alle personer som allerede er hentet fra medlemssystemet, men også finnes i NDA-databasen. Disse vil erstatte de opprinnelige medlemmene som ble hentet, slik at riktig signaturstatus vises.

Hvis personen som er logget på er i Hovedstyret vil ikke bare personer som matcher de som er hentet fra medlemssystemet bli hentet, men alle personer som har signert en kontrakt for den gitte gruppen vil også hentes fra NDA-databasen.

Det er tre mulige signerings statuser:

- Personer som finnes i medlemssystemet, men ikke i NDA-databasen vil ikke ha mottatt en kontrakt, og deres status settes til usignert med et rødt ikon.
- Personer som finnes i NDA-databasen, men ikke har signert enda vil ha et gult ikon som indikerer at de har mottatt en kontrakt, men ikke signert denne enda. Det vil også vises en dato for når kontrakten ble sendt til personen.
- Personer som finnes i NDA-databasen og har signert vil ha et grønt ikon ved siden av navnet sitt. Her vil det også stå dato for når kontrakten ble signert.

Kravet om oversikt over signaturstatus er oppfylt.

5.1.1.8 Oversikt over gamle styremedlemmer som har signert

Personer i Hovedstyret vil kunne se alle som har signert en kontrakt for en gitt gruppe, uavhengig om de fortsatt er i styret, som nevnt i 5.1.1.7. Dette kravet er dermed oppfylt.

5.1.1.9 Filtrere på gruppe eller utvalg

Gruppeledere som er ledere i mer enn en gruppe eller utvalg vil ha tilgang til en dropdown meny der de kan velge mellom alle grupper eller utvalg de er leder for. Det samme gjelder personer som er i Hovedstyret, bare at Hovedstyret vil ha tilgang til alle grupper og utvalg, inkludert grupper og utvalg som er slettet fra medlemssystemet, da de har en kopi i NDA-databasen. Dette kravet er oppfylt.

5.1.1.10 Regelmessig sletting av data som ikke er nødvendig

Som nevnt i 5.1.1.5 så opprettes en person i NDA-databasen når en kontrakt blir sendt til den. Det å lagre informasjon om personer som ikke har signert kontrakten sin er unødvendig. Det kjøres derfor en sjekk en gang i døgnet som sjekker om det finnes noen personer i NDA-databasen som mottok kontrakten sin for ett år siden, men enda ikke har signert den. Hvis noen slike personer finnes så vil de slettes fra databasen. Dette kravet er dermed oppfylt.

5.1.1.11 Regenerering av kontrakt

Etter at en kontrakt er sendt til en person, kan det genereres og sendes ut en ny kontrakt for den aktuelle personen uavhengig av om den gamle er signert eller ikke. Den nye kontrakten vil da erstatte den gamle. Kravet om regenerering av kontrakter er dermed oppfylt.

5.1.1.12 Generering og sending til personer utenfor gruppestyrer

Gruppeleder og medlemmer av hovedstyret kan generere og sende ut kontrakter til personer som ikke er i gruppestyre. Dette gjøres ved å trykke på en knapp på hjemmesiden der man kan putte inn informasjon om personen kontrakten skal sendes til. Deretter vil kontrakten bli generert som vanlig med den riktige informasjonen, og sending vil fungere som med alle andre kontrakter. Dermed er dette kravet oppfylt.

5.1.1.13 Visning av signerte kontrakter

Både gruppeledere og medlemmer av hovedstyret vil kunne se de signerte kontraktene for alle personer de har tilgang til ved å trykke på "Vis taushetserklæring" på den aktuelle personen. Denne knappen er kun synlig for personer som har en ferdig signert kontrakt. Dette kravet er oppfylt.

5.1.1.14 Opplasting av signatur

Brukere kan laste opp, erstatte og se signaturen de har lastet opp. Dette kravet er oppfylt.

5.1.1.15 Automatisk signering med opplastet signatur

Når en person som har en signatur lagret på profilen sin genererer en kontrakt så vil signaturen de har lagret på profilen sin automatiske puttes inn i kontrakten. Dette kravet er dermed oppfylt.

5.1.1.16 Nedlasting av spesifikk kontrakt

Når man ser på en spesifikk kontrakt har man mulighet til å laste den ned ved å trykke på ikonet av en pil som peker nedover i høyre hjørne av pdfen. Dette er innebygd i pdf-viseren. Dette kravet er oppfylt.

5.1.1.17 Nedlasting av flere kontrakter

Det er en "Last ned kontrakter"-knapp på hjemmesiden. Når denne knappen trykkes på så lastes alle ferdig signerte kontrakter i den respektive gruppen ned. Her gjelder de samme reglene som visning - gruppeledere har kun tilgang til kontraktene til det nåværende styret, mens hovedstyret har tilgang til signerte kontrakter for gamle styremedlemmer også. Dette kravet er dermed oppfylt.

5.1.1.18 Personsøk

Det er enkelt å søke etter personer ved hjelp av søkefeltet på hjemmesiden. Søket blir gjort ved hjelp av fuzzy-søk, noe som gir treff selv om strengen ikke passer helt med det du søker etter. Dette kravet er oppfylt.

5.1.1.19 Filtrering på signaturstatus

Det er mulig å kun vise personer som ikke enda har signert en taushetserklæring ved hjelp av "Vis kun usignerte"-knappen på hjemmesiden. Dette kravet er oppfylt.

5.1.1.20 Filtrering på signeringsår

Når et medlem av Hovedstyret er logget på vil de ha mulighet til å filtrere på signeringsår ved hjelp av knappen på hjemmesiden. Det er kun medlemmer av hovedstyret som ser denne knappen, da gruppeledere kun vil se det nåværende styret (som regel 7-12 personer) til sin gruppe eller utvalg. Dette kravet er oppfylt.

5.1.1.21 Unik URL for signering

Mottakere vil få en unik link for signering av kontrakten sin på epost. Denne linken er unik og gir dem muligheten til å signere kontrakten sin uten å logge inn, da det kun er gruppeledere og medlemmer av hovedstyret som skal kunne logge inn. Etter at personen har signert kontrakten gjennom den unike linken, vil linken deaktiveres. Dette kravet er oppfylt.

5.1.1.22 Mottaker skal få en kopi av den signerte kontrakten

Etter at mottaker har signert kontrakten vil de motta en kopi av den ferdig signerte kontrakten på epost. Dette kravet er oppfylt.

5.1.2 Ikke-funksjonelle krav

5.1.2.1 Brukbarhet

Det har blitt gjennomført brukertester der faktiske brukere har fått utforske systemet uten assistanse fra utviklerne, med mål om å verifisere brukervennligheten. Utviklerne samlet også inn tilbakemeldinger fra disse testene og har gjort justeringer basert på dem.

Systemet er gjort responsivt og er brukbart på en rekke forskjellige enheter, inkludert datamaskiner og mobile enheter. Dette gjøres for eksempel ved å pakke sammen alle filtreringsknappene og alternativene i menyer istedenfor selvstendige knapper når skjermen er mindre enn en bestemt størrelse.

Systemet har blitt testet på de mest populære nettleserene på markedet, og virker som forventet på samtlige av de. Basert på alle disse punktene anses kravet som oppfylt.

5.1.2.2 Dokumentasjon

Det har blitt skrevet en omfattende brukermanual på produkteier sin interne wiki som har blitt lest gjennom og godkjent av produkteier. Det er også kommentarer i koden som dokumenterer alle steder der det er nødvendig med en forklaring. Teamet anser også dette kravet som oppfylt.

5.1.2.3 Vedlikehold

Systemet har blitt utviklet med populære og lett tilgjengelige teknologier og kodebasen er godt dokumentert. Dette var alt produkteier ønsket seg fra vår side når det kom til krav om vedlikehold, så dette kravet har blitt oppfylt.

5.1.2.4 Testing

Systemet har gjennomgående testing på frontend, som effektivt adresserer de fleste potensielle scenarioer. Imidlertid er det noe begrenset testing på backend-siden, da teamet har støtt på utfordringer med oppsett av databasemocking. På nåværende tidspunkt omfatter testene kun autentisering og innlogging, ettersom dette ikke krever opprettelse av en simulert database. Derfor betraktes dette kravet som delvis oppfylt.

5.1.2.5 Kontinuerlig integrasjon

Systemet har fått satt opp kontinuerlig integrasjon ved hjelp av GitHub Actions. Dette fungerer som det skal når det kommer til å bygge både front- og backend og for å kjøre backend testene. Det var litt problemer med å kjøre frontend testene gjennom dette systemet, og det er noe teamet ikke fikk til å fungere. Dette kravet betraktes dermed som delvis oppfylt.

5.1.3 Refleksjon rundt sluttproduktet

5.1.3.1 Styrker ved løsningen

Systemet dekker i stor grad alle behovene til produkteieren. Det forenkler betydelig prosessen med å generere og distribuere taushetserklæringer i forhold til nåværende praksis, samtidig som det bidrar til å gjøre lagringen av disse dokumentene mer oversiktlig og pålitelig. Denne forbedringen oppnås ved å integrere og benytte informasjon fra medlemssystemet.

Resultatene fra brukertester bekrefter at det utviklede systemet er intuitivt og enkelt å bruke, uten krav til spesiell teknisk kompetanse. Effekten av å ha et system som betydelig forenkler prosessen, samtidig som det er brukervennlig, vil føre til at terskelen for å ta personvern på alvor innen idrettsorganisasjonen senkes. Dette vil resultere i en høyere prioritering av personvern.

5.1.3.2 Svakheter ved løsningen

Ettersom systemet er utviklet fra bunn var det aldri meningen at systemet skulle stå fullstendig ferdigstilt ved prosjektslutt. Dermed er det naturlig at enkelte svakheter etterlates for videreutviklere å løse. Alle svakheter er tydelig kommunisert til produkteier slik at de kan tas opp ved senere tidspunkter.

Testdekningen av tjenerkoden er lav. End-to-end testene kan enkelt modifiseres for å teste mer av tjeneren, men dette er per nå ikke implementert i stor grad. Frontend testene kjører heller ikke i CI pipelinen, som gjør testprosessen noe mindre automatisert.

Det er også enkelte mindre bugs i systemet som teamet er klar over. Dette er hovedsakelig bugs som omhandler brukergrensesnittet, og ingenting som hindrer selve funksjonaliteten av systemet. Alle oppdagede bugs er lagt inn som godt beskrevede issues i prosjektets issueboard.

At signeringene håndteres med håndsignatur istedenfor digital signatur gjennom f.eks BankID er også en grunnleggende svakhet. Signering gjennom BankID ville vært den optimale løsning iforhold til sikkerhet og legitimitet av signeringene. Som nevnt tidligere var dette ikke en beslutning som var opp til teamet og ta, og derfor ikke noe vi kunne endre.

5.1.3.3 Valg av teknologier

Teamet er ekstremt fornøyd med valget av teknologier som ble benyttet. Til tross for at teamet hadde relativt begrenset erfaring med samtlige teknologier på forhånd, møtte vi overraskende få utfordringer underveis i prosjektet.

React var veldig greit å komme i gang med. Det var også svært gunstig at flere av de andre mikrotjenestene i NTNUI-økosystemet allerede var utviklet med React. Dette gjorde at vi kunne hente inspirasjon og lære av eksisterende løsninger når vi trengte det. Vi hadde som mål å oppnå et enhetlig design på tvers av mikrotjenestene, og det var fantastisk å kunne ta inspirasjon fra sider som allerede var like på tvers av tjenestene. For eksempel kunne vi studere og implementere en konsistent loginside.

Ved å bruke Tailwind gjorde vi arbeidet med utseendet betydelig enklere enn om vi skulle ha gått den tradisjonelle veien med ren CSS. Tailwind ga oss et sett med forhåndsdefinerte stiler og klasser som vi kunne bruke direkte i koden. Dette førte til en raskere og mer effektiv utviklingsprosess, da vi slapp å skrive og vedlikeholde store mengder egen CSS-kode. Med Tailwind kunne vi enkelt tilpasse stiler og lage en visuelt tiltalende brukeropplevelse.

Vi opplevde også en smidig start med Node.js. Akkurat som med React hadde et par andre mikrotjenester i NTNUI-økosystemet brukt Node. Vi undersøkte disse når vi skulle implementere lignende funksjonalitet. Dette ga oss muligheten til å dra nytte av andre sine erfaringer, samt passe på at det er et gjenkjennbart oppsett av prosjektet. Et eksempel på dette var når vi håndterte autorisering av endepunkter ved hjelp av middleware. Ved å studere hvordan andre mikrotjenester hadde løst dette, kunne vi implementere et sikkert og pålitelig autentiseringssystem.

Når det gjaldt databasen valgte vi MongoDB, og det viste seg å være enkelt å sette opp og bruke. Vi kunne raskt få alt opp og kjøre, og det var få hindringer underveis. Vi støtte imidlertid på en utfordring når vi skulle utføre backend-testing som var avhengig av faktiske databasekall. Å håndtere mocking av disse kallene var noe mer komplisert, og teamet rakk ikke få dette til å fungere. Til tross for denne spesifikke utfordringen, var det ellers en smertefri opplevelse å arbeide med MongoDB, og vi satte pris på fleksibiliteten og ytelsen det ga.

Alt i alt er teamet svært fornøyd med valgene når det kom til teknologier, og mener at de valgene som ble tatt gjorde det mulig å bygge et stabilt og vedlikeholdbart system. Vi lærte også mye om teknologiene vi brukte underveis, og dette vil være nyttig kunnskap for oss i fremtidige prosjekter.

5.1.3.4 Personvern

Teamet har konsekvent prioritert personvern gjennom hele prosessen. Vi har sørget for at ingen unødvendige personopplysninger blir lagret. I NDA-databasen blir kun nødvendig informasjon om enkeltpersoner lagret, og personer uten en ferdig signert kontrakt blir automatisk slettet ett år etter at kontrakten ble sendt ut. Dette skyldes at det ikke er nødvendig å beholde opplysninger om enkeltpersoner som ikke har fullført signeringsprosessen.

5.2 Administrative resultater

5.2.1 Fremdriftsplan

Fremdriftsplanen var et essensielt verktøy som ga teamet en omfattende oversikt over hvordan prosjektperioden skulle forløpe. For å sikre en grundig planlegging, ble den delt opp i kategorier som var lik de som ble brukt i timeføringslistene. Vi brukte tidsestimater til å forutsi hvor mye tid teamet ville bruke på de ulike aktivitetene. I fremdriftsplanen var det noen overordnede tidsfrister som teamet prøve å holde seg til. Dette sørget for at teamet alltid hadde konkrete mål å jobbe mot og at vi holdt oss på riktig kurs.

Selv om det var nødvendig å justere planen underveis på grunn av endringer i prioriteringer, klarte vi likevel å fullføre alle aktivitetene til riktig tid. Teamet er fornøyd med fremdriftsplanen, og merket at det var et viktig verktøy for planlegging av prosjektperioden.

5.2.2 Tidsforbruk

Det var undervisning i et annet fag i første halvdel av semesteret, noe som begrenset hvor mye tid som ble brukt på prosjektet. Antall timer per uke har holdt seg ganske stabilt hvis man separer mellom før og etter eksamen. Etter eksamen i dette faget økte antall timer i uken betydelig, helt frem til innleveringsfristen.

Når det gjelder bruken av disse timene, var det noe variasjon mellom planen og det endelige resultatet. Det var spesielt betydelig færre timer som ble brukt enn planlagt på hovedrapporten, noe som var forventet siden oppgaven var en utviklingsoppgave. Det ble imidlertid brukt flere timer enn planlagt på både dokumentasjon og planlegging, spesielt på dokumentasjon. Teamet hadde undervurdert mengden dokumentasjon som var nødvendig, og de gjorde det meste av dokumentasjonen sammen, noe som førte til at tiden som ble brukt ble nesten doblet. Dette ble gjort for å sikre at begge medlemmene av teamet var enige.

5.2.3 Arbeidsmetode

Teamet fulgte ikke noen spesifikke utviklingsmetoder, da det ikke føltes nødvendig når det kun var to stykker. Til tross for dette var det et forsøk på å ha struktur i arbeidet, og inspirasjon ble hentet fra smidige prosessrammeverk, som for eksempel Scrum.

5.2.3.1 Versjonskontroll og code reviews

Teamet brukte versjonskontroll aktivt under utviklingen. Plattformen som ble valgt å gjøre dette på var GitHub. For å forbedre oversikten og forebygge konflikter, implementerte vi en praksis med å jobbe med kun én oppgave per gren (branch), ofte referert til som "feature branching" eller "issue branching".

Et annet motiv bak denne tilnærmingen var å gjøre pull requests mer håndterbare og effektivisere kodegjennomgangen (code reviews). Teamet hadde en streng praksis hvor alle endringer i kodebasen måtte bli gjennomgått, testet og godkjent av den andre parten, samt at alle CI-jobber måtte vellykket passere, før den kunne gjennomføres. Derfor ble alle endringer lagt frem gjennom pull requests, som så ble gjennomgått av den andre parten. CI-jobbene blir automatisk kjørt når en pull request blir åpnet. Dette ga den andre parten muligheten til å komme med forslag til endringer, stille spørsmål eller godkjenne endringen. Prosjektet var konfigurert slik at ingen endringer kunne utføres i kodebasen før de ble godkjent av den andre parten og CI-jobbene var vellykket passert.

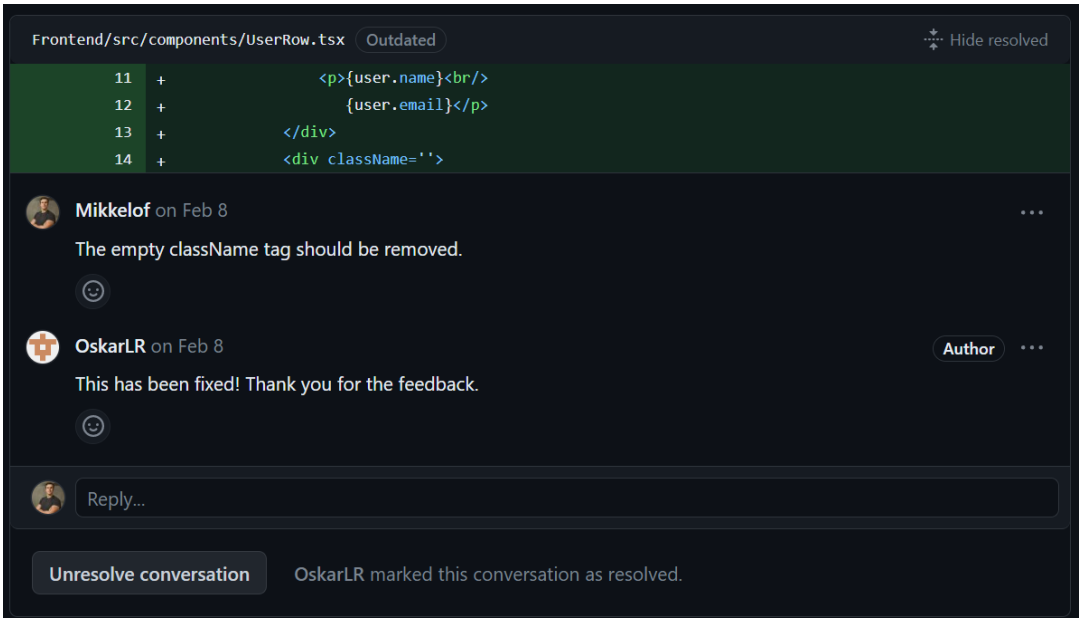


Figure 5.1: Eksempel på en kommentar fra en code review i en pull request

Når en pull request ble godkjent og branchen skulle slås sammen med kodebasen, brukte vi squashing for å gjøre endringshistorikken i kodebasen mer oversiktlig.

Teamet er veldig fornøyde med bruken av git. Det oppsto naturligvis et par git-konflikter, men ingen store problemer.

5.2.3.2 Issue board

Teamet benyttet GitHub's issue board aktivt for å effektivt organisere arbeidet. Issue boardet ble brukt til å holde oversikt over pågående oppgaver, prioritere dem og vite hvem som var ansvarlig for hver oppgave. Nye oppgaver ble jevnlig lagt inn i boardet når de oppsto, og de ble merket med merkelapper som anga prioritet og temaet for oppgaven. Disse oppgavene ble grundig beskrevet, og ofte inkluderte de bilder eller videoer for å tydeliggjøre hva som skulle gjøres eller hva som var feil. Når en person begynte å jobbe med en oppgave, tildelte de oppgaven til seg selv i issue boardet for å unngå at den andre personen startet på den samme oppgaven.

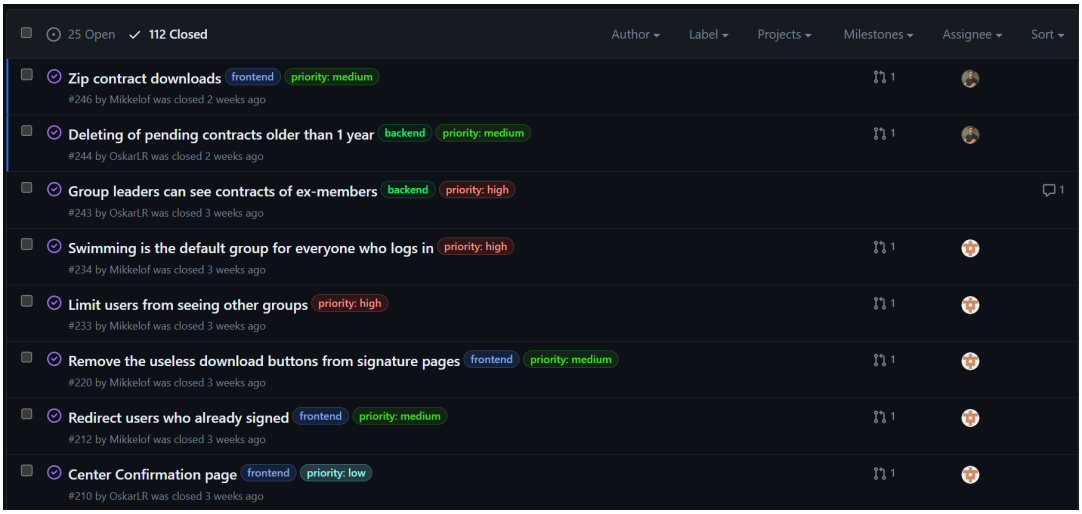


Figure 5.2: Issue board

5.2.4 Gruppearbeid

I løpet av prosjektperioden har vi oppnådd en velbalansert kombinasjon av individuelle og felles arbeidsøkter. Dette gjelder særlig programmeringen. De mer utfordrende oppgavene som krever kontinuerlig dialog, har blitt gjennomført ved hjelp av parprogrammering. Denne tilnærmingen innebærer at en person er ansvarlig for å skrive koden, selv om begge teammedlemmene er fysisk til stede, samtidig som vi opprettholder en kontinuerlig dialog gjennom hele prosessen. For de enklere oppgavene har individuelt arbeid fungert bra, da det gir oss større fleksibilitet. Når det gjelder dokumentasjon, har vi i stor grad arbeidet sammen fysisk.

Det har også vært viktig for teamet å sikre likestilling mellom begge parter, og vi har hatt en felles beslutningsprosess der vi blir enige før avgjørelser tas.

Chapter 6

Konklusjon og videre arbeid

6.1 Konklusjon

Prosjektperioden forløp stort sett som planlagt. Ved å opprettholde en god kommunikasjon med produkteieren, utføre brukertesting med faktiske sluttbrukere, nøye planlegging og effektivt samarbeid internt i teamet, oppnådde vi et resultat som både teamet og produkteieren er svært fornøyde med.

I tillegg til å oppfylle NTNUI sine behov, har utviklingen av taushetserklæringssystemet også bidratt til læring og kompetanse for teamet. Gjennom dette prosjektet har teamet styrket sine ferdigheter innen systemdesign og implementering, samt utviklet bedre forståelse for personvern og det å produsere systemer som skal brukes av faktiske brukere.

Brukersamtalene viste at de største utfordringene med fysisk signering omhandlet generering, distribuering og tilbakefinning av taushetserklæringer.

Tilbakemeldingene fra NTNUI og deres brukere, indikerer at de digitale løsningene som ble implementert adresser de største utfordringene på en gunstig måte. Teamet håper at dette systemet vil fortsette å tjene NTNUI godt i fremtiden, og at det vil være et verdifullt verktøy for å opprettholde konfidensialitet og ivareta personvernet til alle involverte parter.

6.2 Videre arbeid

Alle de opprinnelige målene som ble satt i oppstartfasen er nådd, og teamet anser derfor systemet som ferdigutviklet. Selv om dette er tilfellet ser teamet flere muligheter til forbedring og videreutvikling.

6.2.1 Andre typer kontrakter

Produkteier uttrykte en potensiell videreutvikling angående muligheten til å generere og lagre ulike typer kontrakter gjennom det eksisterende systemet. I tillegg til taushetserklæringer, kunne det være en spennende utvidelse å inkludere funksjonalitet for trenerkontrakter og potensielt andre typer kontrakter. Dette ville være et naturlig neste steg for systemet, og ville åpnet det for mer allsidig bruk.

6.2.2 Gjenstående oppgaver på issueboard

Det gjenstår fremdeles en del oppgaver på issueboardet på GitHub. Disse oppgavene har ikke særlig høy prioritet, da de hovedsakelig omhandler mindre forbedringer, refaktorisering eller ubetydelige feil. Alle disse oppgavene er grundig dokumentert med kommentarer og bilder/videoer som viser hva som er feil der det er relevant. Her er noen eksempler på gjenværende oppgaver:

- Det bør inkluderes informasjon om hvem som genererte og sendte kontrakten i e-posten mottakeren mottar.
- Man burde kunne generere og sende kontrakter til flere enn en person om gangen.

- Hvis man filtrerer etter et spesifikt år, og deretter endrer skjermstørrelsen til en betydelig større skjerm (for eksempel fra mobil til PC, slik at filtermenyen blir til separate knapper igjen), vil personene ikke lenger være filtrert etter året.

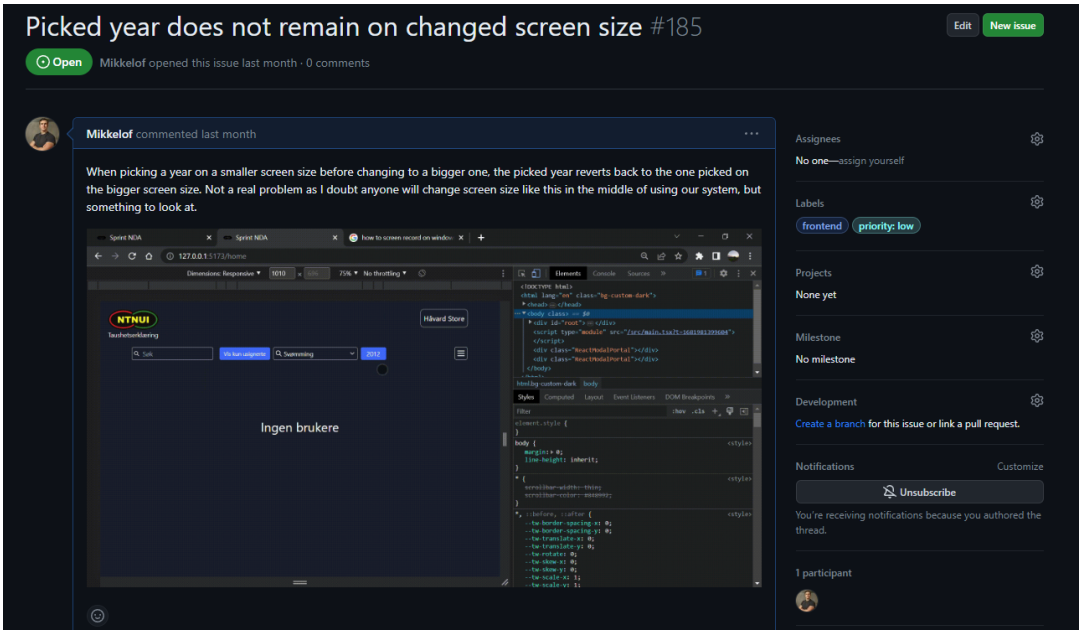


Figure 6.1: Eksempel på hvordan et gjøremål vi har lagt igjen ser ut. Her er det en video som tydelig viser hva feilen er og hvordan man gjenskaper den.

6.2.3 Test-dekning i backend

Som nevnt i kapitel 5 er ikke test-dekningen i backend like omfattende som teamet skulle ønske at den var. Dette er noe som burde jobbes videre med på et senere tidspunkt.

6.2.4 Kjøre frontend tester gjennom CI

Det ble også nevnt i kapittel 5 at teamet slet med å få kjørt frontend testene gjennom kontinuerlig integrasjon. Disse testene er ferdiglaget og kjører lokalt, det er kun gjennom CI det var problemer med å kjøre dem. Dette er noe som burde håndteres på et senere tidspunkt.

6.2.5 Pagination

Teamet bestemte seg for å ikke prioritere pagination når det kom til å hente personer fra NDA-databasen, ettersom det for øyeblikket bare er mulig å vise én gruppe av gangen. Hver gruppe har kun ca 8-12 styremedlemmer, og kun noen av disse byttes ut hvert år. Det vil være flere år til mengden personer i en gitt gruppe er stor nok til at pagination har noen reell effekt på systemet, men det kan være noe som bør vurderes senere.

Samfunnspåvirkning

Personvern er et emne som blir hyppig diskutert i dagens samfunn. Det tar ikke lang tid mellom hver gang det blir rapportert om personvernsrelaterte saker i mediene, enten det er lekkasjer av informasjon eller innsamling av informasjon på uetiske måter. Dette er et viktig tema som ikke bør nedprioriteres, spesielt ikke av en organisasjon av NTNUIs størrelse. Teamet tror at dette systemet forenkler prosessen så mye at det ikke lenger vil nedprioriteres.

NTNUI har siden 1910 vært en sentral del av studentmiljøet i Trondheim. De har vært en av de fremste pådriverne for inkludering og fremming av fysisk aktivitet blant studenter, og har gjort studietiden betydelig bedre for utallige studenter. Som en ideell organisasjon som drives utelukkende av frivillige, er de avhengige av alle som er villige til å bidra for å opprettholde driften av klubben. Ved å utvikle dette systemet føler teamet at de har spilt en rolle i å effektivisere driften av en klubb som har hatt og vil fortsette å ha en betydelig innvirkning på den fysiske og mentale helsen til utallige studenter.

Det vil også være litt papirbesparelse, men denne er ganske neglisjerbar da det generelt sett er snakk om under 1000 A4-ark i året.

Referanser

1. «Microservice Architectures: More than the Sum of Their Parts?» IONOS Digital Guide, https://www.ionos.com/digitalguide/w development/microservice-architecture/. Åpnet 29. april 2023.

2. MongoDB Documentation. https://www.mongodb.com/docs/. Åpnet 1. mai 2023.

3. «About». Node.js, https://nodejs.org/en/about. Åpnet 3. mai 2023.

4. About Npm | Npm Docs. https://docs.npmjs.com/about-npm/. Åpnet 3. mai 2023.

5. Affairs, Assistant Secretary for Public. Wireframing. https://www.usability.gov/how-to-and-tools/methods/wireframing.html. Åpnet 3. mai 2023.

6. Atlassian. «What Is Version Control | Atlassian Git Tutorial». Atlassian, https://www.atlassian.com/git/tutorials/what-is-version-control. Åpnet 3. mai 2023.

7. «Axios». Npm, https://www.npmjs.com/package/axios. Åpnet 3. mai 2023.

8. Express - Node.Js Web Application Framework. https://expressjs.com/. Åpnet 4. mai 2023.

9. «Guidelines for Better Pull Requests | Kenneth Skovhus». Guidelines for Better Pull Requests | Kenneth Skovhus, https://www.skovhus.dev/blog/better-pull-requests. Åpnet 4. mai 2023.

10. «Hello World». GitHub Docs, https://ghdocs-prod.azurewebsites.net/en/get-started/quickstart/hello-world. Åpnet 4. mai 2023.

11. JavaScript Component Testing and E2E Testing Framework | Cypress. https://www.cypress.io/. Åpnet 4. mai 2023.

12. JavaScript With Syntax For Types. https://www.typescriptlang.org/. Åpnet 4. mai 2023.

13. Jest. https://jestjs.io/. Åpnet 4. mai 2023.

14. «Nodemailer». Npm, https://www.npmjs.com/package/nodemailer. Åpnet 4. mai 2023.

15. «Nodemon». Npm, https://www.npmjs.com/package/nodemon. Åpnet 4. mai 2023.

16. «Non-Disclosure Agreement (NDA) Explained, With Pros and Cons». Investopedia, https://www.investopedia.com/terms/n/n Åpnet 4. mai 2023.

17. «Ntnui-Tools». Npm, https://www.npmjs.com/package/ntnui-tools. Åpnet 4. mai 2023.

18. «Pdf-Lib». Npm, https://www.npmjs.com/package/pdf-lib. Åpnet 4. mai 2023.

19. React. https://react.dev/. Åpnet 4. mai 2023.

20. «React-Router». Npm, https://www.npmjs.com/package/react-router. Åpnet 4. mai 2023.

21. «React-Signature-Canvas». Npm, https://www.npmjs.com/package/react-signature-canvas. Åpnet 4. mai 2023.

22. «Tags · Nodejs/Node-v0.x-Archive». GitHub, https://github.com/nodejs/node-v0.x-archive. Åpnet 4. mai 2023.

23. Tailwind CSS - Rapidly Build Modern Websites without Ever Leaving Your HTML. https://tailwindcss.com/. Åpnet 4. mai 2023.

24. «Taushetserklæring | IP-leksikon». Bryn Aarflot, https://baa.no/kunnskapssenter-for-immaterialrett/ip-leksikon/taushetserkl Åpnet 4. mai 2023.

25. User Testing: The Ultimate Guide. https://blog.hubspot.com/service/user-testing. Åpnet 6. mai 2023.

26. «Uuid». Npm, https://www.npmjs.com/package/uuid. Åpnet 6. mai 2023.

27. «Vite». Npm, https://www.npmjs.com/package/vite. Åpnet 6. mai 2023.

28. «What is Figma?» Figma.com, https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma. Åpnet 6. mai 2023.

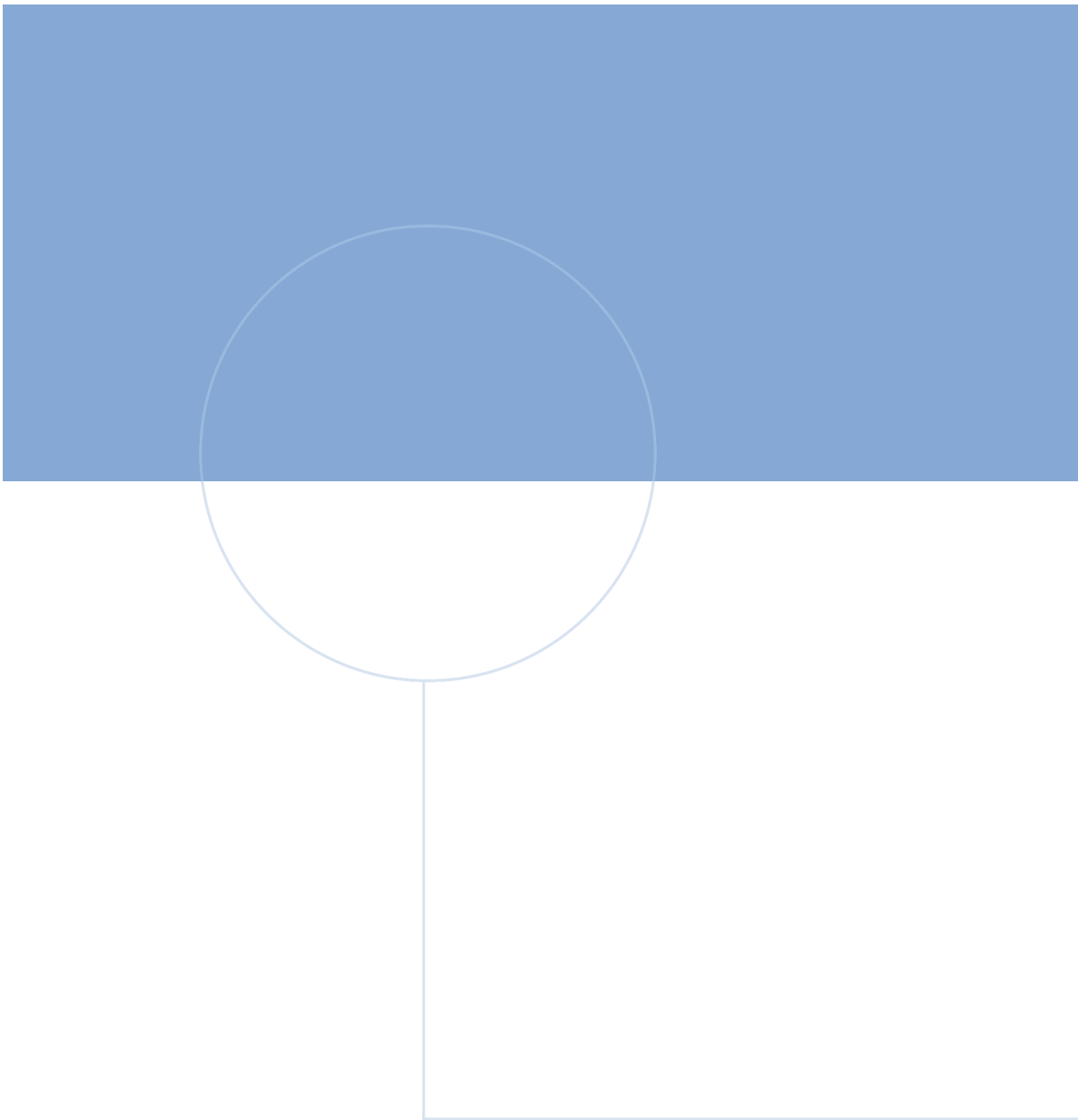
29. «What Is Pagination? – TechTarget Definition». WhatIs.Com, https://www.techtarget.com/whatis/definition/pagination. Åpnet 6. mai 2023.

30. «About Continuous Integration». GitHub Docs, https://ghdocs-prod.azurewebsites.net/en/actions/automating-builds-and-tests/about-continuous-integration. Åpnet 7. mai 2023.

31. «Squashing Commits in GitHub Desktop». GitHub Docs, https://ghdocs-prod.azurewebsites.net/en/desktop/contributing-and-collaborating-using-github-desktop/managing-commits/squashing-commits-in-github-desktop. Åpnet 7. mai 2023.

32. «Difference between System Testing and End-to-End Testing». GeeksforGeeks, https://www.geeksforgeeks.org/difference-between-system-testing-and-end-to-end-testing/. Åpnet 7. mai 2023.

33. «Unit Testing | Software Testing». GeeksforGeeks, https://www.geeksforgeeks.org/unit-testing-software-testing/. Åpnet 7. mai 2023.



NTNU

Kunnskap for en bedre verden