NTNU
Kunnskap for en bedre verden

Bacheloroppgave i Webutvikling

System for oversikt over flere Scrum-prosjekter i parallell

Forfattere:

Martine Jacobsen

Silje Jeanette Fruseth Lien

Audun Meek Olsen

Susanne Waaler Stenshagen

16.05.2018

Simasu

"simasu" (します)

Et japansk ord gruppen har valgt å navngi løsningen som presenteres i rapporten. Ordet oversetter til "I'll do it" på engelsk. Ordet er også et anagram formet av forbokstaver i hvert gruppemedlem sitt fornavn; Silje, Martine, Audun og Susanne. Vår visjon er å utforme en løsning som opprettholder en god arbeidsmoral – en holdning som frasen "I'll do it!" underbygger.

Sammendrag

Tittel	System for oversikt over flere Scrum-prosjekter i parallell
Dato	16.05.2018
Deltakere	Martine Jacobsen Silje Jeanette Fruseth Lien Audun Meek Olsen Susanne Waaler Stenshagen
Veileder	Eivind Arnstein Johansen
Oppdragsgiver	Escio AS
Stikkord	Scrum, prosjekter i parallell, brukersentrert design, brukstesting/brukertesting, prototype, web-applikasjon
Antall sider	101
Antall vedlegg	30
Tilgjengelighet	Åpen

Kort beskrivelse av oppgaven:

Denne oppgaven er gitt etter et ønske fra Escio AS om bedre oversikt over bruken av menneskelige ressurser på tvers av flere samtidige prosjekter. Formålet med oppgaven er å utvikle et system som skal gi oversikt over samtidige prosjekter og kartlegge ressursfordeling. Basert på brukerbehov foreslås et sett med løsninger som sammen skal gjøre det mulig å bedre utnytte menneskelige ressurser ved gjennomføring av flere parallelle Scrum-prosjekter. Systemet skal utvikles som en moderne web-applikasjon.

For å få større innsikt i utviklingsmetoden Scrum er denne metoden brukt som grunnlag for egen prosjektstyring. Rapporten presenterer, med bakgrunn i teori og litteratur, ulike metoder for design- og utviklingsprosess som har vært avgjørende for fremgang og struktur på bachelorprosjektet. Det har resultert i et forslag til et system som gir Escio oversikt over allokering av ressurser for flere parallelle Scrum-prosjekter, samt kartlegging av ressursbelastning innad i bedriften. Rapporten presenterer en detaljert utviklingsprosess, evaluering av resultatet og forslag til videre utvikling.

Abstract

Title	System for overview of multiple Scrum projects in parallel
Date	16.05.2018
Participants	Martine Jacobsen Silje Jeanette Fruseth Lien Audun Meek Olsen Susanne Waaler Stenshagen
Supervisor	Eivind Arnstein Johansen
Employer	Escio AS
Keywords	Scrum, projects in parallel, user-centered design, usability testing, prototype, web application
Number of pages	101
Number of appendix	30
Availability	Open

Short description of the bachelor thesis:

The bachelor thesis is based on a request from Escio AS which describes the need for a system which can enhance their existing ways of allocating human resources across multiple simultaneous projects. The purpose of the thesis is to develop a system which can provide an overview of projects running in parallel and map resource allocation for better strategizing. Based on user needs, a set of solutions is suggested that together will avoid strain and better utilize human resources throughout several parallel Scrum projects. The system will be developed as a modern web application.

In order to gain insight into the development method Scrum, this method is used as a foundation for the project management. The report presents various methods based on theory and literature for design and development. Such methods have been decisive for the progress and structure of the project. This has resulted in a proposal for a system that gives Escio an overview of resources for multiple Scrum projects in parallel, as well as a mapping of resource allocation within the company. The report presents a detailed development process, evaluation of the results and proposals for further development.

Forord

Dette prosjektet er vår avsluttende bacheloroppgave i webutvikling ved NTNU i Gjøvik, og tar for seg utvikling av et system for oversikt og kartlegging av ressursfordeling for prosjekter i parallell. Vi har arbeidet jevnt med oppgaven siden semesterstart, og har hatt et godt samarbeid innad i gruppen.

Vi vil benytte anledningen til å takke oppdragsgiver, Escio, for godt samarbeid gjennom prosjektarbeidet. Vi vil også takke vår veileder, Eivind Arnstein Johansen, for gode veiledninger gjennom hele våren. En stor takk rettes også til andre som har bidratt med hjelp og støtte underveis i prosessen.

Martine Jacobsen, Silje Lien, Audun Olsen og Susanne Stenshagen
Gjøvik, 16.05.2017

Innholdsliste

Tabell- og figurliste

Tabeller

Figurer

1. Innledning

1.1 Introduksjon

I dette innledningskapitlet vil vi presentere bakgrunnen for prosjektet og hvordan vi har valgt å angripe problemstillingen. Innholdet i dette kapitlet legger grunnlaget for videre innhold i rapporten.

1.2 Prosjektbeskrivelse

1.2.1 Bakgrunn

Bachelorprosjektet er gitt av selskapet Escio som blant annet leverer tekniske løsninger innenfor web og applikasjoner. Escio bruker samarbeidsverktøyet *Atlassian Jira* for å organisere sine prosjekter. *Jira* brukes til å holde oversikt på fremdrift i prosjekter og er levert av selskapet *Atlassian*, som tilbyr et mangfoldig utvalg av team-orienterte verktøy. Jira er deres løsning på organisering av smidige prosjekter.

Escio benytter seg av den smidige arbeidsmetoden Scrum og opplever i dag at Jira ikke tilfredsstiller behovet for belastningsoversikt. Selv har de ikke funnet en tilleggsløsning som gir den oversikten de ønsker på ressursbelastning.

1.2.2 Oppgavebeskrivelse

Som nevnt over benytter Escio seg – per skrivende dato – av Atlassian Jira, som selv hevder å være det mest utbredte verktøyet for smidige team. Det brukes til å holde oversikt over de ulike faktorene som inngår i fremdriften og utfordringer rundt et smidig prosjekt. En av disse faktorene er muligheten for å tilegne oppgaver til ressurspersoner. Escio ønsker å se en mer helhetlig oversikt for hvordan disse er disponert innad i prosjekter, samt en oversikt der belastning på ressurspersoner er satt opp mot hverandre. Escio ønsker ikke en alternativ løsning til Jira, men en *tilleggspakke* som avdekker svakheter ved allokering av ressurser i prosjektene sine. Dette mener de er en av manglene ved Jira.

1.2.3 Problemstilling

Oppgaven er å utvikle et system som skal gi oversikt, kartlegge ressursfordeling og foreslå løsninger for bedre allokering av ressurser for prosjekter i parallell. Systemet skal utvikles som en moderne webapplikasjon.

1.2.4 Formål

Formålet er å utvikle en løsning som sikrer optimal ressursallokering for effektiv, lønnsom og fornuftig arbeidsprosess hos Escio. Løsningen skal gi prosjektleder et oversiktsbilde for å kunne gjøre riktig valg i arbeidsfordeling, både for bedriften og for de ansatte.

1.3 Avgrensninger og rammer

1.3.1 Prosjektavgrensninger

Det er vesentlig å avgrense omfanget til prosjektarbeidet, på bakgrunn av begrensninger vi må forholde oss til. Vi må blant annet ta hensyn til tidsrammer og høye kompetansekrav. Original oppgavebeskrivelse skildrer en Node-basert web-applikasjon som integreres med Jira og bygger på et JavaScript-rammeverk. Det foreslås videre i oppgaveteksten at kunstig intelligens kan være en medspiller for å foreslå omdisponering av ressurser. Oppgaveteksten antas å være formulert omfattende for å inkludere flere studieprogrammer. Grunnet oppgaven sitt store omfang, og tidlig dialog med oppdragsgiver, er det en gjensidig enighet om å redefinere arbeidskravene. Å realisere en kobling mot Jira, fasilitert av server-funksjonalitet basert på Node.js, er derfor et langsiktig mål. Fokus i dette prosjektet omhandler å konseptualisere en løsning, samt overlevere et godt teknisk fundament for videre utvikling. Teknologier som Node.js vil fortsatt være viktig for å utvikle en fungerende løsning, og vi forklarer hvordan vi ser for oss å anvende denne teknologien i *kapittel 7 – Videreutvikling*.

1.3.2 Kostnader

Det er ikke avsatt budsjett for utarbeidelse av dette prosjektet. Alt som eventuelt påløper av kostnader må gruppen selv betale, og vil være gjort i samtykke med alle gruppemedlemmer.

For å få god innsikt og forståelse i smidige metoder, har gruppen – etter ønske fra oppdragsgiver – valgt å følge metoden Scrum i bachelorprosjektet. For å få en dypere forståelse av hva oppdragsgiver ønsker en løsning på, velger vi å bruke Atlassian Jira for å komplimentere arbeidsmetoden Scrum, og har derfor blitt enige om å kjøpe abonnement til denne tjenesten. Andre kostnader vil være reiseutgifter for møter med oppdragsgiver. Dette anser vi som eneste konkrete kostnader i forbindelse med prosjektet, da andre programvarer og miljø vi vil benytte oss av er gratis.

Escio skrev i deres originale utkast av oppgaven et ønske om bruk av webteknologien Node.js som grunnlag for å utforme løsningen. JavaScript, et språk som før kun ble brukt til klientsiden av en webløsning, kan nå med Node.js anvendes til å utvikle back-end funksjonalitet. Vi har ikke lært Node.js i løpet av vår tilværelse på NTNU og er derfor enige som gruppe at vi skal være forberedt på kostnader rundt nettbaserte kurs som dekker for

eksempel Node.js og annen teknologi som kan bli nødvendige for å fullføre prosjektet. Vi vil også kjøpe pensumbøker dersom det trengs.

1.3.3 Tidsramme

Prosjektet ble introdusert i november 2017, med oppstart vårsemesteret 2018, 10. januar, og frem til innlevering av rapporten 16. mai, 2018.

1.3.4 Ressurser

Et langsiktig mål for prosjektet er å utvikle en fullstendig tilleggspakke som kan gjøres tilgjengelig for alle brukere av Atlassian Jira. Den vil da bli lagt tilgjengelig som en selvstendig webapplikasjon som aksesseres via kontrollpanel-fanen. Løsningen vil bli hostet på egne servere, men applikasjonen vil interagere med Jira sin rest-API.

1.3.5 Fremdriftsplan og prosjektorganisering

For å sikre god og kontinuerlig fremdrift har vi valgt å sette opp et utgangspunkt for oppgaver og gjøremål i et detaljert Gantt-skjema. Bruk av Gantt-skjema som metode presenteres nærmere i *kapittel 2*.

1.3.6 Gjenstående arbeid

I *kapittel 7* utdyper vi hva som gjenstår av arbeid, med forslag til hva som kan videreutvikles. Gjenstående arbeid inkluderer et grundig back-end, herunder innhenting av data via Jira sin rest-API og bruk av Node.js for å fasilitere dette. Etter tidlige samtaler med oppdragsgiver, setter vi dette som et videre mål for løsningen og en eventuell mulighet for prosjektet dersom tid tillater det. Flere funksjoner som er tiltenkt i front-end er ikke implementert. Disse vurderes til videre utvikling grunnet tidsprioritering.

I første omgang var det et mål å ha en fullstendig utvidelse til Jira. Det avklares raskt at dette kan være for tidkrevende, og vi vurderer dette til å være et langsiktig mål.

1.4 Risikoanalyse

Gruppen har valgt å foreta en risikoanalyse for å kartlegge mulige utfordringer og hendelser underveis i bachelorprosjektet. Årsaken til at dette gjøres, er for å være best mulig forberedt og sikre at dersom noe skulle skje, vil vi ha retningslinjer og nøye gjennomtenkte tiltak klare, som er laget uten at panikk påvirker evnen til å finne gode løsninger. Det bidrar også til økt oppmerksomhet på hva som kan gå galt, og hvordan vi som gruppe kontinuerlig bør jobbe for at dette ikke skal skje.

Tabell 1.1 – Risikodiagram

	Ubetydelig	Alvorlig	Svært alvorlig
Svært usannsynlig		5	7 + 8
Sannsynlig	6	1 + 4	
Svært sannsynlig		2	3

1.4.1 Potensielle risikomomenter og tiltak

1. Vanskeligheter med å få avtalte møter med oppdragsgiver
   TILTAK: Ha åpen dialog med oppdragsgiver, diskutere ulike tider og komme til enighet, inngå kompromiss om nødvendig; f.eks. ikke være fulltallig gruppe. Organisere møter etter oppdragsgiver sine ønsker, ikke motsatt.
2. Feildisponering av tid
   TILTAK: Re-evaluere og omprioritere oppgaver og fullføre de viktigste først.
3. Tidsfrister opprettholdes ikke
   TILTAK: Kom til bunns i problemet og diskutér ulike måter for å opprettholde neste frist.
4. Manglende kompetanse
   TILTAK: Søk kompetansen: Spør fagfolk, eller finn informasjon på Internett.
5. Kommunikasjonssvikt blant gruppemedlemmer
   TILTAK: Samle gruppen, diskutere problemet og deretter løse det via dialog.
6. Sykdom på enkeltmedlemmer
   TILTAK: De resterende gruppemedlemmene fordeler oppgavene seg imellom om mulig.
7. Tap av data
   TILTAK: Definere forhåndsregler for å forebygge denne risikoen; f.eks. ta i bruk mest mulig Cloud baserte tjenester fra anerkjente aktører.
   Let i arkivene og finn seneste versjon om tilgjengelig.
8. Et gruppemedlem slutter
   TILTAK: Finne kompromiss med oppdragsgiver rundt nedskalering av oppgaven slik at arbeidsmengde reflekterer antall gruppemedlemmer igjen.

1.4.2 Oppsummering av risikoanalyse

Våre største risikomomenter er knyttet til arbeidsmengde og tidsfrister. Det forutsetter at hvert gruppemedlem er realistiske med tidsberegninger og ikke tar på seg mer enn det en klarer å gjennomføre. I tillegg til tiltakene som er kartlagt i analysen, vil det lønne seg å beregne lenger tid enn en tenker er realistisk. På denne måten vil gruppen ha gjennomførbare tidsfrister og minimere risiko for at tidsfrister ikke overholdes, samt feildisponering av tid.

1.5 Arbeidsflyt og –miljø

For å sikre god arbeidsflyt innad i gruppen, samt ta høyde for at hvert enkelt gruppemedlem har andre emner i tillegg til bachelorprosjektet, har vi valgt å sette av mandager til å jobbe med disse. Resterende arbeidsdager vil derfor være satt av til å jobbe med bachelorprosjektet, samtidig må det påberegnes jobbing i helger dersom det skulle være nødvendig. For å skape et godt gruppemiljø er det ønsket at vi sitter sammen og jobber så godt det lar seg gjøre. Ettersom et av gruppemedlemmene ikke holder til i Gjøvik, er det avtalt at dette gruppemedlemmet er i Gjøvik torsdager og fredager, samt andre dager om det er arrangementer eller møter som hele gruppen skal delta på. Det å holde gruppen samlet kan bidra til bedre kommunikasjon, samt det at vi blir bedre kjent vil kunne bidra til økt trivsel og godt samarbeid gjennom perioden.

Det er fastsatt veiledningsmøter med veileder hver fredag, om ikke annet blir avtalt. Dette mener vi vil bidra til å sikre god flyt og fremgang, ettersom det er et tydelig ønske fra veileder å se progresjon og endringer fra møte til møte.

Møter med oppdragsgiver vil skje siste fredag i hver sprint. Som nevnt, etter gruppens og oppdragsgivers ønske, følges Scrum-metoden, for utvikling av løsningen. Dette mener vi også vil bidra til god arbeidsflyt ettersom det krever resultater hver tredje uke, samtidig vil gruppen få mer innsikt og forståelse i en arbeidsmetode som er mye brukt i utviklingsbransjen. Derfor, med veiledning fra oppdragsgiver, blir det gjennomført jevnlig sprintplanlegging og sprint-reviews. Sprintplanlegging vil i stor grad gjøres innad i gruppen, men vil også være tema på møter med oppdragsgiver.

For å sikre at læringsmål blir oppfylt, skal hvert gruppemedlem utfordre seg selv ved å få arbeidsoppgaver de ikke har kunnskap om eller erfaring med. Det er likevel viktig å ta hensyn til prosjektets tidsramme, og om nødvendig må arbeidsoppgaver fordeles i henhold til kunnskap og erfaring. Hvert gruppemedlem skal lære noe, men det skal ikke gå på bekostning av tidsramme og fremgang. Alle i gruppen skal til enhver tid ha oversikt over status på prosjektet og egne arbeidsoppgaver.

For et godt arbeids- og gruppemiljø er alle enige om viktigheten av å være ærlige og ydmyke – i forhold til hverandre, veileder og oppdragsgiver.

1.6 Mål

1.6.1 Hovedmål

Vi vil utarbeide en løsning for Escio som gir et informativt og oversiktlig innblikk i pågående prosjekter, samt viser den enkelte ansattes arbeidsmengde. Løsningen vil kunne gi forslag til prioriteringer og omrokkeringer, basert på den enkeltes ferdigheter og tilgjengelige arbeidstimer.

1.6.2 Effektmål

Ved gjennomføring av dette prosjektet vil vi gi prosjektledere et verktøy for å allokere sine ressurspersoner på en smartere måte. Med ressurspersoner, menes ansatte i et Software-utviklingsteam som prosjektledere har tilgjengelig til disposisjon. Et ønsket effektmål av løsningen vil være å forebygge risikoen for at ansatte blir utsatt for uforsvarlige arbeidsmengder. Til slutt er målet med en ferdig utviklet løsning å styrke Atlassian Jira ved å tilby ny og verdifull funksjonalitet til tjenesten.

1.6.3 Resultatmål

Målene vi ønsker å oppnå for løsningen ved gjennomføring av dette prosjektet, er å programmere en fremtidssikker løsning i Vue.js som kartlegger ressursbruk for prosjektledere. Et sekundært mål er å utvikle en løsning som bidrar til å effektivisere tidsbruk i et Scrum-orientert selskap.

1.6.4 Læringsmål

Etter endt prosjekt vil vi ha oppnådd kunnskaper om ny teknologi som Vue og Node. Målet er å bli bedre kjent med disse konkrete teknologiene, samt det å utfordre oss selv ved å prøve ut – for oss – ukjent teknologi. Et annet svært viktig læringsmål er prosjektgjennomføring og -ledelse av et større prosjekt, som vil gi oss nyttig erfaring videre i høyere studier og i arbeidslivet.

1.7 Øvrige roller

1.7.1 Presentasjon av oppdragsgiver

Oppdragsgiver er teknologiselskapet Escio, med 12 medarbeidere som holder til på Gjøvik og Hamar. De har kunder over hele landet, blant annet European Space Agency, Akershus fylkeskommune, Fagforbundet og Gausdal Landhandleri (Escio, 2018).

Håvard Narvesen og Terje Krogstad har vært gruppens bindeledd fra Escio. Begge er prosjektledere og seniorutviklere. Terje har vært gruppens kontaktperson fra bedriften, og har hatt en sentral rolle gjennomgående i hele prosjektet. Håvard har fungert som Terjes erstatter ved tilfeller der Terje selv ikke hadde mulighet til å møte. Både Terje og Håvard har bidratt som kunde, produkteier, testperson, og intervjuobjekt.

1.7.2. Presentasjon av veileder

Gruppens veileder under prosjektet har vært Eivind Arnstein Johansen. Han er universitetslektor ved Bachelor i mediedesign ved NTNU i Gjøvik, der han underviser i emner som *Ergonomi i digitale medier, responsiv web, prototyping for mobil og nettbrett* med flere (NTNU, 2018). Eivinds ekspertise innenfor design, universell utforming og prototyping

er av betydelig verdi for prosjektet og gruppens fremgang. Ukentlig veiledningsmøte har sørget for lærdom, gode diskusjoner og konstruktive tilbakemeldinger.

1.8 Målgruppe

Målgruppen er ansatte i Escio med rollen Scrum Master, da vår løsning vil bli et internt arbeidsverktøy med størst verdi for de som behøver et overordnet blikk på effektiviteten rundt prosjekter og fremgang. Dersom løsningen blir vellykket, kan denne legges ut på markedet som en plugin til Atlassian Jira. Dette vil gi oss en større sekundær målgruppe som dekker Software-team som utvikler etter smidige metoder.

1.9 Bakgrunn og kompetanse

Alle gruppemedlemmene studerer webutvikling ved NTNU i Gjøvik. Faglig har alle vært gjennom de samme obligatoriske emnene som leder opp mot bachelorprosjektet. Gjennom studiet har emner som omhandler koding, programmering, brukersentrering og webprosjekter bidratt til at vi har opparbeidet oss kompetanse til å gjennomføre dette prosjektet.

Gruppens medlemmer har et mangfoldig kunnskapsnivå. Interessen for webutvikling strekker seg også utenfor studiet, i form av prosjekter på hobby- og freelancebasis. Noen interesserer seg mer for programmering, andre det visuelle og noen er mer opptatt av brukeropplevelser. Helhetlig anser vi dette å være en god sammensetning av kompetanse som komplimenterer hverandre.

I løpet av prosjektperioden må gruppemedlemmene lære seg Node.js og Vue.js. En grundigere forståelse av Git og bruk av kommandolinjen er nødvendig for effektivt samarbeid. Noen gruppemedlemmer må også lære seg preprosesserings-språkene Sass, Pug og CoffeeScript. Disse teknologiene presenteres nærmere i *kapittel 2*.

1.10 Terminologi

Vi bruker flere ulike begreper gjennomgående i rapporten. Noen er avklart i teorikapitlet, andre blir forklart her.

Storypoints: En bestemt tallrekke for å estimere oppgavers relative størrelse i forhold til hverandre, i et Scrum-miljø.

Velocity: Antall storypoints et Scrum-team klarer å levere hver sprint.

Brukstesting: Gruppens egne vri på *brukertesting*. Vi mener brukertesting er uheldig og ikke samsvarer med realiteten, ettersom det er systemet, ikke brukeren som skal testes.

Språk eller språkene: Kode- og programmeringsspråk brukes for variasjon og lesbarhet.

Open source: Et engelsk lånebegrep – betyr en åpen kildekode til, for eksempel, et dataprogram som er tilgjengelig for alle.

Rammeverk: Begrepet brukes i overført betydning, og benyttes i ulike kontekster. Blant annet for PACT-analyse og Scrum, men mest innen webteknologi. Der defineres rammeverk som et verktøy som legger til rette for implementering av visse teknologier og konsepter.

Abstraksjonslag: I datavitenskap regnes et abstraksjonslag som et forsøk på å gjemme implementeringsdetaljene for et gitt sett av funksjonalitet for å forenkle samkjøring av ulike systemer.

Forking: Et kjent konsept innen programvareutvikling. Begrepet omhandler kopiering av kildekoden tilhørende et prosjekt for å starte ny selvstendig utvikling på det originale prosjektet.

NPM: Et akronym for *Node Package Manager*, og er primært et register av open source moduler.

1.11 Rapportstruktur

Rapporten er delt inn i åtte ulike kapitler. I neste kapittel beskriver vi alle metoder og verktøy, i tillegg drøfter vi teorien som brukes. Deretter analyserer vi brukerintervju, den eksisterende løsningen, utarbeider en konkurrentanalyse og til slutt tas et valg angående webteknologier. I *kapittel 4* diskuterer vi prosessen med prototype fra start til slutt, samt gjennomgår de ulike brukstestene vi utførte. Neste kapittel tar for seg implementeringsprosessen, før vi går over til en diskusjon av vår endelige løsning. I *kapittel 7* legger vi frem forslag til videreutvikling av løsningen, og til slutt drøftes resultatet og vi konkluderer i *kapittel 8*.

2. Teori, metoder og verktøy

2.1 Introduksjon

I dette kapitlet presenterer vi teori, metoder og verktøy som benyttes i prosjektet. Disse er relevante for alle steg i prosessen, samt danner grunnlaget for valgene vi gjør. Aktuell teori blir beskrevet, som et dypdykk i Scrum-metoden, brukersentrert design, og ny webteknologi som legger grunnlaget for å utvikle en moderne web-applikasjon.

2.1.1 Utvalgt litteratur

Under prosjektarbeidet benytter vi oss av pensumbøker fra tidligere emner, da vi anser disse som troverdige kilder. I tillegg tar vi i bruk annen litteratur relatert til prosjektet, herunder flere bøker og fagfellevurderte artikler for å se ulike synspunkter.

Når det kommer til webteknologi, eksisterer det et mangfold av teknologier til samme formål omfavnet av et landskap som er i konstant endring. Webutvikling som fagfelt er i konstant utvikling, med stadig nye teknologier som etableres der det per skrivende dato finnes minimalt med faglitteratur. Derfor benytter vi oss av artikler som ikke er fagfellevurderte fra anerkjente frontfigurer. Disse artiklene hentes fra publiseringsplattformen Medium (www.medium.com), i tillegg til teknologienes egne nettsider. På grunn av store kvantiteter av blant annet språk og rammeverk som fyller samme formål, tok vi utgangspunkt i ulike undersøkelser i interessen av å begrense antallet kandidater vi tok til vurdering. Dette for å spisse fokus og unngå arbitrære valg.

2.2 Prosjektorganisering og samhandlingsverktøy

2.2.1 Gantt-diagram – en overordnet plan

Gantt-diagrammer brukes for å planlegge et prosjekt. Det beskriver detaljert hvilke oppgaver og aktiviteter som skal utføres, med bestemte start- og sluttdatoer. Diagrammet gir en tydelig oversikt over prosjektets tidsramme, nødvendige oppgaver og om noen av disse overlapper med hverandre. Gantt-diagrammer gir en visuell fremstilling av prosjektplanen. I oppstartsfasen av et prosjekt kreves ofte slik planlegging at en går gjennom alt som skal til for at prosjektet skal fullføres og en eventuell løsning kan publiseres. Samtidig er et Gantt-diagram velegnet til å dokumentere og følge hver oppgaves fremgang (Rosenberg, Stephens og Collins-Cope, 2005). I utviklingsprosjekter vil ikke Gantt være tilstrekkelig for planlegging, ettersom det i stor grad setter urealistiske og strenge tidsrammer, og tar ikke høyde for endringer og uforutsette hendelser (Rosenberg, Stephens og Collins-Cope, 2005). Likevel kan et Gantt-diagram anses å

fungere som en overordnet plan og gi et godt utgangspunkt for videre planleggingsmetoder.

2.2.2 Atlassian Jira

Verktøyet brukes for å fasilitere et Scrum-basert prosjektarbeid. Jira er et viktig verktøy for gruppen, ikke bare fordi det kan bidra til å opprettholde en god arbeidsflyt, men også fordi oppgavebeskrivelsen omhandler utformingen av en tilleggspakke til Jira. Det anses derfor å være av stor verdi for gruppen å bli kjent med verktøyet sitt grafiske brukergrensesnitt, samt styrker og svakheter ved tjenesten.

2.2.3 Git & Github

For den tekniske utarbeidingen av løsningen, en web-applikasjon, behøves det verktøy som fasiliteter effektivt og organisert samarbeid i en kodebase.

Git er et *open source* versjonskontroll-system som gir mulighet for kontinuerlig lagring av en kodebase under utviklingsprosessen. Lagringen foregår i form av *øyeblikksbilder*, der hver iterasjon av prosjektet blir lagret og loggført. Slik har en mulighet til å gå tilbake til et tidligere stadie i prosjektet ved behov (Pham og Pham, 2012, s. 189).

Git yter også godt for det kollaborative aspektet ved programvareutvikling. Med Git kan kode lagres sentralt ett sted, slik at flere utviklere kan arbeide med koden i parallell, etterfulgt av å *dytte* sine endringer tilbake til den sentrale kodebasen (Pham og Pham, 2012, s. 189).

For å benytte seg av Git sine kollaborative egenskaper, må kildekoden bli lagret på en server som alle kollaboratører kan hente fra. GitHub er en tjeneste for nøyaktig dette formålet. GitHub tilbyr gratis offentlig lagring av kataloger som administreres via Git. På GitHub loggføres også historikk og statistikk for hver bidragsyter.

2.2.4 Google Drive

Prosjektarbeidet i sin helhet vil kreve administrering av et mangfold av dokumenter og filer. Eksempler er refleksjonsnotater, arbeids- og møtelogger, samt analyser og iterasjoner på prototyper. Google Drive er en nettbasert tjeneste for lagring av dokumenter i skyen. Drive kan bidra til å styrke vårt samarbeid ettersom flere har tilgang til dokumenter, og uten risiko for å tape data da alt er lagret på en server, ikke lokalt. Drive inkorporerer også Google Docs, en tjeneste som tilbyr muligheten for kollaborativ redigering av dokumenter, som til gjengjeld gjør det enklere for alle å bidra.

2.3 Analyseverktøy

2.3.1 Brukerintervju

Intervjuer er godt egnet i startfasen av et prosjekt. En kan avdekke forventninger og ønsker, eksisterende praksis, rutiner og hensikter med det som skal utvikles. Effekten av et intervju kan styrkes, blant annet ved å stille åpne spørsmål. Et godt intervju er planlagt, og intervjueren er bevisst på hva som sies til intervjuobjektet (Sandnes, 2011, s. 240-241).

Relevant informasjon en er interessert i å få fra brukere, er blant annet hvordan, hvor og når løsningen skal brukes. Man kan bør få informasjon om hvordan dagens situasjon er, og hvilke problemer som eksisterer med dagens løsning. I tillegg er det aktuelt å spørre om den mentale modellen, altså hvordan brukerne tenker at løsningen skal fungere (Cooper *et al.*, 2014, s. 43).

Fokusert intervju

Et fokusert intervju har en avgrensning, og kan fokusere på innhenting av ideer der hensikten er å avdekke hva og hvordan brukere tenker om noe. Et slikt intervju avdekker også eventuelle mentale modeller brukeren har. En kan også fokusere på kartlegging av dagens situasjon, blant annet ved å spørre om fremgangsmåter, rammer og unntak (Sandnes, 2011, s. 240).

2.3.2 PACT-analyse

PACT-analyse er svært sentralt i boken "Designing Interactive Systems" (Benyon, 2010) og er navnet på et rammeverk utarbeidet for utvikling av interaktive systemer. Denne analysen er et verktøy for å kartlegge og forstå variasjonen i de ulike elementene som PACT er bygget opp av, samt hvordan de står sammen. De fire elementene er mennesker (People), aktiviteter (Activities), kontekster (Contexts) og teknologier (Technologies). Hver for seg beskriver de aktuell og nødvendig informasjon i forbindelse med bruken av en bestemt løsning eller for å kunne oppnå et bestemt mål. Denne delen tar utgangspunkt i hvordan Benyon (2010) presenterer PACT-analyse som metode:

Når en snakker om elementet *mennesker* er formålet å forstå hvordan aktuelle brukere er forskjellige, samt være oppmerksom på at disse menneskene kan ha ulike funksjonsevner og -nedsettelser. Dette kan være med på å påvirke hvor tilgjengelig og brukervennlig løsningen eller teknologien er. Mennesker kan være nødt til å benytte seg av teknologi for å dekke bestemte behov.

Dette skjer ved hjelp av interaksjon med teknologien, og elementet *aktivitet* er knyttet til alle nærliggende aktiviteter som direkte eller indirekte bidrar til at behovet dekkes. Aktivitetene kan ha ulik grad av kompleksitet, og ved analysering kan det være nyttig å kartlegge karakteristikken av aktiviteten ved hjelp av motsetninger. Om aktiviteten er enkel eller komplisert, om den kan gjennomføres med få eller mange steg, sannsynlighet for hyppig utførelse, eller om det skjer det kun én gang på ett bestemt tidspunkt av dagen.

Dette er viktig å kartlegge for å kunne ta hensyn til ulike faktorer som kan påvirke bruken av et system.

Kontekster er situasjoner der behovet oppstår eller ved anvendelse av løsningen. Dette kan bidra til påvirkning i hvordan løsningen skal brukes. Kontekster kan være fysiske, sosiale og organisatoriske. Det antas å kartlegge hva som kreves av løsningen, teknologien og brukerne. I gitte situasjoner vil denne analysen kunne sette fokus på viktige aspekter en må ta hensyn til i utviklingen, og kan blant annet være i form av sikkerhet og personvern. Viktige samfunnsaktuelle temaer kan være avgjørende for realiseringen av løsningen, og hvorvidt den er ønskelig å bruke.

Systemer avhenger i de fleste tilfeller av *teknologi*, og det er hvilken teknologi som benyttes som kartlegges i PACT-analysens siste element. I forbindelse med teknologi fokuseres det på å identifisere hva som må inn og ut av data, hvordan dette skal vises, samt hvilke deler og funksjoner som må kommunisere med hverandre for at løsningen skal fungere – og ikke minst ha verdi.

PACT-analyse bygger ofte på informasjon hentet fra intervju som er gjort i tidlig fase. Disse intervjuene søker å få innblikk i brukernes behov. Formålet med analysen er å skape en oversikt over, samt gå mer i dybden på, hva som kommer frem i intervjuet. Det hjelper designere og utviklere å forstå hva brukerne egentlig har behov for. Det er et verktøy for å kategorisere informasjonen og sette den i et forståelig og oversiktlig system. PACT-analyse er et godt utgangspunkt for videre metoder i en designprosess, ettersom informasjonen i en slik analyse kan sies å være grunnlaget personas og scenarier er utviklet etter. Dette fordi personas og scenarier består blant annet av detaljerte beskrivelser om personer som bruker en løsning og situasjoner rundt bruken (Benyon, 2010).

2.3.3 Konkurrentanalyse

Markedet er svært konkurransepreget, og det er derfor utslagsgivende å ha kunnskap om konkurrentene. For å få en introduksjon til markedsføringens verden, benytter gruppen seg av boken til Philip Kotler: *Markedsføringsledelse* (2005). Denne boken blir sett på som en ledende bok innen markedsføring, og Kotler selv som «verdens fremste autoritet innen markedsføring» (Gyldendal, 2018).

I boken poengterer Kotler (2005, s. 197) at «fremgangsrike bedrifter utarbeider og bruker sine egne systemer for [...] å samle informasjon om konkurrentene sine». Det kan være enkelt å finne de mest nærliggende konkurrentene: de som dekker de samme behovene og tiltrekker de samme kundene, gjerne kalt de *aktuelle* konkurrentene som finnes i samme bransje. Det er dermed ofte de *potensielle* konkurrentene som er mer faretruende – morgendagens bedrifter som kan komme med ny og bedre teknologi (Kotler, 2005, s. 199).

En populær konkurrentanalyse som blir brukt av alt fra nybegynnere til fagkyndige er SWOT-analysen. Kotler (2005, s. 75) forklarer en slik analyse som «en generell vurdering av en bedrifts sterke og svake sider, muligheter og trusler». SWOT er et akronym fra de engelske ordene *strengths, weaknesses, opportunities* og *threats*. Formålet med en SWOT-

analyse er å avdekke nye markedsmuligheter, hvor et sentralt spørsmål er å finne ut om bedriften skal fokusere på å bedre sine sterke sider, eller sette i gang prosesser for å minimere trusler og svakheter (Kotler, 2005, s. 77).

2.4 Prosjektstyringsverktøy – Smidige metoder

2.4.1 Agile – en kort introduksjon

«*Agile* er en måte å tenke på» forklarer Jongerius (2012, s. 19). Smidige metoder oppstod rundt sytti-tallet, men det ble først populært på nitti-tallet under navnet *lightweight methods*, skriver han videre. Grunnlaget ble satt i 2001, da *Agile Manifesto* ble skapt (Pham og Pham, 2012, s. 3). Dette manifestet består av fire ulike verdier som separeres i to deler. Venstre side vektlegges foran den høyre siden, og verdiene er som følger:

> Personer og samspill fremfor prosesser og verktøy
> Programvare som virker fremfor omfattende dokumentasjon
> Samarbeid med kunden fremfor kontraktsforhandlinger
> Å reagere på endringer fremfor å følge en plan (Beck *et al.*, 2001).

Smidige metoder kjennetegnes ved kontinuerlig forbedring, med jevnlig tilbakemelding fra kunden (Beck *et al.*, 2001). «Ved bruk av smidige metoder er endring viktigere enn å følge en streng plan», hevder Jongerius (2012, s. 19). Det er også viktig å poengtere at «smidige metoder ikke er en egen metodikk i seg selv, men heller et paraplybegrep som beskriver flere iterative og inkrementelle metoder innen programvareutvikling», slik Kayes, Sarker og Chakareski (2016) beskriver i sin studie. De mest kjente smidige metodene er blant annet Extreme Programming (XP), Lean, Crystal og Scrum.

2.4.2 Scrum

Scrum er en smidig metode, ofte sett på som en av de enkleste metodene å forstå, men en av de vanskeligste å implementere godt (Lacey, 2012, s. 1). Ken Schwaber og Jeff Sutherland er noen av personene som blir sett på som grunnleggerne av Scrum, da de utformet metoden allerede i 1990 (Jongerius, 2012, s. 20). Scrum har mange likhetstrekk fra smidige metoder, samtidig som det har blitt innarbeidet nye verdier og metoder gjennom årene. Videre i dette underkapitlet peker vi på de mest brukte og utbredte rollene, hvordan Scrum organiseres, samt oppgavebehandling i team.

Grunntanken bak Scrum er å ha en overordnet visjon, kontra å utarbeide konkrete mål og ha en arbeidsplan som er fastsatt for fremtiden. Kundens ønsker, arbeidsressurser og miljøet rundt produktet kan stadig endres. Ved å holde fokus på brukernes faktiske behov og være åpne for endringer, kan gi best mulig sluttresultat (Jongerius, 2012, s. 20).

Pham og Pham (2012, s. 13) poengterer at Scrum er et rammeverk for programvareutvikling og ledelse, med et sterkt fokus på at medlemmene i et Scrum-team er selvorganiserte og føler eierskap og stolthet i hva de produserer. Videre skriver

forfatterne at Scrum reduserer risiko ved at en opparbeider en mer adaptiv prosjektstyringsprosess. Lacey (2012) argumenterer motstridende at Scrum er et verktøy, dermed ikke en metode. Deretter forteller Lacey at Scrum fokuserer på effektivitet, iterasjon og samarbeid. Felles for alle tilhengere og brukere av Scrum, er fokuset på å levere et *Minimum Viable Product (MVP)* etter hver iterasjon. Dette vil si å produsere separate komponenter klare til levering, i blokk-iterasjoner over flere uker (Jongerius, 2012, s. 20).

Sprinter

En sprint er en forhåndsdefinert periode, der gjennomsnittlig lengde er to til fire uker (Kayes, Sarker og Chakareski, 2016). Sprinter er sykluser, eller iterasjoner, der Scrum-teamet velger ut noen oppgaver og brukerhistorier til et potensielt overførbart produkt som kan leveres til kunden. Disse velges ut på et såkalt *sprint planning meeting*. Under første del av dette møtet, kommer kunden eller produkteier med ønsker om hva de vil skal utarbeides under sprinten. Deretter velger Scrum-teamet ut de enkelte oppgavene som de vet de kan klare å levere i løpet av den påfølgende sprinten (Pham og Pham, 2012, s. 272).

Underveis i sprinten skal Scrum-teamet ha *daily standups,* som er daglige møter på morgenen hvor hele teamet samles for å oppdatere hverandre. Møtet bør ikke ta mer enn femten minutter, og skal inneholde hva medlemmene har gjort siden sist, hva de skal jobbe med den dagen og til slutt om de har problemer eller konflikter de ønsker å ta opp i fellesskap (Jongerius, 2012, s. 95).

Et møte for sprint retrospektiv er et relativt kort møte der Scrum-teamet går felles igjennom hva som fungerte godt og mindre godt ved slutten av en sprint. De ser på positive og negative sider – både hvordan samarbeidet var og hvilke oppgaver de fullførte, videre går de inn på hva som eventuelt gjorde at de ikke klarte å fullføre alle oppgavene tildelt sprinten. Dette møtet er for at Scrum-teamet skal ta lærdom av, reflektere og forbedre seg til neste sprint (Pham og Pham, 2012, s. 272).

Etter fullført sprint samles Scrum-teamet igjen for å ha *sprint review*. Her vises det en demonstrasjon til kunden, og eventuelt produktets interessenter av hva som har blitt gjort under sprinten (Pham og Pham, 2012, s. 272).

Oppgavebehandling i Scrum

Oppgavene i en sprint blir satt til ulike statuser, basert på hvor langt et medlem har kommet på den. De tre vanligste statusene er *to-do, in-progress* og *done*. *To-do* er en oppgave som ingen har begynt på, så her kan en prosjektleder allokere ansvar for oppgaven. *In-progress* derimot, er en oppgave et medlem er i gang med. *Done* er derfor en fullført oppgave. «Definisjonen av *ferdig* varierer ofte avhengig av prosjektsituasjonen» skriver Pham og Pham (2012, s. 115) i boken *Scrum in Action*. Forfatterne foretrekker at *done* vil si noe som kan leveres til produksjon, testes og deretter evalueres – enten av brukere eller kunde. Poenget er likevel at alle rollene involvert i Scrum-teamet har kommet til enighet om hva de anser som *ferdig* og leveringsklart etter hver sprint.

Storypoints og hvordan benytte det

Scrum-team benytter seg ikke av timeantall når de fører logg. I Scrum måles oppgavens størrelse etter *storypoints*. Disse tallene er estimater som er relative til hverandre, og baserer seg løst på Fibonacci-sekvensen. Hvert tall i denne sekvensen er summen av de to foregående tallene – 1, 2, 3, 5, 8, 13, 21 og videre. Det er populært å bruke tallene 20, 40, 80 og 100 som de neste etter 13, derav brukes kun et utdrag av Fibonacci-sekvensen.

Effektiviteten til Scrum-team øker over tid, ettersom de blir bedre kjent med hverandre og arbeidsrutinene. Likevel kan uforutsette hendelser oppstå, og derfor er det lurt å ha estimater fra tidligere sprint i bakhånd, for å justere storypoints høyere eller lavere i kommende sprint, slik kan teamet sikre fullføring av alle oppgaver i backlog (Jongerius, 2012, s. 94).

Velocity, eller laghastighet, kan bli en hindring for gjennomføring av sprinter, ettersom lag-hastighet ikke er sammenlignbar mellom ulike Scrum-team (Pham og Pham, 2012, s. 178).

Poker planning er en estimat-teknikk som brukes av Scrum-team for å finne den relative størrelsen på de ulike oppgavene satt i en backlog (Pham og Pham, 2012, s. 61). Det finnes mange ulike måter å utføre poker planning på, men en vanlig metode er å benytte seg av en applikasjon. I en slik applikasjon velger hvert medlem velger ut et storypoint-tall som de mener passer til oppgaven, og deretter viser alle frem sitt tall samtidig til gruppen. Herfra kan en diskutere sine meninger, spesielt dersom noen mener oppgaven er verdt én storypoint, mot en annen som oppfatter den som 13 storypoints.

Et *Burndown*-diagram viser til den daglige utviklingen av den pågående sprinten. Her er hovedessensen å få oversikt over hvor mye arbeid (eller storypoints) som er ferdige, mot hvilke oppgaver som gjenstår (Kayes, Sarker og Chakareski, 2016).

Testing i Scrum

Testing spiller en sentral rolle i Scrum. Hvorvidt et Scrum-team har mulighet til å levere noe etter hver sprint, vil ofte ha sammenheng med hvor godt utført testingen er. Blir tester organisert og gjennomført riktig, har gjerne teamet mer å levere til oppdragsgiver (Pham og Pham, 2012). Kayes, Sarker og Chakareski (2016) presenterer i sin artikkel en oversikt over noe de har valgt å kalle *Product Backlog Rating (PBR)*, som gir innsikt i testprosessen og dens kvalitetsgrad i Scrum når en produserer og itererer et produkt.

Scrum-teamet

Et Scrum-team består av *Scrum Master*, produkteier og øvrige teammedlemmer. Alle medlemmer i et Scrum-team har en viktig rolle, der teamet samlet skal inneha den nødvendige kompetansen for å utvikle en spesifikk løsning. Kompetanse kan for eksempel være nødvendige kode- og programmeringsferdigheter, eller kunnskap om design og testing. Selv om et Scrum-team innehar god kombinasjon av kunnskap, er det likevel ikke gitt at de følger Scrum metoden til punkt og prikke, enten det gjelder hierarki eller organisering av prosjekt (Pham og Pham, 2012).

Produkteier er, som navnet tilsier, ansvarlig for å opprettholde produktet etter kundes, samt andre interessenters ønsker til prosjektet. Produkteier har også en sentral rolle i å ivareta produktets mål og visjon (Pham og Pham, 2012). Ved oppstart av nytt prosjekt starter produkteier med å utforme en liste over krav som skal gjennomføres på bakgrunn av kundens målsetning. En slik liste kalles en *product backlog*. Pham og Pham (2012, s. 7) skriver i boken *Scrum in Action* at «[...] enkelt sagt, så er product backlog en prioritert liste over krav, som kan inkludere alt fra forretningsfunksjoner til teknologier til tekniske problemer til feilrettinger».

En annen sentral rolle for et Scrum-team er Scrum Master, en prosjektleder som skal styre et team i riktig retning for å forstå og anvende Scrum i prosessen. Har en denne rollen, er det viktig å være klar over at en bør ha evnen til å løse konflikter, både innad i teamet, men også dersom det oppstår utenforstående konflikter med for eksempel kunde. Scrum Master bør også kunne fjerne eventuelle hindringer, samt legge til rette for at teamet kan bli et effektivt og høyt ytende team (Pham og Pham, 2012).

I et av eksemplene beskrevet i *The Scrum Field Guide* (Lacey, 2012), er en av fallgruvene å tolke rollen som Scrum Master på lik linje med blant annet avdelingsleder eller seksjonssjef. Lacey (2012) presenterer en historie som illustrerer hvordan bedrifter implementerer Scrum uten god nok kjennskap og kunnskap til de ulike rollene, kan tolke disse hierarkisk.

Noll *et al.* (2017) utførte en studie om rollen til Scrum Master. Her stiller de spørsmålet om hvilken Scrum-rolle en prosjektleder skal få, dersom en bedrift går fra tradisjonell plandrevet utvikling til en smidig metode som Scrum. I studien peker Noll *et al.* (2017) på at en Scrum Master ofte har flere oppgaver enn opprinnelig utnevnt, særlig oppgaver som en typisk prosjektleder har. Dette mener Noll *et al.* (2017) gir en negativ virkning på Scrum-team, som kan forebygges dersom en prosjektleder heller går inn i rollen som produkteier.

I motsetning til mer tradisjonelle prosjektorganiseringer er ledelsesansvaret delt mellom de tre sentrale rollene i et Scrum-team: Scrum Master, produkteier og medlemmer. Grunnet det mer splittede ansvaret, kommer det tydeligere frem at alle på et team er avhengig av at det jobbes som et lag og at hver individuelle person gjør sin jobb (Pham og Pham, 2012).

2.5 Designmetoder

2.5.1 Brukersentrert design

Brukersentrert design går ut på at det er brukeren, og ikke teknologien, som er i fokus ved utvikling av en løsning. Utviklingen skjer i iterative sykluser, og hensikten er å få bedre forståelse for brukerne og deres behov (Sandnes, 2011, s. 15).

Universell utforming

Definisjonen på universell utforming fremmes av The Center for Universal Design NC State University, som hevder at universell utforming betyr at «utformingen av produkter og miljøer skal kunne brukes av alle mennesker, i størst mulig grad uten behov for tilpasning eller spesialdesigning» (NC State University, 1997). Det er også syv prinsipper som er knyttet til dette; *enkel og intuitiv i bruk, forståelig informasjon, toleranse for feil, like muligheter for alle, fleksibel i bruk, lav fysisk anstrengelse og størrelse* og *plass for tilgang og bruk*. Dette presenteres nærmere under punktet om *designprinsipper*.

I Norge i dag finnes det en lov og en forskrift som omhandler universell utforming. *Diskriminerings- og tilgjengelighetslovens §1* (Lovdata.no, 2008) forklarer lovens formål, som er å sikre at alle har like muligheter og rettigheter, uavhengig av funksjonsevne, til å delta i samfunnet – samt hindre diskriminering.

I samme lov *§11. Plikt til universell utforming av informasjons- og kommunikasjonsteknologi (IKT)* (Lovdata.no, 2008) fastsettes krav om at alle nye IKT-løsninger som utvikles etter 1. juli, 2011 skal være universelt utformet. Dette er mer spesifisert i *Forskrift om universell utforming av informasjons- og kommunikasjonsteknologiske (IKT)-løsninger* (Lovdata.no, 2017) der *§4. Krav til utforming av IKT-løsninger* viser til minstekravene som følger *Web Content Accessibility Guidelines 2.0* (W3C, 2008) sine standarder. Blant punkter som kan fremheves – fordi de direkte er knyttet til kodestandard og det visuelle – er:

«1.3.1 Informasjon og relasjoner (Nivå A): Ting skal være kodet slik det ser ut som»

Dette punktet har som formål å sikre at visuell eller auditiv informasjon er formatert på slik måte at det ikke endres dersom presentasjonsformatet endres. Det vil si dersom en bruker benytter seg av en skjermleser som leser av innholdet, er innholdet uendret og oppfattes på lik måte som en bruker som benytter seg av vanlig nettleser. Der brukere med normalt syn forstår struktur og sammenheng ved hjelp av visuelle virkemidler som størrelse, farger og avstander, vil en skjermleser som leser opp innholdet forstå dette ved hjelp av struktur- og seksjonstagger i HTML-koden. Når elementer kodes slik det ser ut, sikrer det at informasjonen er forståelig og tilgjengelig for alle.

«1.4.1 Bruk av farge (Nivå A): Ikke bruk presentasjon som bygger utelukkende på farge.»

For brukere som er fargeblinde eller av andre årsaker ikke oppfatter farger og fargeforskjeller, må informasjon som formidles gjennom fargeforskjeller også kunne formidles på andre visuelle måter. Dette kan for eksempel være bruk av SVG-bilder, der all informasjonen om bildets oppbygging er lagret i en fil. Alternativ tekst kan legges på bilder for å gi beskrivelse av bildets innhold.

«1.4.3 Kontrast (minimum, Nivå AA): Kontrastforholdet mellom teksten og bakgrunnen må være minst 4,5:1.»

Fargekontraster er et svært viktig punkt, og sikrer brukere tilgang til informasjon uten å benytte seg av andre tekniske hjelpemidler. Dette kriteriet bygger på et minimumskrav til kontrast mellom bakgrunnsfarge og fargen på teksten over. Her gjelder unntaket om

tekst som er ment som dekorativt element, og som på ingen måte formidler informasjon. Det skilles mellom stor og liten tekststørrelse, der mindre tekststørrelse krever større kontrast. For å sikre riktig kontrastforhold, eksisterer det flere verktøy tilgjengelig både på nett og for desktop-nedlastning.

«2.4.2 Sidetitler (Nivå A): Bruk nyttige og tydelige sidetitler.»

Når en bruker nyttige og tydelige sidetitler, sørger dette for at brukeren unngår å lete seg frem til ønsket informasjon. Det hjelper brukeren til å orientere seg rundt på siden, samt identifisere hvor på nettsiden han eller hun befinner seg. På denne måten kan en bruker raskt og enkelt finne relevant informasjon, skille innhold fra hverandre, samtidig som det gjør innholdet oversiktlig og strukturert.

Ikoner og symboler

Visuelle symboler, ved siden av tekst, hjelper brukere å gjenkjenne dem uten å måtte lese slik at de kan identifiseres raskere (Cooper *et al.,* 2014, s. 451). Sandnes (2011) poengterer at ikoner ikke bør brukes til pynt, da det kan skape støy i brukergrensesnittet. En annen utfordring er at betydningen av ikoner og symboler ofte er kulturelt betinget, samt at grafiske symboler generelt krever mer lagringsplass enn tekst (Sandnes, 2011, s. 94-99).

2.5.2 Designprinsipper

Benyon (2010) lister opp tolv designprinsipper han mener er aktuelle og viktige å følge ved utvikling av interaktive systemer. Disse prinsippene har han samlet fra flere kjente personer innen design og utvikling. Designprinsippene tar utgangspunkt i brukersentrert design og skal sikre en god løsning, der en bruker får dekket sitt behov og oppnådd sine mål på en tilfredsstillende og uanstrengt måte. En kan tolke og presentere designprinsipper på ulike måter, og det eksisterer flere bøker og artikler rundt temaet. En anerkjent person innenfor dette temaet er Donald Norman og hans bok *The Design of Everyday Things* (2013). I denne boken går han i dybden på flere prinsipper som gjør hverdagslige ting, systemer og teknologier lett å forstå og bruke, og hvorfor. Samtidig presenterer han gode eksempler på hva som gjør at noe ikke fungerer. På denne måten kan det være enklere å forstå designprinsippene og hvordan en bør bruke de riktig.

Designprinsipper skal sørge for at systemer er tilgjengelig og forståelige. I følge Norman (2013) er *tilgjengelighet* og *forståelse* å være de mest elementære kjennetegnene ved godt design. Med tilgjengelighet menes god synlighet for brukeren hvilke handlinger som er mulig å gjennomføre, hvor de er plassert og hvordan handlingene kan utføres. Forståelse er viktig for å kunne bruke en løsning. For å vite hva en funksjon gjør, er det viktig å forstå at det faktisk er en funksjon og det bør være tydelig hvilke handlinger som utføres og hva resultatet av disse handlingene vil være (Norman, 2013). Hvilke handlinger skjer i systemet og hvilke konsekvenser dette vil ha for omverdenen – ikke kun i løsningen (Benyon, 2010).

Tilbakemeldinger er essensielt for følelsen av kontroll. En løsning bør kommunisere at den reagerer på en handling, samtidig gi tilbakemeldinger dersom det oppstår feil og gi

mulighet til å gå tilbake eller angre. Dette bør indikeres umiddelbart. Med en gang en tilbakemelding er forsinket, kan det oppfattes som at noe ikke fungerer som det skal (Benyon, 2010). I all hovedsak skal designprinsipper legge til rette for brukeren ved å sikre sømløs og effektiv bruk der sluttresultatet vil være å oppnå et bestemt mål eller å dekke et behov. Brukeren skal forstå løsningen, føle at han eller hun har kontroll og til enhver tid vite hva som skjer.

Don Norman avslutter *The Design of Everyday Things* (2013, s. 298) med å si «teknologien vil endres, men de grunnleggende prinsippene for interaksjon forblir de samme». Samtidig presenterer Benyon (2010) en teori der menneskers aktiviteter danner grunnlaget for teknologiske krav, og aktiviteten endrer seg i takt med de teknologiske endringene. Mennesker tilpasser seg ny teknologi. Dette fører igjen til nye krav til teknologi, som deretter endrer seg for å tilfredsstille disse kravene. Dette er en syklus som gjentar seg. Det kan tolkes som at Norman (2013) og Benyon (2010) er uenige. For selv om teknologien endrer seg, vil de psykologiske prinsippene knyttet til menneskers samhandling med objekter og systemer forbli de samme. Men likevel vil de til en viss grad kunne endres. Dette fordi nye standarder blir satt og mennesker lærer seg og blir mer kjent med teknologien. De får et dypere grunnlag for forståelse og aksept. Likevel må ny teknologi ta utgangspunkt i Normans (2013) prinsipper, fordi det kan bidra til at nyere teknologi vil være enkel og mer forståelig å bruke.

2.5.3 Gestaltlovene

Gestaltlovene – lovene om form – analyserer hvordan mennesker tolker helhetlige visuelle inntrykk. Disse lovene er spesielt nyttige for design og analyse av brukergrensesnitt (Sandnes, 2011, s. 64-65).

Forgrunn og bakgrunn

Geometriske former deles inn i forgrunn og bakgrunn, der forgrunnen ligger foran bakgrunnen. Det at forgrunn og bakgrunn kan være tvetydig, er en effekt som ofte blir brukt innenfor grafisk design, men er en uønsket effekt i brukergrensesnitt (Sandnes, 2011, s. 65-67).

Nærhet

Loven om nærhet sier at vi grupperer elementer som er nær hverandre. Nærhet og distanse brukes til å organisere informasjon. Elementer som ligger nært hverandre, tolkes å høre til samme gruppe (Sandnes, 2011, s. 67-68).

Likhet

Elementer med samme form, størrelse, farge eller tekstur blir gruppert sammen. Denne loven brukes gjerne sammen med loven om nærhet for å uttrykke at elementer tilhører samme gruppe (Sandnes, 2011, s. 68-70).

Sammenkoblinger

Elementer som visuelt kobles sammen vil bli gruppert. Sammenkoblinger fremhever at spesifikke elementer hører sammen. Loven om sammenkoblinger kan overstyre lovene om nærhet og likhet, da elementer som er langt unna hverandre likevel vil oppfattes som en gruppe om de er sammenkoblet (Sandnes, 2011, s. 70-72).

Symmetri

Symmetriloven sier at symmetriske elementer grupperes. Denne loven fungerer selv om det er mangel på nærhet og likhet, ved at for eksempel former og posisjoner er symmetriske. Symmetriloven brukes mindre enn nærhetsloven og likhetsloven, men det er likevel greit å være klar over dette (Sandnes, 2011, s. 72-73).

Kontinuitet

Loven om kontinuitet kan bidra til å fremheve et oppsett i en grafisk fremstilling. Elementer som ikke er direkte sammenkoblet, men som følger usynlige linjer, oppfattes å tilhøre samme gruppe. Ved å organisere elementer langs usynlige linjer, vil man oppfatte linjer selv om de ikke eksisterer (Sandnes, 2011, s. 73-75).

Lukkethet

Loven om lukkethet sier at vi fyller inn manglende informasjon, og kan oppfatte former og mønstre som ikke egentlig er der. Dette utnyttes ofte i logoer, men sjeldnere i utformingen av brukergrensesnitt (Sandnes, 2011, s. 76).

2.5.4 Mentale modeller

En mental modell er en form for forklaring på en persons tankeprosess rundt hvordan noe fungerer. Det blir lagret i hukommelsen som generelle observasjoner, altså er de knyttet til personlige erfaringer. Siden mentale modeller baseres på observasjoner og antakelser, er de derfor ofte unøyaktige og ufullstendige. De danner et grunnlag for forventninger, og kan brukes til å forutsi hva som vil skje i nye situasjoner. Både utviklere og brukere av en løsning utvikler mentale modeller. Problemer oppstår når disse modellene ikke samsvarer. Derfor er utviklerens rolle å få innsikt i brukerens mentale modell, for deretter å utnytte denne når et brukergrensesnitt skal utvikles (Sandnes, 2011, s. 56-57).

2.5.5 Designprosess

Designprosessen består av flere faser der ulike metoder kan bli brukt. Ofte bygger disse metodene på hverandre, fordi de benytter seg av informasjon som avdekkes i tidligere metoder. På denne måten behandles informasjonen flere ganger, og bidrar til å avdekke ny informasjon. I første omgang er det naturlig å gjennomføre intervjuer for å kartlegge brukernes mål og behov. Informasjonen som hentes fra slike intervju brukes blant annet som grunnlag for gjennomføring av PACT-analyse. PACT-analysen konkretiserer og

kategoriserer denne informasjonen slik at personas og scenarier kan utformes med et så godt utgangspunkt som mulig. Personas representerer typiske brukere i målgruppen, og bidrar til å holde fokus på deres behov videre i prosessen. Scenarier er med på å skape et detaljert funksjonelt bilde av løsningen, og kan derfor være et godt fundament for å utvikle prototyper. Prototyper er realistiske skisser for utvikling av en løsning og brukes til å teste ulike idéer og funksjoner. Resultater fra brukstesting vil deretter analyseres og danner grunnlaget for nye iterasjoner. Endringer blir gjort, nye prototyper utformes, og disse testes så for å avdekke feil, mangler og eventuelle forbedringer før en går i gang med utviklingen.

2.5.6 Personas og scenarier

Personas er fiktive personer som representerer typiske brukere av en løsning som skal utvikles. Scenarier er en beskrivelse av aktiviteten en personas skal utføre og i hvilken sammenheng og kontekst denne aktiviteten typisk gjennomføres. Som nevnt, tar disse, ifølge Benyon (2010) gjerne utgangspunkt i tidligere gjennomført PACT-analyse. Grunnlaget for dette kan antas å være fordi en PACT-analyse vil kunne gi en forståelse av løsningens krav. De ulike elementene i analysen kan gi klarere indikasjoner på hva som forventes av både en løsning og en typisk bruker. Samtidig tar den hensyn til flere elementer, blant annet mulige brukere. Dette er med på å gi et grunnlag for å utforme personas og scenarier (Benyon, 2010).

Cooper *et al.* (2014) argumenterer for at personas best utvikles direkte basert på brukerintervjuer og observasjoner. Altså ikke kun PACT-analyse. Dette begrunnes med at nesten alle aspektene ved personas blir tatt fra brukeropplysninger eller oppførsel (Cooper *et al.*, 2014). Selv om dette kartlegges i PACT-analysen, kan det være mer aktuelt å gå tilbake til selve intervjuet. Her hentes informasjonen direkte fra bruker uten å være bearbeidet.

Personas

Å utforme personas er en metode spesielt egnet for designere ettersom det forenkler prosessen ved å tydeliggjøre hvem en designer for og hvilke hensyn en må ta ved ulike avgjørelser og designvalg. Valgene som blir gjort, gjøres derfor på bakgrunn av brukerens preferanser (Benyon, 2010).

Som nevnt er personas et sett med fiktive personer. Disse skal representere typiske brukere av en løsning, og ved bruk av denne metoden bør en utforme mer enn én personas. Årsaken er at ingen mennesker er identiske, selv om de tilhører samme brukergruppe. I de fleste tilfeller vil ikke kun én person bruke løsningen. Personas er gjerne svært detaljert beskrevet, der en utformer en spesifiserende profil som beskriver personlighetstrekk, interesser og andre personlige fakta (Benyon, 2010).

Som fellestrekk blant flere teorier skal en personas-profil inneholde fullt navn, alder, sivilstatus, barn, stillingstittel, interesser, samt en detaljert beskrivelse av personlighetstrekk. Denne beskrivelsen skal gjerne være skrevet sammenhengende som en fortelling og dermed virkeliggjøre denne personen. Dette bidrar til å gjøre en persona

realistisk, noe som er viktig for at denne blir en reell representant for en typisk bruker. Nettopp denne personifiseringen av en troverdig bruker gjør at designere får et mer personlig forhold til brukerne av løsningen som utformes. Bruk av personas gjør også at en unngår *elastiske brukere*, selvrefererende design, og *edge cases* (Cooper *et al.*, 2014).

En elastisk bruker er en bruker som tøyes i de retninger designere føler passer deres løsning. Selvrefererende design er når designere eller utviklere lager en løsning som baseres på deres egne mentale modeller og mål. Edge cases er situasjoner i løsningen som kan oppstå, men som oftest ikke vil skje for de fleste brukere. Det er noe likevel viktig å ta høyde for dette, i design og programmering, men det skal ikke være hovedfokus (Cooper *et al.*, 2014).

Sekundære personas

En sekundær persona har spesifikke tilleggsbehov som kan innlemmes i løsningen, uten at det skal ødelegge for primærbehovet løsningen lages for. Når en lager personas, bør en først fokusere på primær personas, for deretter å gjøre små justeringer for å imøtekomme behovet til en sekundær persona (Cooper *et al,* 2014, s. 89).

Scenarier

Scenarier er beskrivelse av aktiviteter en typisk bruker vil gå gjennom og i hvilke sammenhenger dette skjer ved bruk av en løsning. Scenarier er med på å kartlegge hvordan løsningen blir brukt, samtidig som det bidrar til å skape et bilde på hvordan løsningen skal fungere for at brukeren oppnår målet sitt. Her vil PACT-analysen kunne komme til nytte og kan gi føringer på aktuelle scenarier. Scenarier er gjerne mer detaljert beskrivelse av situasjoner og aktiviteter som er aktuell ved bruk av løsningen, der også målet for bruk blir presentert (Benyon, 2010).

Benyon (2010) presenterer fire typer scenarier; *historier, konseptuelle scenarier, konkrete scenarier* og *use cases*. Disse bygger ofte på hverandre, i rekkefølgen de er presentert:

Historier

Historier er ofte utfyllende og fortellende for å skape et realistisk og detaljert inntrykk av en situasjon som er årsak til at en løsning er aktuell å bruke. De inneholder ofte trivielle fakta og detaljer.

Konseptuelle scenarier

Denne type scenario kan sies å være en abstrahert versjon av en historie der det trivielle og detaljerte gjerne fjernes. Et konseptuelt scenario består som regel av flere historier og viser seg ofte å være nyttig for å generere ideer til til design. Slike scenarier er lite spesifikke med tanke på teknologi og funksjonalitet, men gir gjerne en overordnet forståelse av hva som kreves av løsningen.

Konkrete scenarier

En spesifikk utfordring eller problemstilling er ofte grunnlaget for konkrete scenarier. I et slikt scenario vil en kun inkludere funksjoner som er aktuelle for en konkret situasjon.

Blant annet vil slike scenarier fokusere mer på tekniske aspekter ved design av grensesnitt. Slikt fokus kan være nyttig for å kartlegge om noe fungerer eller ikke, og hvorvidt ulike valg er med på å påvirke brukervennlighet. En bruker gjerne konkrete scenarier som utgangspunkt ved utforming av prototyper ettersom de er svært spesifikke og detaljerte mot løsningens funksjonalitet, og vil derfor kunne brukes som grunnlag for designvalg.

Brukerhistorier

Brukerhistorier brukes i smidige metoder, og er vanligvis korte setninger skrevet slik: *Som bruker ønsker jeg å (...)*. Brukerhistorier er mer som uformelle krav enn scenarier; de beskriver ikke flyten til brukere i et større bilde, og beskriver heller ikke hva brukerens sluttmål er. Dette er avgjørende for å fjerne unødvendige interaksjoner og designe målrettet mot hva brukerne virkelig trenger (Cooper *et al.,* 2014, s. 104).

Brukerreiser

Brukerreiser beskriver hvordan personas bruker en løsning, fra første eksponering til brukerne har nådd målene sine. Brukerreiser blir gjerne lagt opp som fortellinger. Forskjellige reiser viser ulike aspekter av en løsning, ut ifra ulike mål. Brukerreiser gir også designere mulighet til å ta personas gjennom sekundære baner der løsningen hjelper til å rette opp i problemer som kan oppstå underveis (Cooper *et al.,* 2014, s. 136).

Bruksmønster

Use Cases, eller bruksmønster, handler om å beskrive og illustrere interaksjon mellom bruker og løsning. Bruksmønster beskriver detaljert hvordan en løsning brukes ved å gi et bilde på hva en bruker gjør, hva løsningen gjør og hvordan disse to elementene reagerer på hverandres handlinger. Et illustrert bruksmønster trekker linjer mellom bruker og handlingene som utføres (Benyon, 2010). Se vedlegg 16.

Nettsidekart

For å konseptualisere design til en nettside finnes det modeller som hjelper å kartlegge hierarki og flyt. Nettsidekart *(sitemap)* benyttes til slik konseptualisering av en nettside sin struktur, og er et nyttig verktøy for designere. Nettsidekart bistår i å gjøre designere mer bevisst på løsningen detaljer ved en løsning, og ved å eksternalisere objekter og forhold i et design kan en lettere avgjøre hvor effektivt logikken i et design fungerer (Benyon, 2010, s. 214). Rosenfeld, Morville og Arango (2015, s. 374) forklarer i sin bok *Information Architecture* at et nettsidekart avbilder et overblikk av en nettsides struktur og navigering i form av et diagram. Dette diagrammet viser forholdet mellom en nettsides informasjonselementer. Forfatterne forklarer videre at et høyt-nivå nettsidekart er mer komplekst og inkorporerer flere faktorer som gjør at en avbilder ikke bare en topp-bunn navigasjon, men også bunn-topp ved å kartlegge forholdene mellom sidene, i tillegg til forhold mellom innholdskomponenter og sidene.

2.5.7 Skisser og wireframes

Skisser brukes til å skildre idéer. De lages ofte på kort tid, og inneholder få detaljer. De utføres også med billig verktøy, for eksempel papir og blyant. Det viktigste er å fange ideene med en gang de oppstår, og det kan være enklere å tegne de ned, enn å skrive tekster. Ofte tegner en også flere skisser samtidig, slik at det blir en samling som viser omfanget av flere ideer. Hensikten med å skissere, er å stimulere kreativitet, formidle ideer og oppfordre til diskusjon. Dette gjør igjen at nye ideer kan skapes, og en kommer et skritt nærmere målet (Sandnes, 2011, s. 265-266).

Når en er ferdig med å utvikle ideene, er det viktig å få på plass struktur. En wireframe er en enkel teknisk tegning som viser hvor de forskjellige elementene i en nettside skal være, samt viser strukturen i et sett med nettsider (Sandnes, 2011, s. 267).

2.5.8 Prototyper

En prototype er en konkretisering av designerens forståelse av brukernes behov. Dette utformes til en visuell skisse av løsningen som skal dekke de oppfattede behovene. Ved å bruksteste prototypen gjennom hele prosessen, kan en stadig endre brukergrensesnitt etter tilbakemeldinger fra brukstester. Formålet med prototyping er å bekrefte at behov dekkes gjennom den tenkte løsningen, og problemer avdekkes på et tidlig stadie. En minsker sjansen for at sluttprodukt inneholder store feil, og det er større sjanse for at sluttprodukt tilfredsstiller brukerne. Dette er noe som gjøres før selve programmeringen starter, så en sparer tid, penger og innsats (Sandnes, 2011).

I boken *Prototyping: A Practitioner's Guide* forklarer Warfel (2009) verdien av prototyping ved å sammenligne bruk av kravspesifikasjoner og wireframes. Spesielt sammenlignet med en skriftlig liste av spesifiserte krav, vil prototyper visualisere krav tydeligere. Dette gir mulighet for å teste og oppleve om de aktuelle krav er av verdi eller om de kan forkastes. Warfel (2009) er tydelig på hvorfor prototyping er verdifullt tidlig i en prosess; det bidrar til å visualisere idéer og teste ut om disse fungerer eller kan forkastes. Dette støttes i en artikkel skrevet av Cypriano og Pinheiro (2015). Artikkelen diskuterer gjennomgående bruk av prototyping som en iterativ prosessmetode, i motsetning til å bli brukt som et av de siste stegene for å verifisere og teste én aktuell løsning. Artikkelen er skrevet på bakgrunn av et prosjekt der forfatterne erfarer viktigheten av prototyping for deres sluttresultat. Blant annet avdekker brukstesting hvilke elementer som er elementære i grensesnittet, samt hvilke steg som oppleves som overflødige. I stedet for å kode og programmere de ekstra stegene, illustreres disse i enkle prototyper og forkastes før de går videre i prosessen.

Lo-fi prototyper

Lo-fi prototyper er tidlige skisser av den tenkte løsningen, og kan ikke forveksles med en endelig løsning. Dersom en tidlig prototype ser for ferdig ut, kan en kunde feilaktig tenke at utviklingsprosessen nesten er ferdig. Derfor brukes ofte papirprototyper på dette stadiet, slik at det tydelig kommuniserer et uferdig produkt. En papirprototype kan enkelt

endres på, eller forkastes (Sandnes, 2011). Papirprototyper er også raskeste måte å få testet ut ideer på, da de enkelt kan endres eller forkastes.

Hi-fi prototyper

Hi-fi prototyper er mer avanserte prototyper, som også gjerne inneholder koding. Det er ingen ferdig løsning, men fallgruven er at en kan få inntrykket av at det er et ferdig produkt. Det kreves mer ressurser for å lage hi-fi prototyper, men argumenter for å lage en slik prototype er blant annet dersom en skal gjennomføre omfattende uttesting. Altså at et konsept skal prøves ut på en større brukergruppe. Det blir en mer realistisk testing enn ved lo-fi prototyper, da prototypen er mer ekte (Sandnes, 2011). En får også testet i et mer autentisk miljø enn ved bruk av papirprototyper.

2.5.9 Brukstesting

Brukstesting[1], også kjent som det litt mer uheldige ordet *brukertesting*, er en samling av teknikker som brukes for å se hvordan brukere samhandler med et produkt. Målet er vanligvis å måle produktets brukervennlighet. Dette kan blant annet gjøres ved å få brukere til å utføre konkrete oppgaver, og se på hvilke problemer de støter på underveis. Med slike tester synliggjøres ofte svakheter og styrker ved løsningen (Cooper *et al.*, 2014, s. 57)

Gonzotesting

En gonzotest er en test der en oppsøker brukerne der de er, slik at en kan teste løsningen i en realistisk setting. Ved å oppsøke brukerne der de er, kan en få andre resultater enn ved testing i en testlab. Testen blir mindre kunstig ved at brukerne får utføre oppgaver i sitt naturlige miljø, men ulempen er at man bruker tid på å reise rundt, og det kan være vanskeligere å gjøre opptak på en enkel måte (Toftøy-Andersen og Wold, 2011, s. 129).

Planlegging av brukstest

Når en skal planlegge brukstester, bør en ha klare mål som forankres i scenarier. Scenarier kan danne grunnlaget for en eller flere oppgaver gitt til brukere. En konkret oppgave gjør det lettere å forbedre prototypen slik at systemet kan respondere på brukernes forventede handlinger. En godt planlagt og strukturert brukstest gjør det mulig å sammenligne hvordan ulike testpersoner løser samme oppgave (Sandnes, 2011, s. 307).

Forberedelser til brukstest

I tillegg til å planlegge brukstest, bør en forberede selve prosedyren med testingen på forhånd. Dette kan gjøres ved at deltakerne går igjennom hele prosessen med en venn eller en kollega på forhånd – altså en test av testen. På denne måten får en felles

[1] Cooper *et al.* (2014) bruker ordet *usability testing*, istedenfor det mer kjente ordet *user testing*. Vi valgte derfor å oversette det til ordet *brukstesting*, da vi mener dette er et mer beskrivende ord for aktiviteten som gjennomføres. Det er ikke brukeren som testes, det er løsningen og hvordan den er å bruke.

forståelse for hendelsesforløpet, og feil eller utfordringer med gjennomføringen kan identifiseres (Sandnes, 2011, s. 307).

En annen viktig del av forberedelsene, er å finne representative testpersoner. Testpersoner bør ikke være personer med et nært forhold til de som skal utføre testene, da de kan være redde for å uttrykke seg negativt. En person en ikke kjenner, vil være mer tilbøyelig til å komme med kritikk (Sandnes, 2011, s. 308).

Avtale

Før en brukstest gjennomføres, lages en skriftlig godkjennelse som tydeliggjør hensikten med testen, at deltakelse er frivillig, og at testpersonen har mulighet til å trekke seg fra testen på hvilket som helst stadium. Det bør poengteres at resultatene er konfidensielle, og en vil bli anonymisert i eventuelle rapporter (Sandnes, 2011, s. 310).

Gjennomføring av brukstest

Brukstester gjennomføres vanligvis i et rom eller kontor. Det bør ikke brukes av andre samtidig, for å unngå unødige forstyrrelser. Før gjennomføring av en brukstest, tildeles det gjerne forskjellige roller til de som utfører testene. Dette kan for eksempel være testleder og observatør. Testleder fører ordet, og er den eneste som snakker med brukerne under brukstestene. Leder sitter gjerne ved siden av brukerne for å skape et likeverdig forhold, og hjelper brukerne med å tenke høyt. Testleder skal derimot ikke gi hint eller tips til hvordan en oppgave skal løses, da det er prosessen i seg selv som er interessant. En kan la brukerne slite til de står helt fast, da går en videre til neste oppgave (Sandnes, 2011, s. 309-311)

Hvorfor skal man bruksteste?

Brukstesting er en måte å evaluere på, og er spesielt effektiv for å sammenligne ulike designvarianter, slik kan en se hva som er den mest effektive løsningen. Tilbakemeldinger fra brukere er nyttig når en skal validere eller presisere spesifikke designelementer (Cooper *et al.,* 2014, s. 140).

Vanlige ting som testes:
Navngiving – gir navnene på knapper/i menyen mening? Passer enkelte ord bedre enn andre?
Organisering – er informasjon gruppert på en meningsfull måte? Er de plassert på steder der kunder finner det logisk å se etter det?
Førstegangsbruk og synlighet – er vanlig innhold lett å finne for nye brukere? Trengs det instruksjoner? Er de i så fall lett å finne?
Effektivitet – kan kunder effektivt nå målene sine? Begår de feiltrinn? I så fall hvor, og hvor ofte? (Cooper *et al.,* 2014, s. 140).

Evaluering

Etter at brukstest er gjennomført, er det viktig at en møtes for å evaluere hvordan det gikk. På møtet ser en på hvilke utfordringer som testpersonene møtte, og skriver de ned i en egen liste. Dersom en utfordring ble støtt på flere ganger, er dette også vesentlig å få med. Men det viktigste er å finne ut hva som er de mest kritiske problemene. Dette kan gjøres ved å lage en ny liste, der en rangerer problemene. De mest alvorlige problemene er det viktigst at en jobber med, slik at de er fikset til en eventuell neste brukstest (Krug, 2010).

2.6 Webteknologier

For å utforme en web-applikasjon, behøves teknologier som kan fasilitere den ønskede funksjonaliteten. Dette delkapitlet fremstiller teknologier som sammen kan benyttes for å lage en fullverdig løsning for sluttbrukeren. Delkapitlet vil i tillegg belyse konsepter og teknologi som kan brukes for å sikre en kodebase som er gunstig å arbeide i.

2.6.1 Fundamentale språk

Sandnes (2011, s. 350) forklarer i boken *Universell utforming av IKT-systemer* at studenter ofte kan møte en typisk fallgruve hvor hovedkriterier for valg av teknologier og plattformer er at de er "nye og spennende". Utover dette forblir valgene arbitrære, og man kan møte problemer der for eksempel et programmeringsspråk ikke har ordentlig fotfeste og har derfor kanskje ikke tilstrekkelig med dokumentasjon.

Per skrivende dato eksisterer det et mangfold av språk som kan anvendes til utvikling av web-applikasjoner. Stack Overflow – et nettsamfunn som hevder å ha 50 millioner månedlige besøkende Software-utviklere (Stack Overflow, u.å.) – har i 2018 lansert sin årlige undersøkelse kalt *Developer Survey Results 2018* hvor de offentliggjør statistikk som blant annet omhandler de mest populære språkene for programvareutvikling i 2017. Spørreundersøkelsen baserte seg på 101 592 respondenter fra 183 land (Stack Overflow, 2018). Undersøkelsen viser at de tre mest populære språkene er JavaScript, HTML og CSS, alle språk som benyttes til webutvikling. Til tross for at JavaScript, HTML og CSS er en typisk kombinasjon som anvendes til utvikling av web-applikasjoner, brukes de sjeldent alene, ettersom de alle er primært kjent som klient-side språk. For å tjene en bruker med dynamisk data brukes ofte server-side språk som PHP, Python eller C#. Denne konvensjonen har nylig blitt brutt med Node.js, et open source JavaScript-servermiljø introdusert i 2011, som tillater bruk av JavaScript for å legge til rette for server-funksjonalitet.

2.6.2 Preprosesserings-språk

Begrepet *pseudokode* skildrer ikke-kjørbar kode skrevet i en enkel og lesbar form brukt for å forenkle prosessen fra programbeskrivelse til implementering. *Pseudo* betyr noe som fremstår som noe det ikke er (Merriam-Webster, 2018). Å minske gapet mellom

pseudokode og datakode har stor verdi for utviklere ettersom det fremmer lesbar og mer håndterbar kode, noe som også er gunstig for samarbeid. *Open source-scenen* av webutvikling-landskapet tilbyr mange kritiske tilnærminger til HTML, CSS og JavaScript i form av *preprosessor*-språk. Dette er språk med nøyaktig samme hensikt som de respektive originalspråkene, men som fremmer mer ergonomisk og leselig kode, lignende pseudokode.

Et prosesseringsspråk har flere kjennetegn. De er uleselige for en nettleser, og trenger en særegen kompilator for å transformere de tilbake til originalspråket. På denne måten gjøres det mulig å tolkes av en nettleser. Et annet kjennetegn er at de forvandler originalspråkene sin syntaks til et mer lesbart, ofte minimalistisk format og med ekstra funksjonalitet, og vil da ligne mer på pseudokode. *Syntactic sugar* er et begrep innen datavitenskap som skildrer tillegg til et kodespråk som ikke har noen funksjonell effekt, men forandrer hvordan en uttrykker funksjonaliteten. CoffeeScript – en JavaScript-preprosessor – har gjort syntactic sugar til en idiomatisk del av sitt språk for å bidra til å skape leselig kode.

Følgende JavaScript kode:

```
“case "Monday": go(work); break;”,
```

tilsvarer følgende i CoffeeScript:

```
”when "Monday" then go work”
```

CoffeeScript, i tillegg til språkene Pug og Sass, har alle en syntaks som fremmer minimalisme. Dette betyr i praksis at språkene er *whitespace sensitiv*. En *whitespace sensitiv syntaks* betyr at koden sitt hierarki må struktureres med innrykk istedenfor klammer eller tags. Slik syntaks minimerer anstrengelser for utviklere da mindre skriving kreves og vil resultere i programkode hvor alt redundant er skåret av, og inneholder kun det nødvendige. Resultatet er filer som er kortere, og derfor enklere å håndtere for en utvikler.

2.6.3 Byggeprosesser

For å fasilitere bruk av preprosess-kode trenger en et automatisert miljø som kompilerer kode slik at den blir tolkbar for en nettleser. Slikt automatisert miljø blir ofte omtalt som *byggeprosess*.

> A build process is a set of tasks which run over your projects files, compiling and testing code during development and used to create the deployment version of your site (Kearney og Gaunt, 2018).

Kearney og Gaunt (2018) forteller videre i sin artikkel *Set Up Your Build Tools* at den primære rollen til en byggeprosess er å gjøre et skille mellom koden man utvikler i, referert til som *utvikler-versjonen,* og koden brukt i en nettleser, referert til som *produksjons-*

versjonen. En byggeprosess har stor verdi for både utviklerne og sluttbrukerne, ettersom en har frihet til å transformere koder og filer når byggeprosessen beveger de over til produksjons-versjonen. Dette gjør det mulig å legge klare skiller mellom kode som egner seg for utvikler, og kode som best tjener nettleser.

Filoppdeling

En kan selv definere automatiserte oppgaver som transformerer kode i form av å minimere og optimalisere for både hastighet og nettleserkompatibilitet. I tillegg kan en velge å flette sammen flere filer til en større fil. Dette er en gjensidig fordel for utviklere og sluttbrukere ettersom det tillater semantisk oppdeling av filer. Dette er enklere å arbeide med, i tillegg til å være en mindre byrde på sluttbruker ettersom de kun må laste ned én fil. Etterspørsel fra klient til server blir i dag primært håndtert av HTTP/1.1 protokollen, en protokoll som kun kan håndtere én forespørsel av gangen (Chishkala, 2016). En forespørsel kan blant annet være en JavaScript eller CSS-fil, og en vil derfor ikke kunne praktisere å jobbe i mange filer av samme type uten at dette har negativ effekt på sluttbruker grunnet lange sekvenser av forespørsler til server. Byggeprosesser som konkatenerer innhold fra flere filer vil derfor tilby effektiv nedlastning og dermed mindre ventetid for bruker. For å videre optimalisere nedlastingstider kan en også bruke byggeverktøy for komprimering av binære filer, eksempelvis bildefiler, for å sørge for minst mulig belastning på sluttbruker.

Sourcemaps

Å skrive kildekode som ikke representerer produksjonskode – for eksempel CoffeeScript som kompilerer til JavaScript – kan by på utfordringer når en må feilsøke i koden. Om en JavaScript-tolk ikke klarer å prosessere et skript, vil feilmeldingen vises i utviklerkonsollen til en nettleser. Feilmeldingen vil vise til hvilken linje i JavaScript-filen feilen oppstår. Om en ikke skriver direkte produksjonskode, vil en ikke ha kjennskap til filen hvor feilen oppstår, og feilmeldingen vil derfor ikke være hjelpsom.

Sourcemaps er teknologi med formål å tilby feilsøking på kildenivå av optimalisert JavaScript-kode (Lenz og Fitzgerald, 2011). Sourcemaps kan derfor kartlegge innhold i kildekode til innhold i prosessert kode, og derfor gjøre at en nettleser sin utvikler-konsoll kan henvise til hvor en feil oppstår i kildekoden. Sourcemaps er et uvurdelig verktøy om en skal feilsøke og har et vesentlig skille mellom kilde- og destinasjonskode. Om en for eksempel kjører kode gjennom *Babel*, et verktøy for JavaScript best kjent for sin evne til å forvandle ES6 – den neste versjonen av JavaScript – til kode som kjører i en nettleser i dag (Pick, 2015). Ved bruk av byggeverktøy kan man konfigurere kompilatorene for preprosesserings-språk til å produsere sourcemaps slik at feilsøkeprosesser forblir enkle.

Stilsjekkere

Et annet bruksområde for byggeverktøy er innføring av stilsjekkere som vil prosessere kildekoden for å se etter unntak til en definert skrivemåte, noe som hjelper utviklere på et team med å gjennomføre en konsekvent kodestil i et prosjekt.

> Coding standards – Often overlooked, not having coding standards wreaks havoc on collective code ownership as each team member implements his or her own personal style. Create them, publish them, and hang them on the wall to remind everyone (Lacey, 2012, s. 13).

En stilsjekker – også kjent som en *linter* – vil ikke se etter programmatiske feil, med unntak til hva et team anser som en beste praksis for egen kode. CoffeeScript sin linter, kalt CoffeeLint, vil for eksempel kunne gi advarsel om en bruker *&&* istedet for *and* og dermed gjøre det lettere å skrive konsekvent og mer leselig kode.

Byggeverktøy

Med *byggeverktøy* menes programvare dedikert til å kjøre byggeprosesser . Det finnes flere alternativ til byggeverktøy blant utviklere av front-end. Å velge byggeverktøy krever også en kritisk tilnærming på samme måte som å velge kodespråk. Nolan (2016) lanserte en undersøkelse i 2016 hvor det primære målet var å avdekke de mest populære verktøyene brukt blant front-end utviklere. Undersøkelsen baserer seg på 5 254 respondenter. Undersøkelsen viste at de tre dominerende byggeverktøyene var Gulp, NPM-scripts og Grunt. Undersøkelsen viser også til omtrentlige trender ettersom Nolan (2016) sammenligner data fra 2016 med tilsvarende data fra 2015, noe som viser at NPM-scripts hadde en økning på 22,65% siden fjoråret. Nolan foreslår at årsaken til økning skyldes at utviklere ønsker å bevege seg bort fra Gulp og Grunt fordi de regnes som *abstraksjonslag*. På overflaten kan dette virke som et argument som fremmer Gulp og Grunt, men kan i praksis oppleves som et ekstra lag å forholde seg til og kompromitterer derfor frihet i programkoden til et byggeskript.

En annen essensiell del av byggeverktøy er deres respektive økosystemer. GruntJS og Gulp er også modulbasert, men deres pakker lagres og distribueres også under NPM sitt register, dette underbygger hvor godt etablert NPM er i open source miljøet.

Forking er en essensiell del av GruntJS og Gulp sine økosystemer, ettersom en modul som for eksempel CoffeeScript ikke er skrevet for GruntJS eller Gulp, men er forket for å tilpasses GruntJS eller Gulp sin forenklede implementeringsprosess. For at byggeverktøy som GruntJS eller Gulp skal fylle rollen som et *abstraction layer* – som forenkler implementeringsprosesser – behøver moduler å bli forket og tilpasset, noe som medfører en stor svakhet, nemlig at modulene er utsatt for å bli utdatert. CoffeeScript sin kompilator er tilgjengelig som en modul under NPM sitt register, der det eksisterer flere forks av modulen for å tilpasse kompilatorer til andre byggeverktøy. Forfatterne av CoffeeScript modulen er ikke ansvarlige for andre forks av deres modul, og det kreves derfor at forfatterne av en fork eller andre open source entusiaster velger å implementere endringene i original-modulen. Nyeste lansering av CoffeeScript er i per skrivende dato versjon 2.2.4, mens Grunt sin tilsvarende fork, kalt *grunt-contrib-coffee* kun benytter seg av CoffeeScript versjon 2.0.1.

2.6.4 Rammeverk

Prosjektet omhandler intuitiv presentering av data fra en tredjeparts tjeneste – Atlasssian Jira. Dataene som hentes fra Jira sin rest-API må presenteres på ulike måter avhengig av hvordan brukere interagerer med løsningen. Alberto Gimeno (2018) forteller i sin artikkel *The deepest reason why modern JavaScript frameworks exist* at å holde et grafisk brukergrensesnitt synkront med løsningen sin tilstand er en primær årsak til hvorfor JavaScript-rammeverk har blitt så utbredt. Han forteller videre at implementering av kode for å holde DOM-objekt synkront med data-logikk-tilstanden uten et rammeverk krever mye skreddersydd kode for ethvert tilfelle der forandringer kan forekomme. Det vil derfor by på tung kode som ikke er særlig robust (Gimeno, 2018).

Selv når en har kartlagt behov for JavaScript-rammeverk, gjenstår valget om hvilket rammeverk som best egner seg for applikasjonen som skal utvikles. Node, Angular og React rangeres som de tre mest populære rammeverkene under *Frameworks, Libraries and Tools* i Stack Overflow sin utvikler undersøkelse utgitt 2018 (Stack Overflow, 2018). Undersøkelsen dekker rammeverk og andre verktøy som brukes til programvareutvikling generelt, ikke kun webutvikling. Av de totalt tolv listet i undersøkelsen, er Angular og React de eneste JavaScript-rammeverkene som anvendes til front-end. Benitte, Greif og Rambeau (2018) offentliggjør i 2018 sin undersøkelse kalt *The State of JavaScript 2017* med 28 000 respondenter fra verden over. Undersøkelsen gir et helhetlig bilde over omfanget til ulike JavaScript teknologier og utviklerne sine preferanser tilknyttet JavaScript sitt økosystem. Undersøkelsen gir et helhetlig bilde over omfanget til ulike JavaScript-teknologier og utviklerne sine preferanser tilknyttet JavaScript sitt økosystem. Undersøkelsen har en egen kategori for rammeverk til front-end. Deres respektive popularitet er fordelt på kjennskap og erfaring, mer konkret om en har hørt om rammeverket og ønsker å lære det eller ikke, eller om en har erfaring og ønsker å bruke det igjen eller ikke. Rammeverkene med mest oppslutning er React, Angular og Vue. Til tross for at 14 000 respondenter har svart at de er kjent med React og ønsker å bruke det igjen, kontra 4,6 000 for Vue og 4,4 000 for siste versjon av Angular, så svarer hele 12 000 at de har hørt om Vue – og ønsker å lære det. Respondentene i undersøkelsen viser absolutt høyest interesse for å lære Vue fremfor andre rammeverk. Av de totalt 12 rammeverkene listet i Stack Overflow sin undersøkelse, er ikke Vue tilstede under kategorien rammeverk (Stack Overflow, 2018), likevel forteller Ian Allen (2018) i sin artikkel hos Stack Overflow Blog – *The Brutal Life Cycle of JavaScript Frameworks* – at Vue er en av de raskest voksende emneknaggene som brukes på Stack Overflow.

Det som kjennetegner rammeverkene React, Angular og Vue er at de alle kan brukes til å fasilitere ulik arkitektur for å strukturere en applikasjon. Angular og Vue følger en modell kjent som *MVC (Model-View-Controller)*. Formålet med en MVC-arkitektur er å separere applikasjonens datalag under *Model*, brukergrensesnittet under *View* og håndtering av brukerinteraksjoner under *Controller*. View har i tillegg som oppgave å observere *Model* – datalaget – for å oppdatere grensesnitt etter nåværende tilstand på data. Controller skal kunne oppdatere Model avhengig av hvordan bruker interagerer med

siden, blant annet ved endring av brukernavn (Kerr, 2013). React er ikke et MVC-rammeverk, men et bibliotek for å bygge brukergrensesnitt av ulike komponenter som er knyttet opp mot en applikasjon sitt datalag (Hunt, 2013). Facebook – en av vedlikeholderne av React – har laget en egen arkitektur kjent som *Flux*. Salihefendic (2015) forklarer i sin artikkel *Flux vs. MVC (Design Patterns)* at Flux strukturerer flyten i web-applikasjon etter fire sekvensielle steg; *actions, the dispatcher, stores*, og *views* for å skape en enveisflyt. MVC-mønster har som regel ingen strenge regler for dataflyt, og er ofte bidireksjonal.

> MVC did not scale well for Facebook's huge codebase. The main problem for them was the bidirectional communication, where one change can loop back and have cascading effects across the codebase (making things very complicated to debug and understand) (Salihefendic, 2015).

Salihefendic (2015) forklarer videre at alle handlinger – for eksempel brukerinteraksjoner – i en Flux-orientert applikasjon blir sendt tilbake til the dispatcher, som igjen oppdaterer de nødvendige datamodellene i stores. Applikasjonens tilstand som dikteres av stores vil deretter bli reflekteres i views. I et MVC-mønster går både forespørsler og responser gjennom controller.

2.7 Oppsummering

I dette kapitlet har vi vist teori, metoder og verktøy vi mener er essensielle for vårt prosjekt. Dette er grunnlaget for en omfattende prosess for å komme frem til en brukervennlig løsning som også fokuserer på verdien av informasjonen som fremvises. Metodene gir oss mulighet til å ta gjennomtenkte valg og veileder oss videre, samtidig som det sikrer en jevn progresjon. Analyseverktøyene som er utdypet i dette kapitlet, bruker vi videre i neste kapittel, der vi presenterer viktige analyser som kartlegger informasjon vi kan benytte oss av videre i prosessen.

Kapitlet skildrer også konsepter som driver ulike webteknologier – disse kan gi verdi til utviklingsprosessen. Grunnet mangfoldet av teknologier tilgjengelige, vil vi i neste kapittel evaluere styrker og svakheter i de instanser hvor valg mellom kandidater må gjøres.

3. Analyser

3.1 Introduksjon

Dette kapitlet tar for seg analyser av brukerintervju, deretter en grundig PACT-analyse og en gjennomgang av bedriftens eksisterende løsning. Til slutt ser vi nærmere produktmarkedet, med analyser av Jiras og vår løsnings største konkurrenter per skrivende dato.

3.2 Brukerinnsikt

3.2.1 Brukerintervju

Brukerintervju gjennomføres med to prosjektledere for å få innsikt i deres nåværende situasjon, den aktuelle problemstillingen, og deres tanker om hva løsningen skal bidra med. Intervjuene gjøres i en tidlig fase for å sikre en grunnleggende forståelse av utfordringene før oppstart.

Forberedelser til intervjuene gjennomføres ved at intervjuspørsmål skrives ned som åpne spørsmål, samt underspørsmål dersom intervjuobjektene ikke svarer utfyllende nok. Roller fordeles – intervjuer, samt observatører som noterer. For å sikre at alle opplysninger kommer med, gjøres taleopptak etter skriftlig avtale. Et samtykkeskjema formuleres, der formålet med intervjuene står grundig forklart. Blant annet opplyses det om at deltakelse er frivillig, at det gjøres notater og lydopptak, og at intervjuobjektene er anonyme, som nevnt av Sandnes (2011). Se vedlegg 22 for samtykkeskjema.

Intervjuobjektene intervjues i deres miljø, i naturlige omgivelser. Intervjuene åpnes med å fortelle hva formålet er, og intervjuobjektet gis tid til å lese igjennom og godkjenne samtykkeskjemaet. Dersom noe er uklart, vil dette bli tatt hånd om før intervjuet begynner. Deretter vil intervjuer ta kontroll, og stille spørsmålene. Selve intervjuene gjennomføres basert på dialog underveis og ikke kun fastsatte spørsmål. Vedlegg 23 viser intervjuspørsmålene som stilles. Disse er, som nevnt, planlagt på forhånd, men under selve intervjuene går vi dypere inn på hvert tema.

Refleksjon

Etter gjennomførte intervju, fikk vi større innsikt i hvordan hverdagen til de ansatte er, per skrivende dato. Intervjuene ble transkribert i etterkant. Dette har vist seg å være veldig nyttig, ettersom det ga en rikere forståelse da vi leste over det i ettertid. Transkriberingen er ikke vedlagt, ettersom samtykkeskjemaet tilsier at dette ikke skal

offentliggjøres. Videre kommer gruppens refleksjoner av intervjuene, og ut av dette lages en PACT-analyse som presenteres i neste underkapittel.

Styrker og svakheter ved Scrum var et gjennomgående tema i intervjuene. Eksempelvis er det ikke enkelt for bedriften å kjøre flere prosjekter parallelt, eller å benytte seg av Scrum fullverdig ettersom rammeverket har strenge regler hva angår møter, sprinter og backlog. Bedriften tilpasser dermed Scrum til deres behov for å utnytte metoden best mulig. Scrum ble nevnt å ha gode kommunikasjonsmuligheter mellom alle parter i et pågående prosjekt. En annen utfordring er at Scrum ikke tar hensyn til uforutsette ting som kan oppstå underveis i en sprint, både internt i prosjektet og ekstern.

Intervjuene gikk også i dybden på hvordan bedriften benytter seg av Jira, spesifikt hva utfordringene er og hvordan de finner løsninger. Jira har for eksempel flere ulike funksjonaliteter som kan være en hindring for nybegynnere. Bedriften har blitt godt kjent med verktøyet og mener det fungerer godt på de fleste punkter. En av Jiras store svakheter er mangelen på å vise flere prosjekter som pågår samtidig. Jira gir ingen oversikt over hva ressurspersoner gjør for øyeblikket, men Jira løser godt det som omhandler samarbeid og å samle informasjon om oppgaver som skal utføres. Bedriften savner en oversikt over ressurser på tvers av prosjektene, samt en samling av data fra sprintene i Jira. Jira kan kobles opp mot ulike plugins, og bedriften har tidligere benyttet seg av et verktøy kalt *Tempo Planner*. Dette verktøyet ble brukt til loggføring av timer, selv om det tilbyr flere funksjonaliteter, som å allokere ressurser. Bedriften tok en vurdering på at denne utvidelsen ikke var god nok til dette formålet. Dagens løsning på utfordringen om å synliggjøre ressursbelastning, er et Excel-ark der alle prosjekter blir satt opp. Ressursallokering og timeantall registreres her.

Intervjuene avsluttet med en gjennomgang av bedriftens prosess fra nytt oppdrag til ferdigstilte sprinter i Jira. Bedriften har tre ledd: Sprintplanlegging med kunde, deretter samles teamet for å utarbeide en backlog, og til slutt har de et møte der de estimerer hva bedriften kan levere til kunden. Kunder er med på planleggingsmøter, og har alltid det siste ordet.

Bedriften opererer etter Scrums prinsipper om sprinter på tre uker. Ansatte oppsummerer uken på en fast dag på ukentlig basis, og dersom det oppstår overbelastning på en ressursperson blir det satt av ekstra tid kommende uke. Bedriften fokuserer dermed mest på fleksibilitet på ukesbasis, som er et viktig punkt for videre gjennomføring av prosjektet.

På bakgrunn av funnene fra brukerintervjuene, kan en detaljert PACT-analyse utarbeides.

3.2.2 PACT-analyse

Elementene i PACT-analysen som skal kartlegges er personer *(people)*, aktiviteter *(activities)*, kontekster *(contexts)* og teknologi *(technologies)* (Benyon, 2010).

Personer

Hovedbrukere av løsningen vil være prosjektledere eller Scrum Master som har ansvar for å planlegge prosjekter og fordele arbeidsoppgaver. Disse personene jobber til daglig i Jira og er vandt med å bruke dette verktøyet. De har erfaring med prosjektstyring, planlegging og tidsberegning. Det er imidlertid viktig å ta høyde for at disse personene kan være nyansatte og derfor ikke kjent med det aktuelle prosjektstyringsverktøyet. En må også ta utgangspunkt til at disse personene kan være fargeblinde, som er viktig med hensyn til fargebruk som et element i løsningen. Språkkonflikter anses ikke å være mest aktuelt å fokusere på med utgangspunkt i oppdragsgiver. Det kan likevel forekomme ansettelser av personer som ikke snakker norsk, og bør likevel tas i betraktning.

Sekundærbrukere av løsningen kan være ledelsen som ønsker å få innblikk i hvordan prosjekter er organisert, hvilke type ressurser som blir brukt hvor og hvilke av disse som blir mest brukt. Dette kan gi indikasjon på om det er aktuelt å utvide eller fokusere mer i en bestemt retning.

Det tyder på å være de samme, faste personene som vil bruke løsningen, da den hovedsakelig er tiltenkt prosjektledere.

Motivasjonen for å utvikle en slik løsning er å se belastning på hver enkel ressursperson og hvordan dette er fordelt totalt, på prosjekter og per uke. Prosjektledere ønsker en oversikt over hvilke ressurspersoner som er tilgjengelig for aktuelle arbeidsoppgaver på nye prosjekter og se en indikasjon om overbelastning på hver enkel person. Dette for å sikre at arbeidsmengden er gjennomførbar for en person innen aktuell tidsramme.

Tabell 3.1 – Oppsummering: Personer (PACT-analyse)

PRIMÆRBRUKERE	SEKUNDÆRBRUKERE
· Scrum Master · Prosjektleder	· Ledelsen
HENSYN	**MOTIVASJON**
· Ikke alle er kjent med prosjektstyringsverktøyet Jira · Fargeblindhet · Språkforskjeller	· Se belastning på hver en enkel ressursperson · Se hvordan belastningen er fordelt totalt, på prosjekter, på sprinter og per uke

Aktiviteter

Noen aktiviteter som gjennomføres, anses som standard ettersom det er bruk av en web-applikasjon. Datamaskin eller mobiltelefon må skrus på, nettleser må åpnes og adressen til nettsiden må skrives inn. Disse tas det ikke hensyn til i analysering av aktuelle aktiviteter.

Når prosjekter skal planlegges og oppgaver fordeles, bør løsningen indikere om valgt ressurspersonen ikke er aktuell eller om valg av person gir lite rom for uforutsette hendelser. Aktiviteten gjøres ved å opprette nytt prosjekt. Krav, tidsramme og

arbeidsoppgaver fastsettes, deretter må det velges hvilke ressurspersoner som skal gjennomføre de ulike oppgavene. Hvem som skal gjøre hva, når de skal gjøre det, og på hvilket prosjekt. Det betyr at brukeren i første omgang må gjennom flere steg for å få maksimal verdi av løsningen.

Likevel er det ønskelig å få opp en oversikt på ukes- og prosjektbasis, samt per ressursperson. Denne oversikten bør være lett tilgjengelig. Det er ønskelig å kunne filtrere innad i oversikten. En bør derfor ha mulighet til å filtrere på hvilke prosjekter en ønsker å se. Når en klikker seg inn på prosjektoversikt vil en kunne se arbeidsbelastningen på de involverte ressurspersonene, samt filtrere på prosjekter. Normal visning vil være å se alt uten filtrering.

Hovedmålet med løsningen er at prosjektleder enkelt skal få en oversikt uten å måtte bruke tid på å kontakte andre prosjektledere innad i bedriften. Det vil si at samarbeid ikke bør være nødvendig, men det hindrer likevel ikke mulighet for å samarbeide.

Formålet med aktiviteten er å hindre at en ressursperson får for stor arbeidsbelastning enn hva som er gjennomførbart og om et prosjekts sprint er planlagt med for stor arbeidsbelastning. Dette antas å være erfaringsbasert, og en bør være realistisk. Her vil det også være sannsynlig å måtte ta hensyn til arbeidsmiljøloven og hva som er tillatt arbeidsmengde for en person.

Tabell 3.2 – Oppsummering: Aktiviteter (PACT-analyse)

OVERSIKT	NYTT PROSJEKT
· Lett tilgjengelig ved innlogging · Lett tilgjengelig på prosjektsiden · Mulighet for å filtrere · Klikke seg videre inn på prosjekt og sprinter	· Alt av nødvendig informasjon må legges inn · Hvem skal gjøre hva, når og på hvilket prosjekt?
	HENSYN
	· Hva som realistisk og fra erfaring er gjennomførbart · Arbeidsmiljøloven

Kontekster

Aktiviteten foregår ofte på kontor i arbeidstiden; ved oppstart av et nytt prosjekt eller for å kartlegge ukens aktiviteter. Ofte er dette en aktivitet prosjektlederen gjør på egenhånd, likevel kan andre prosjektledere også være tilgjengelige (fysisk på samme kontor), noe som gir muligheter for samarbeid. Bedriften har kontor flere steder, derfor antas det å være mest aktuelt at aktiviteten skjer individuelt.

En utfordring som kan oppstå er dersom en oppgave ikke har samme verdi per storypoint som en annen oppgave. Dette kan være en aktuell utfordring ettersom verdien av et storypoint er relativ i forhold til prosjektet. Ressurspersonene har forskjellige oppgaver på ulike prosjekter. Det kan derfor vurderes om dette vil være representativt i en oversikt som inkluderer flere prosjekter.

Andre utfordringer kan oppstå dersom alt av nødvendig informasjon ikke er registrert på hvert prosjekt og hver oppgave. For at løsningen skal fungere optimalt, bør alle oppgaver ha samme type informasjon lagt inn. Dersom en oppgave ikke har fått et

tidsestimat og tidsfrist, vil dette påvirke oversikten, og den antas å ikke være så troverdig og realistisk som den bør være for å tilfredsstille kravene til bedriften.

I alle sammenhenger der data og informasjon skal lagres om brukere, ansatte og personer generelt, vil det ofte være spørsmål tilknyttet personvern. Krever løsningen slik type informasjon som krever ekstra sikkerhetstiltak ved utvikling og drift, er det viktig å ikke basere løsningen på informasjon som ressurspersonene ikke ønsker oppgi og ha lagret. Lov og regler om personvern må følges.

Tabell 3.3 – Oppsummering: Kontekster (PACT-analyse)

HVOR OG NÅR	UTFORDRINGER
· Innendørs på et kontor · Ved oppstart av nytt prosjekt · Kartlegging av ukens aktiviteter · Oppdatering/kartlegging av status på pågående prosjekter · Foregår oftest alene, men kan skje ved samarbeid	· Ulik verdi på storypoints · Ikke all nødvendig informasjon blir oppgitt per oppgave
	HENSYN
	· Hva som realistisk og fra erfaring er gjennomførbart · Arbeidsmiljøloven · Kan en ressursperson nekte at informasjon som er nødvendig for at løsningen fungerer blir lagret? · Personvern

Teknologier

For at løsningen skal fungere optimalt, forutsetter det at det blir registrert hvem som skal gjøre hva, hvordan belastning dette vil gi i storypoints og når det skal være ferdig. Det vi antar må vises av informasjon er mengde arbeid per person og planlagt tidsbruk på prosjekter og oppgaver, samt belastning per ressursperson. Samtidig bør løsningen også vise hvor tilgjengelig en ressursperson er. Det kan være aktuelt å vise en sammenligning av hvor mye en person har å gjøre og hvor mange storypoints personen er tilgjengelig for – i aktuell periode (dersom det er prosjektvis eller på ukesbasis).

Det meste som brukes av informasjon ligger allerede i Jira, men dersom det er nødvendig med mer tilleggsinformasjon for å optimalisere verdien av løsningen, vil løsningen kunne kreve kommunikasjon med en database på egen server.

Løsningen bør vise data og informasjon på en oversiktlig og lett forståelig måte, slik at konflikter og varsler oppdages og håndteres innen rimelig tid. Vi antar at løsningen ofte vil brukes på desktop, men bør likevel utvikles responsivt slik at den kan brukes på mobil og nettbrett. Det kan være aktuelt ettersom en prosjektleder ofte er i møter og derfor ikke alltid har mulighet til å bruke løsningen på desktop.

Tabell 3.4 – Oppsummering: Teknologier (PACT-analyse)

INPUT	OUTPUT
· Selve oppgaven · Hvor stor oppgaven er – tidsmessig og storypoints · Tidsfrist på oppgaven · Hvem som skal utføre oppgaven	· Hvor mange oppgaver er tildelt per ressursperson · Når skal disse være ferdig (satt opp mot hverandre?) · Hvor mye tid har ressurspersonen til rådighet innenfor aktuell tidsramme · Rangering av ressurspersoner basert på tilgjengelighet · Konflikter og varsler – for mye belastning på én person eller lite rom for uforutsette hendelser
KOMMUNIKASJON	INNHOLD
· Enkel database for å lagre informasjon som ikke kan registreres i Jira · Hente ut nødvendig informasjon fra Jira	· Må vises på en oversiktlig og lett forståelig måte · Ingen rom for å mistolke fremstillingen av belastning, varsler og konflikter · Responsivitet - kan vises på mindre skjermer, selv om det er tiltenkt brukt på større skjermer

3.3 Eksisterende løsning

Slik det kartlegges i brukerintervjuene bruker Escio per skrivende dato Jira i prosjektplanlegging og -gjennomføring, samtidig som de holder oversikt over ressursallokering og timebruk i et Excel-dokument.

3.3.1 Analyse

Vår analyse av deres eksisterende løsning vil inkludere Jira og Excel hver for seg, samt hvordan disse to metodene fungerer sammen i praksis. I analysen går vi gjennom strukturen på hver av de; spesifikt hva de gjør, samt styrker og svakheter ved verktøyene. På bakgrunn av dette, vil vi evaluere dagens løsning og se på hvordan denne evalueringen kan påvirke vår løsning.

Struktur

Jira

Jira er – som nevnt tidligere i rapporten – er et prosjektstyringsverktøy spesielt tilrettelagt for Scrum-prosjekter. Her kan en lage produkt backlog og sprint backlog, der en legger inn detaljerte oppgaver, underoppgaver og oppgavehistorier. Hver oppgave kan tildeles en bruker som er registrert på prosjektet eller under bedriftens profil. Jira gir muligheten til å registrere informasjon knyttet til prosjektet og hver oppgave.

Excel

Excel er regnearkverktøyet til Microsoft. Det baserer seg på tabeller og utregninger. Vi antar at de fleste kjenner til hva Excel er og hvordan det fungerer, derfor ser vi ingen hensikt i å gå dypere inn på Excel sin struktur.

Styrker

Jira

Jira er godt kjent blant medarbeiderne i bedriften. De har brukt verktøyet i flere år. Slik det kommer frem i brukerintervjuene løser Jira godt aspektet med informasjonshåndtering og samarbeid på oppgave- og prosjektnivå. Blant annet når det gjelder historikk, versjonskontroll og analyser i etterkant av et prosjekts ferdigstillelse.

Excel

Slik bedriften bruker Excel, gir det oversikt over tilgjengelige og utilgjengelige ressurspersoner. Samtidig gir det svar på estimert timebruk, samt hvilke prosjektledere og ressurspersoner som er tilegnet hvilke prosjekter.

Svakheter

Jira

Dette verktøyet gir ingen helhetlig oversikt over flere prosjekter som pågår samtidig. Det gir ingen indikasjon om overlapp eller andre konflikter mellom prosjektene. Som tidligere nevnt har ikke bedriften funnet tilleggsverktøy som kan gi et tydelig bilde over hvilke ressurspersoner som er overbelastet, og gir derfor ingen svar på om de er tilgjengelige.

Excel

Excel-arket har ingen kobling mot Jira, og alt av data må derfor legges inn og håndteres manuelt. Selv om slike tabellbaserte regneark gir et ryddig oppsett og god oversikt, oppleves det ikke å gi rask visuell indikasjon på status og eventuelle konflikter.

3.3.2 Evaluering

ikke optimal. Bakgrunn for denne vurderingen er at de to verktøyene som benyttes ikke kommuniserer med hverandre og må derfor brukes separat. En optimal løsning mener vi vil være å utvikle et system som fungerer sømløst med Jira, ettersom oppgaven er å utvikle et system som skal gi oversikt, kartlegge ressursfordeling og foreslå løsninger for bedre allokering av ressurser for prosjekter i parallell. En fremtidig løsning vil kunne erstatte Excel-arket som presenteres i analysen.

3.4 Konkurrentanalyse

For å kunne plassere produktet i markedet, benytter gruppen seg av konkurrentanalyser. Disse analysene spiller en viktig rolle i å forstå markedslandskapet rundt løsningen.

Gruppen ser på konkurrerende tjenester til Atlassian Jira, hvilket problem produktet skal løse, samt tilsvarende tjenester som kombinerer begge produktene. For å

systematisere resultatene, benytter gruppen seg av SWOT-analyse. Analysen beskriver styrker, svakheter, muligheter og trusler for en bedrift (Kotler, 2005).

Det fokuseres på tjenester som dekker samme behov og tiltrekker de samme kundene. Samlet ser vi på fire ulike tjenester, som anslås å være konkurrenter til løsningen. Kriterier er tilsvarende tjenestetilbud som Jira og andre løsninger som viser ressursbelastning.

3.4.1 Konkurrent 1: Wrike

Første konkurrent er Wrike (www.wrike.com), en programvare for prosjektstyring og samarbeid. Vi anser Wrike som en av de største konkurrentene, ettersom de tilbyr en fullverdig løsning med statusoversikt over oppgaver, ressursbelastning på ansatte og valg mellom flere ulike maler. Deres svakhet i forhold til oppdragsgivers ønske er at Wrike ikke er kompatibel med Jira.

Tabell 3.5 – SWOT-analyse av Wrike

STYRKER	SVAKHETER
· Enkelt og moderne grensesnitt · Templates & Gantt-diagram innebygd · Interaktive rapporter · Oversiktlig · Gratis for små team (1-5 personer) · Har mulighet til å vise ressursbelastning over ansatte	· Dyrt per bruker ($ 10/24 per bruker per måned) · Ikke kompatibel med Jira
MULIGHETER	**TRUSLER**
· Bli kompatibel med Jira · Mer datavisualisering · Mobilvennlig applikasjon	· Tempo Planner · Workday · Jira · Trello

3.4.2 Konkurrent 2: Tempo Planner

Tempo Planner (www.tempo.io) er en utvidelse til Jira, og tilbyr visning av ressursbelastning. Denne utvidelsen ble brukt av oppdragsgiver tidligere, men ble ikke ansett som god nok til dette formålet, blant annet fordi de har for mange funksjoner som ikke var av verdi for oppdragsgiver.

Tabell 3.6 – SWOT-analyse av Tempo Planner

STYRKER	SVAKHETER
· Kompatibel for Jira · Mobilvennlig applikasjon · Drag&Drop timeline · Billig per bruker	· Rotete med for mange funksjoner · Kan bli dyrt i sammenheng med Jira
MULIGHETER	**TRUSLER**
· Mer datavisualisering/statistikk · Mer moderne design	· Workday · Wrike · Trello

3.4.3 Konkurrent 3: Workday

Workday (www.workday.com) er en konkurrent som tilbyr mange av de samme mulighetene som de foregående konkurrentene, men fokuserer mer på en enhetlig løsning med fokus på finans.

Tabell 3.7 – SWOT-analyse av Workday

STYRKER	SVAKHETER
· Detaljert analysering · Minimalistisk design · Oversiktlig · Mobilvennlig applikasjon · Tilbyr mer: Finans/lønninger	· Fant ikke pris · Ingen enkel måte å registrere en konto
MULIGHETER	**TRUSLER**
· Bli kompatibel med Jira	· Tempo Planner · Jira · Wrike

3.4.4 Konkurrent 4: Atlassian Jira

Jira (www.atlassian.com/software/jira) er godt etablert på markedet, og benyttes av oppdragsgiver. Atlassian er et stort selskap som tilbyr blant annet Confluence, BitBucket og Jira. Den store fordelen er at disse verktøyene samhandler godt med hverandre.

Tabell 3.8 – SWOT-analyse av Jira

STYRKER	SVAKHETER
· Har lang fartstid på markedet · Er kompatibel med mange verktøy (Confluence, BitBucket o.l.) · Har mange utvidelser · Billig per bruker	· Rotete med for mange funksjoner · Tar lang tid å lære nok til å ha fullt utbytte · Unødvendig kompleksitet
MULIGHETER	**TRUSLER**
· Mer oversiktlig og brukervennlig design · Vise ressursbelastning	· Workday · Wrike · Trello

3.4.5 Evaluering av SWOT-analyse

Etter å ha gjennomført en SWOT-analyse av Jira og utvalgte konkurrenter, ser vi at Jira har en fordel med å være underlagt et større selskap som kom tidlig på markedet. Andre konkurrenter kan dekke de samme behovene, men ut ifra bedriftens ønsker kan de ikke utkonkurrere Jira i fullverdig grad.

3.5 Valg av webteknologier

I *kapittel 2 – Teori, metoder og verktøy* ble det fremstilt ulike webteknologier som potensielt egner seg for løsningen. Flere teknologier kan dekke samme behov, og derfor presenteres gruppens refleksjoner i de situasjoner der vi må velge mellom ulike kandidater.

3.5.1 Språk

HTML, CSS og JavaScript er per skrivende dato de kodespråkene med størst fotfeste blant programvareutviklere (Stack Overflow, 2018), HTML ble oppfunnet i 1989, etterfulgt av CSS i 1995 og JavaScript begynte å ta form i 1996 – kun ett år etter (Webb, 1999). Det faktum at språkene har vedvart fra slutten av det 20. århundre til 2018, i tillegg til å være de mest populære språkene blant programvareutviklere er et testament til språkene sin utholdenhet og tilpasningsevne. En slik utholdenhet forteller at en kan oppnå et godt utgangspunkt for utarbeiding av fremtidssikre nettsider. Karakteristikker som utholdenhet og språkenes universelle bruk, underbygger en trygghet. Dette betyr i praksis at en har mangfoldig dokumentasjon tilgjengelig, som bevises av Stack Overflow sin undersøkelse.

De tre språkene er en typisk kombinasjon som anvendes innen webutvikling, men som sjeldent brukes alene. For å fasilitere server-funksjonalitet, brukes ofte andre språk som for eksempel PHP, Python eller C#. I interesse av å ikke bruke språk der det er vanskeligere å forutsi deres grad av adopsjon i fremtiden, kan det være av verdi å ikke ta med flere språk enn HTML, CSS og JavaScript. For å håndtere server-funksjonalitet kan en derfor bruke Node.js, som er JavaScript-basert. Ved å begrense antall unike språk som innføres, avverger en også risikoen ved å ha et høyere kompetansekrav som potensielt kan stagnere utviklingen i et prosjekt. Å bruke Node.js til server-funksjonalitet, tillater mer JavaScript-basert utvikling og skaper et mindre kompetansekrav som senker terskelen for å bidra.

I tillegg til å operere med de ovennevnte språkene som er godt etablert, kan en fortsatt gjøre seg flere kritiske tilnærminger, som bevist av preprosesserings-språk. Pug, Sass og CoffeeScript plasserer et sterkt fokus på syntactic sugar og en whitespace sensitiv syntaks for å skape en kode som er semantisk, lesbar, minimal og håndterbar. De tre preprosesserings-språkene tilbyr også tilleggsfunksjonalitet utover sine respektive originalspråk. Dette antydes å medbringe verdi internt i et team, da alle opererer med lesbar, minimalistisk kode og kan i tillegg være gunstig for nykommere, fordi det vil være enklere å engasjere seg i en allerede eksisterende kodebase.

3.5.2 Utviklermiljø

For å automatisere byggeoppgaver i en kodebase, blant annet for å kunne fasilitere bruken av preprosesserings-språkene, må et utviklermiljø benyttes. Som vist til i *delkapittel 2.6,* er Grunt, Gulp og NPM de dominerende alternativene. NPM har et stadig voksende open source-samfunn, med over 600 000 moduler tilgjengelig, og en gjennomsnittlig vekst på 494 nye moduler per dag (DeBill, 2018). På grunn av NPM sitt fotfeste, følger det en trygghet der en vet at en har direkte tilgang til et mangfold av moduler og deres API. Å ikke bruke abstraksjonslag som Grunt og Gulp gjør at en kan avverge typiske risikoer som utdaterte moduler eller teknologi som ikke har blitt tilpasset deres respektive implementeringsmønster. Å bruke NPM kontra å ikke begrense seg til et tredjeparts byggeverktøy sine implementeringsrammer, gir større frihet til å utvikle et miljø tilpasset ens egne prinsipper rundt kodekvalitet.

3.5.3 Rammeverk

For å strukturere flyten mellom data i en web-applikasjon, er det gunstig å inkorporere et JavaScript-rammeverk. Det er høy konkurranse mellom slike rammeverk, hvor Angular, React og Vue er mest utbredt per skrivende stund. En faktor som er viktig å ta hensyn til, er rammeverkene sine respektive størrelser. En minst mulig kodebase betyr raskere lastetider – en viktig faktor for en god brukeropplevelse. Angular, React og Vue er klientside-rammeverk, og deres størrelser vil direkte påvirke en sluttbruker.

Å lage en lettvektig løsning er en høy prioritering for oss. Angular har en filstørrelse på 143 kilobyte, React på 43 kilobyte og Vue står ved 23 kilobyte (Cuelogic Insights, 2018) – Vue er her klart minst. Størrelsen til de respektive rammeverkene kan vokse avhengig av funksjonalitet som ønskes, da alle støtter tilleggspakker. Å vite at Vue med sin mindre filstørrelse også støtter tilleggskomponenter underbygger en trygghet når en skal bygge en web-applikasjon da kompleksiteten kan øke proporsjonalt med kodebasens størrelse.

3.6 Oppsummering

I dette kapitlet har vi gjennomført en PACT-analyse basert på brukerintervjuer. Vi har sett på konkurrentene til Jira, og evaluert styrker og svakheter ved ulike webteknologier. Disse analysene gir verdifull innsikt som kan brukes videre i designprosessen til å utvikle prototyper av løsningen. Dette ser vi nærmere på i neste kapittel.

4. Designprosess

4.1 Introduksjon

I foregående kapittel ser vi informasjonen som avdekkes i analysemetodene vi benytter oss av. Denne innsikten anses å være essensiell for å bygge videre på en grundig prosess med utforming av prototyper. I dette kapitlet legger vi frem hvordan vi har gjennomført idéutvikling med utgangspunkt i metoder som personas og scenarier, brukerreiser og prototyping. Vi vil gå nærmere inn på hvordan vi bruker metodene og hvordan dette har innflytelse på designvalg.

4.2 Idéfase

4.2.1 Personas og scenarier

Basert på brukerintervjuer og PACT-analyse, utarbeides det tre personas med tilhørende scenarier (vedlegg 13). Vår første persona representerer hovedmålgruppen, da han er prosjektleder og trenger en oversikt over prosjekter som blir kjørt i parallell for å få et bedre inntrykk av den nåværende belastningen. Dette er oppdragsgiver sitt primære fokusområde i følge oppgavebeskrivelsen. Det er også laget to personas i tillegg, som er sekundære. De henvender seg mer mot spesifikke deler av løsningen som de kan dra ekstra nytte av. Sekundære personas og scenarier knyttet til disse, har vi tilegnet oss kunnskap om via samtaler med prosjektledere og ansatte i bedriften. Det andre personas skildrer en medarbeider som ikke ønsker å oppleves som konfronterende, og velger derfor ikke å opplyse prosjektleder om eventuelle belastninger. Denne personas ble utformet for å understreke at løsningen kan bidra til et arbeidsmiljø som er mer transparent. Det tredje personas viste til medarbeider med rolle å opprettholde god kommunikasjon med kunder av oppdragsgiver. Oppdragsgiver har informert om at kunder som ønsker å endre et oppdrag sine spesifikasjoner underveis kan by på utfordringer, i form av uforventet arbeid. Noe som de kanskje ikke har kapasitet til. Det tredje personas vektlegger at løsningen må utformes med et intuitivt grensesnitt og et minimalistisk design slik at løsningen kan doble som et presentasjonsverktøy. Å gi kunde innsikt i bedriften sin interne belastning vil bidra til en ærlig og inkluderende dialog mellom bedrift og kunde.

4.2.2 Brukerhistorie

I samarbeid med oppdragsgiver lages det tre ulike brukerhistorier, se vedlegg 14.

Disse hjelper med å konkretisere hva brukere egentlig vil oppnå med løsningen. Brukerhistoriene legges inn i verktøyet Jira og markeres som *story*. En story i Jira betyr oppgaver som ofte går over flere sprinter som kort beskrivelser vesentlig overordnet funksjonalitet – ofte brukerhistorier – som kan brytes ned i mindre oppgaver. Brukerhistoriene blir referansepunkter under prosjektarbeidet for å hjelpe å realisere ønsket funksjonalitet.

4.2.3 Brukerreiser

Brukerreiser utarbeides og baseres på den generelle rollen og scenariet til primær personas, lagt i vedlegg 15. Reisene viser alle stegene fra brukeren oppdager behovet, til målet er nådd – som nevnt av Cooper *et al.* (2014, s. 136).

Brukerreiser lages i en tidlig fase, før utviklingen av løsningen er påbegynt. Dette gjør at vi danner et bilde av hvordan løsningen kan fungere, samt kartlegge videre steg i prosessen, slik Benyon (2010) presiserer.

Videre i prosessen evaluerer vi brukerreiser med oppdragsgiver. Vi diskuterer blant annet hvordan brukerreisene oversettes til teknisk utvikling, og kompleksiteten dette vil medbringe. Eksempelvis velger vi å nedprioritere brukerreisen for omrokkering av oppgaver, se vedlegg 15. Dette fordi funksjonaliteten vil øke kompleksiteten betraktelig når teknisk utvikling begynner. Oppdragsgiver informerer om at løsningen burde formes etter hva som var realistisk med tanke på tidsbegrensninger. Brukerhistorien for omrokkering av oppgaver ble derfor naturlig å nedprioritere.

Fordelen med å lage flere brukerreiser som baserer seg på ulike brukerhistorier, er en løsning der grunnlaget er lagt i realistiske evalueringer, aktiviteter og situasjoner. I tillegg kan flere brukerreiser vise ulike perspektiv av løsningen, dersom man tar utgangspunkt i ulike mål (Cooper *et al.*, 2014, s. 136).

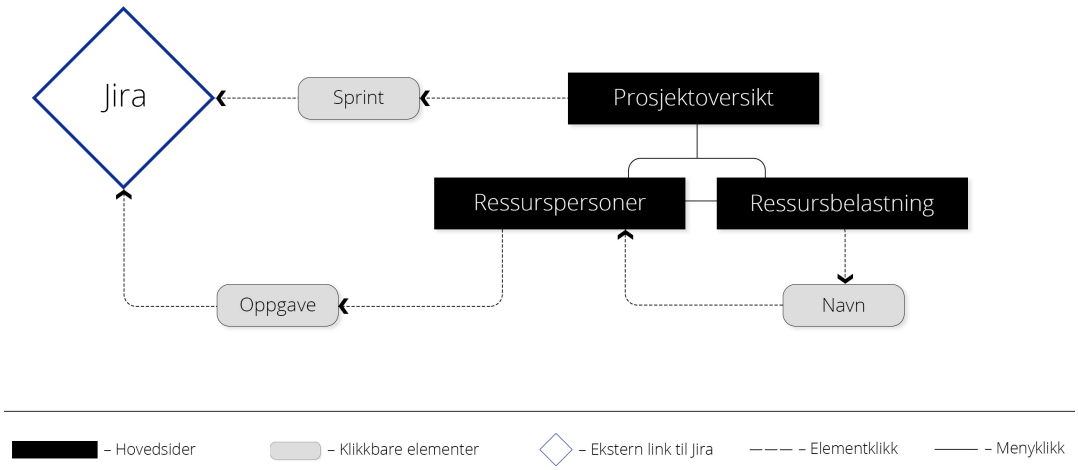
4.2.4 Bruksmønster

To bruksmønster utformes, inspirert av brukerreiser. Eksempelvis viser vedlegg 16 alle steg i prosessen fra prosjektleder åpner applikasjonen og skal allokere oppgave, til arbeidsoppgave er tildelt en ressursperson. Et annet bruksmønster viser i vedlegg 16 alle steg i prosessen fra en overbelastning er oppdaget, til bruk av applikasjonen er fullført. Det er ønskelig å nå målet med så få steg som mulig, for å minimere kompleksiteten til løsningen. Dette antas å gi et godt utgangspunkt for videre skissering.

4.2.5 Nettsidekart

For å få et overordnet blikk over løsningen, utarbeider vi et høyt-nivå nettsidekart. Målet er å forme klare indikasjoner på de ulike rutene brukeren kan benytte seg av for å navigere siden – på så få klikk som mulig. I vedlegg 17 ser man oversikten over løsningen med de ulike undersidene, samt hvordan innholdskomponentene forholder seg til hovedsidene. Nettsidekartet kartlegger hierarki og flyt, og viser forholdet mellom informasjonselementene. Nettsidekartet itereres på fortløpende tidlig i prosjektarbeidet,

eksempelvis viser nyeste iterasjon av kartet hvordan vi ønsker å skape samhandling mellom undersidene *ressurspersoner* og *ressursbelasnting*. Vi illustrerer ruter hvor en kan klikke på innholdskomponenter som gjengir ansattnavn i *ressursbelastning*, for å føre brukeren videre til *ressurspersoner*. Vi planlegger at en slik navigering skal sentrere siden på den ansatte som tidligere ble trykket på. Dette bidrar til samhandling mellom sidene, hvor alternativet vil være to handlinger; å klikke på navigasjons-menyen etterfulgt av å bla nedover til ønsket person. Videre så avklarer nettsidekartet også hvordan elementer utenfor vår løsning, herunder Jira, samhandler med innholdskomponenter i løsningen.



Figur 4.1 – Nettsidekart.

4.2.6 Skissering og wireframes

For å begynne konkretiseringen av våre nåværende konsepter og tanker, skisserer alle gruppemedlemmene nøkkelfunksjonalitet basert på bruksmønstre. Skissene blir presentert og diskutert i plenum blant alle gruppemedlemmer for å opparbeide en felles forståelse av løsningen og dens grensesnitt, samt funksjonalitet Sandnes (2011) påpeker at skisser blant annet brukes til å fange ideer og formidle disse. Alle skissene blir presenteres for oppdragsgiver, se vedlegg 18. Styrker og svakheter blir derfor evaluert sammen med oppdragsgiver. Slik sørger vi for at alle varianter av en mulig løsning blir vurdert og vi kan velge bort skissene som ikke avbilder viktig eller aktuell funksjonalitet. Tilbakemeldinger fra oppdragsgiver farger våre valg i kommende iterasjoner. Denne fasen er spesielt viktig for å skape både et visuelt bilde av løsningen og en ytterligere forståelse av oppgavens funksjonelle og tekniske krav.

4.2.7 Utvikling av lo-fi prototype

Lo-fi prototype utvikles som en videreføring av de første skissene, de tegnes for hånd og bruker farger for et høyere detaljnivå, se vedlegg 19. Denne prototypen representerer de mest fornuftige løsningsforslagene fra tidligere skisser. Prototypen har to undersider, der

målet er å presentere logiske samlinger av data fordelt på minst mulige klikk i navigeringen. Når en legger til rette for at brukeren skal utføre handlinger på færrest mulig klikk, sikrer dette sømløs og effektiv bruk av løsningen. Samtidig oppnår brukeren målet sitt raskest mulig.

Siden for prosjekter kartlegger belastning, dette symboliseres med farger, der rød blir ansett som overbelastning, Grønt er trygt planlagte sprinter, og gul farge viser potensielt overbelastede sprinter. Den neste siden viser allokering av oppgaver hos ressurspersoner. Oppgaver er plassert i en tidslinje-visning for å gi et innblikk i en ressursperson sin gitte arbeidsmengde fram i tid. Vi velger også å bruke kontraster for å understreke oppgavene sine respektive statuser; to-do og in-progress. Dette for å gi fokus på urørt arbeid ettersom det enklere kan allokeres på nytt. Siden har en sidebar med mulighet for å filtrere på prosjekt og sprinter. Sidebaren inneholder også egen sjekk-boks for en storypoints visning som transformerer siden til et stolpediagram. Stolpediagrammet vil presentere storypoints-totalen for enhver ressursperson.

Lo-fi prototype testes ut innad i gruppen, samt på oppdragsgiver. Tilbakemeldinger inkluderer at innhold i sidene burde separeres ettersom visningen for oppgaver oppleves som unødvendig kompleks. Tilbakemelding viser også en kritisk holdning til bruk av kun farger for å vise overbelastning. I ettertid ser også vi at denne løsningen ikke er optimal på grunn av farger som eneste visuelle kobling, noe som ikke er god praksis mot universell utforming.

4.2.8 Utvikling av hi-fi

Ved utvikling av hi-fi prototype for videre testing og iterering, gjøres dette med prototypingsverktøyet Adobe Xd. En av årsakene til valg av dette verktøyet, fordi det er et lettvektig og intuitivt verktøy som på en enkel måte gjør prototypen klikkbar. Prototypen kan også deles videre over skyen, noe som er viktig for samarbeid.

For å skape et inntrykk av design og plassering av elementer, benytter vi oss av moodboards, se vedlegg 20. Der settes ulike bilder som representerer et tema sammen for å skape inntrykk og stemning, som brukes som inspirasjon videre i prosessen. Dette kan bidra til å gjøre ulike designvalg, fordi en ser om noe fungerer uten å ha prøvd det selv.

For å få et rent og minimalistisk uttrykk velger vi å prøve ut svart bakgrunn i vår løsning. Dette er et valg som baserer seg på løsninger vi har inkludert i moodboardet (vedlegg 20). En annen årsak til valg av mørk bakgrunn er for å differensiere oss fra Jira sitt design som per skrivende dato er lyst og hvitt. På denne måten sikrer vi også at løsningen ikke mistolkes som å være en versjon av, eller et forsøk på å erstatte Jira.

Innhold i rutenettet blir gruppert i henhold til gestaltlovene om nærhet og likhet, slik at en enklere kan se hvilke prosjekter og sprinter som hører sammen. Se figur 4.3, s. 52 som vises senere i dette kapitlet. Her plasserer vi elementer i sidemenyene i forhold til gestaltloven om nærhet (Sandnes, 2011, s. 67-70).

Den første hi-fi prototypen testes under første brukstest. Den er interaktiv slik at vi kan gjennomføre tester som gir verdifull tilbakemelding knyttet til interaksjon. Slik unngår vi

å måtte fortelle brukeren at de skal forestille seg at noe skjer. Den første brukstesten er viktig for oss slik at vi kan kartlegge om valg og ulike løsninger fungerer. Samtidig gir det oss tilbakemelding på om vi skal iterere designet utfra de valgene vi allerede har gjort, eller om vi bør gå for sikrere valg, som for eksempel lysere farger. Ved å gjennomføre gode, planlagte brukstester unngår vi å bruke tid på å kode funksjoner og elementer som senere vil bli forkastet. Med god planlegging og utførelse av brukstestene mener vi å legge til rette for at realistiske bruksscenarier blir testet og all informasjon som kommer frem i testene blir notert og arkivert. Slik kan vi i ettertid analysere og bearbeide denne informasjonen for å skape enda bedre løsninger.

Universell utforming

Universell utforming er sentralt under utviklingen av hi-fi prototype. Særlig når det kommer til farger og kontrastforhold mellom tekst og bakgrunn. Bruk av farger er ikke det eneste bruk av visuell kobling mellom elementene på siden, vi har brukt det som et tilleggselement (WCAG 2.0).

Valg av skrifttype

Skrifttypen som blir brukt er Open Sans. Dette er en grotesk skrifttype, optimalisert for både web og mobile løsninger (Google, 2017). Open Sans har en åpen form og god lesbarhet, samt et utvalg av ti ulike skriftsorter. Skrifttypen er også tilgjengelig via Google sitt skriftbibliotek – Google Fonts. Dette legges til grunn for valg av skrifttype.

Videre prosess

Hi-fi prototypen er blitt endret flere ganger gjennom prosessen, og for hver iterasjon bidrar den til å verifisere alle vesentlige funksjoner som allerede er implementert i løsningen. Brukstesting avdekker ikke kun svakheter, men også nye idéer, funksjoner og elementer som kan implementeres for å løse ulike problemer og feil som oppstår underveis (Cooper *et al.*, 2014). I regelmessige møter med produkteier er det stort fokus på prototypen, der vi diskuterer valg vi gjør samtidig som det sørger for at systemkravene oppfylles. Gjennom ukentlige møter med veileder er fokuset på brukeropplevelse, universell utforming og visuelt design. Veileder underviser i emner relatert til dette, som nevnt i *kapittel 1* under *1.7 – Øvrige roller*. Disse møtene bidrar til at prototypen inneholder flere viktige aspekter kan sikre en fungerende prototype som er så nær ferdig løsning som mulig.

4.3 Brukstesting

4.3.1 Første brukstest

Første brukstest utføres på fem tilfeldige personer med kjennskap til emnet programvareutvikling. De kjenner til smidige metoder, men ikke nødvendigvis verktøyet

Jira og konseptet storypoints – som løsningen er basert på. Relevant bakgrunnsinformasjon om dette forklares derfor til hver enkelt person før brukstesten startes.

Bakgrunnsinformasjon som gis til testpersoner

Testperson får en kort visuell presentasjon av Jira, med tilhørende forklaring av at det er et prosjektstyringsverktøy som håndterer prosjekter der smidige metoder blir brukt. Det forklares også at Jira blant annet bruker storypoints, et konsept der en angir poeng til hver oppgave, basert på antatt størrelse på oppgaven.

Jira mangler i dag en måte å få oversikt over belastning, som er relevant når en kjører flere prosjekter i parallell. En vil unngå overbelastning, både i sprinter og ressurspersoner for at ansatte ikke skal bli overarbeidet. Slik kan en gi kunden en realistisk deadline. Formålet er at vår løsning skal løse denne utfordringen.

Scenario

Det gis kun et scenario fordi dette gir et utgangspunkt til videre brukstester, ettersom prosjektet fortsatt er i en tidlig fase. Brukerens atferd under denne testen kan gi oss indikasjon på hva som eventuelt senere bør testes grundigere.

> Scenarie til brukstest:
>
> *Du skal tre inn i rollen som produkteier, og du er ute etter å få en oversikt over belastning på prosjekter og ressurspersoner.*

Tilbakemeldinger og observasjoner under brukstest

Vi observerer at nesten alle testpersoner prøver å trykke på navnet til en ressursperson når de skal finne ut av hva vedkommende har av oppgaver. Samtlige testpersoner klikker mye frem og tilbake på storypoints-siden og ressurspersoner-siden for å få den informasjonen de er ute etter.

Mange testpersoner forstår ikke forskjellen på progresjonsbarene og den visuelle forskjellen på to-do og in-progress. Flere forteller at de trenger et større visuell skille som er mer logisk for å differensiere disse. Det kom også frem at det var forvirrende at det var fullt navn på ressurspersoner-siden og kun initialer på storypoints-siden.

Enkelte testpersoner ser ikke behovet for prosjektsiden. Flere testpersoner foreslår at en bør kunne se totalt antall storypoints som er tildelt en ressursperson. En testperson nevner at å flytte på oppgaver bør kunne gjøres enkelt med drag-and-drop. Flere testpersoner ønsker en tidslinje på storypointssiden for å kunne anslå om en ressursperson faktisk er overbelastet. Vi gjøres oppmerksomme på at scrolling på ressurspersoner kan være tungvint.

Nesten alle testpersoner kommenterer uoppfordret på designet. Fargene og den visuelle fremstillingen er fin, det må gjøres enda tydeligere.

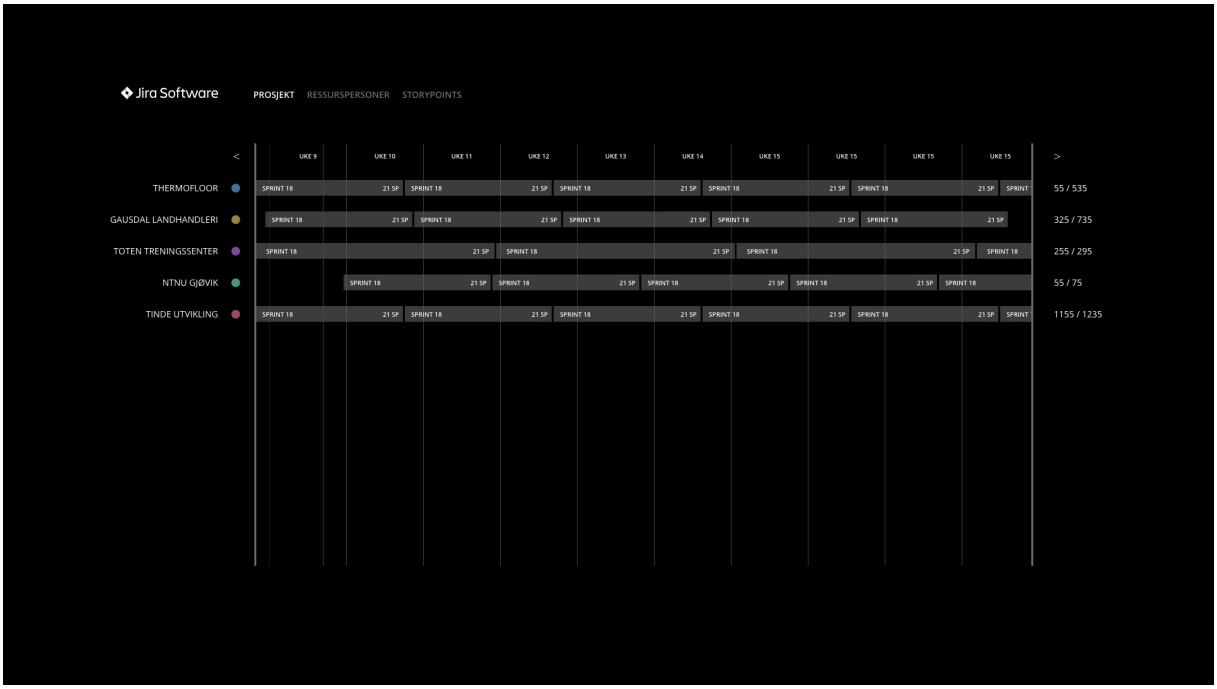
Evaluering etter brukstest

Etter gjennomført brukstest, lærte vi at de største utfordringene var å tydeliggjøre filtreringen for å unngå misforståelser. Samtidig må informasjon fordeles på en logisk måte mellom sidene som omhandlet storypoints og ressurspersoner. Dersom tiden strekker til frem til neste brukstest, er det også mulig å se på hva som skal skje dersom en klikker på et navn.

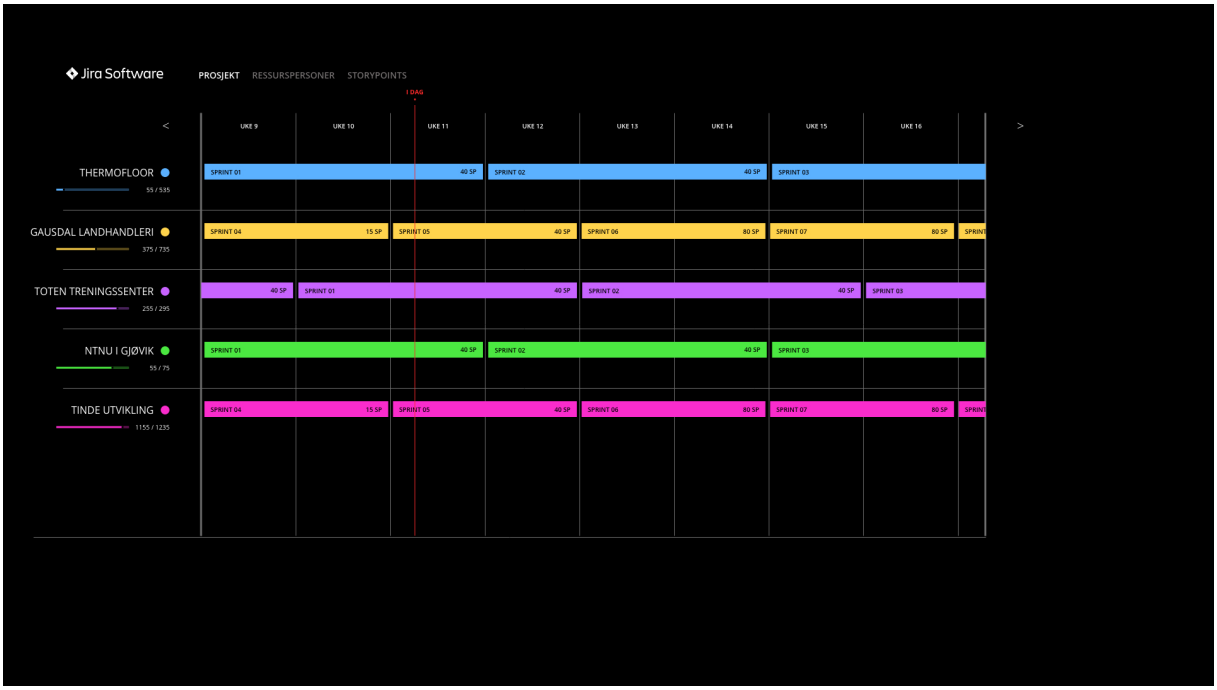
Iterering på prototype etter første brukstest

På bakgrunn av tilbakemeldinger fra testpersoner, diskuteres ulike løsninger. Sterkere farger implementeres, da duse farger forsvinner i den mørke bakgrunnen. Elementer separeres mer i henhold til gestaltloven om nærhet (Sandnes, 2011), og det legges til en indikasjon på dagens dato. Dette mener vi vil tydeliggjøre status på prosjekter, da det bidrar til å se hvor i prosessen en er, og kan gi en indikasjon på at alt som ligger bak dagens dato er noe som burde vært ferdigstilt.

I prosjektsiden får prosjektene progresjonsbarer for å fjerne tallene til høyre. Dette gjøres for å synliggjøre status på de ulike prosjektene. Her utarbeides flere versjoner av progresjonsbarer, se figur 4.4. Ved at flere utforminger av samme element gjøres, kan vi drøfte hvilke som fungerer best, samt hvorfor andre ikke fungerer. Sprinter på hvert prosjekt får også farger, for å koble prosjekt og sprint tydeligere sammen. Slik kan prosjektene raskt separeres fra hverandre.



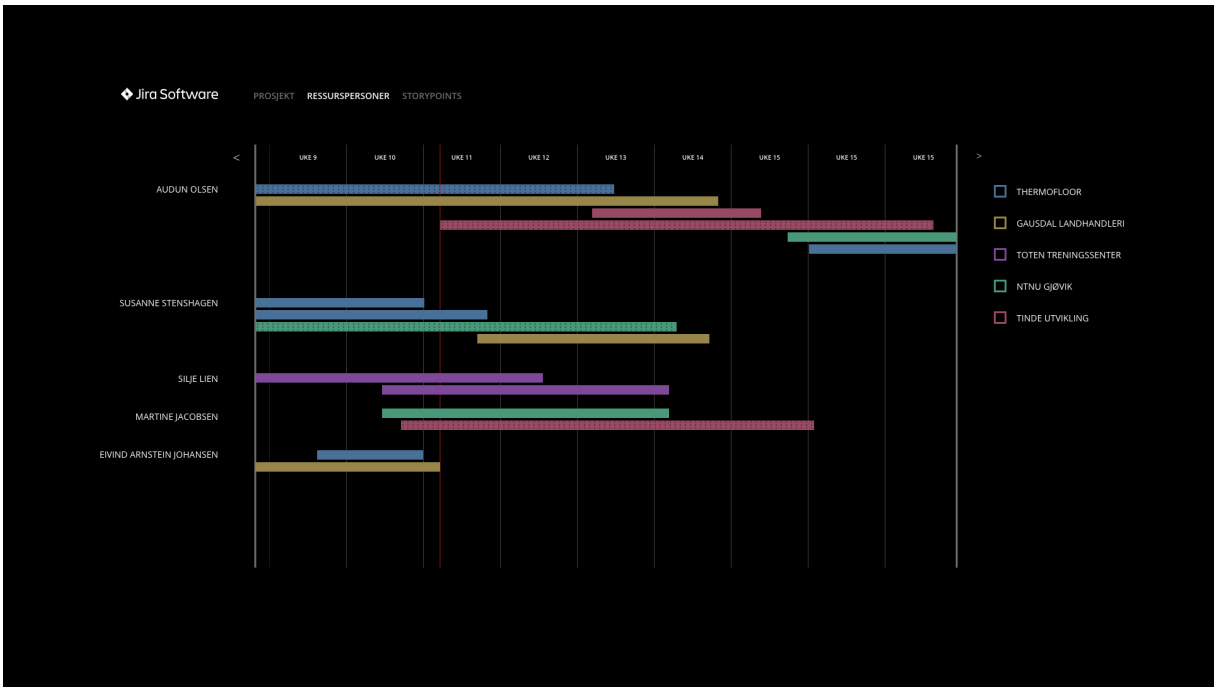
Figur 4.2 – Skjermbilde av første versjon "Prosjekt"-side.



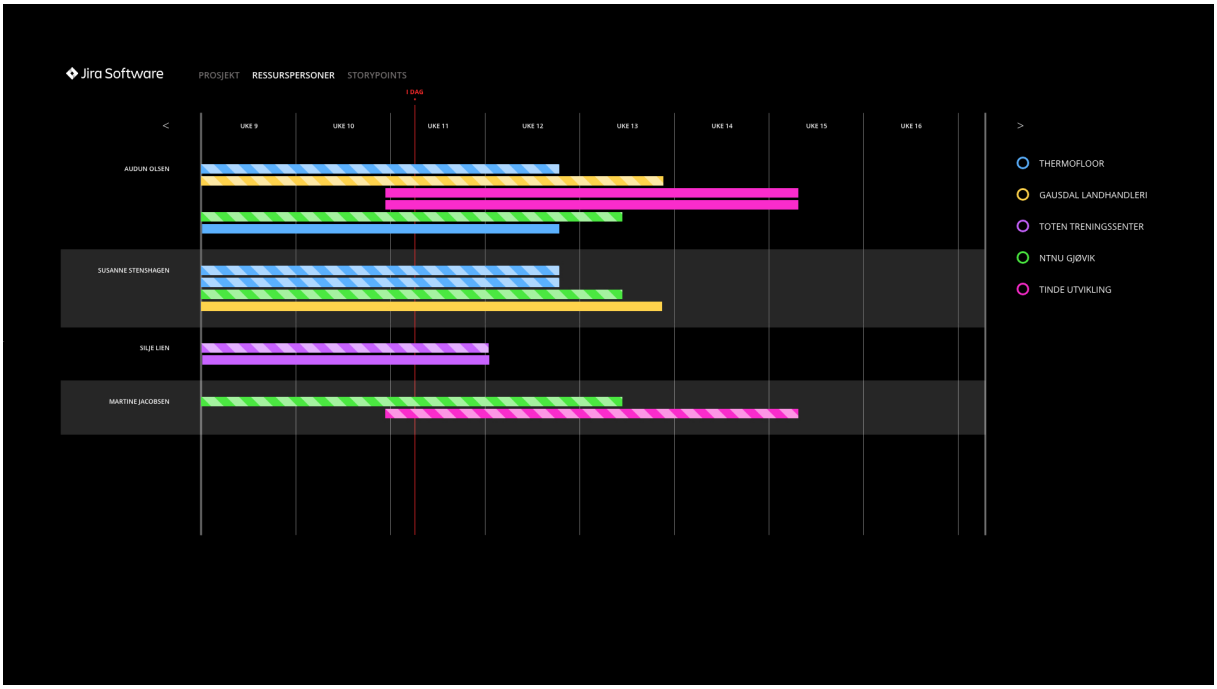
Figur 4.3 – Skjermbilde av andre versjon "Prosjekt"-side.

Figur 4.4 – Skjermbilde av ulike progresjonsbarer i andre versjon "Prosjekt"-side.

Under ressurspersoner endres avkrysningsbokser til radioknapper. Årsaken til at disse endres til radioknapper er for å følge sirkel-designet fra prosjektsiden, og på denne måten sikre et helhetlig uttrykk i alle undersidene. Det legges også på en bakgrunnsfarge horisontalt for å separere de ulike personene vertikalt.

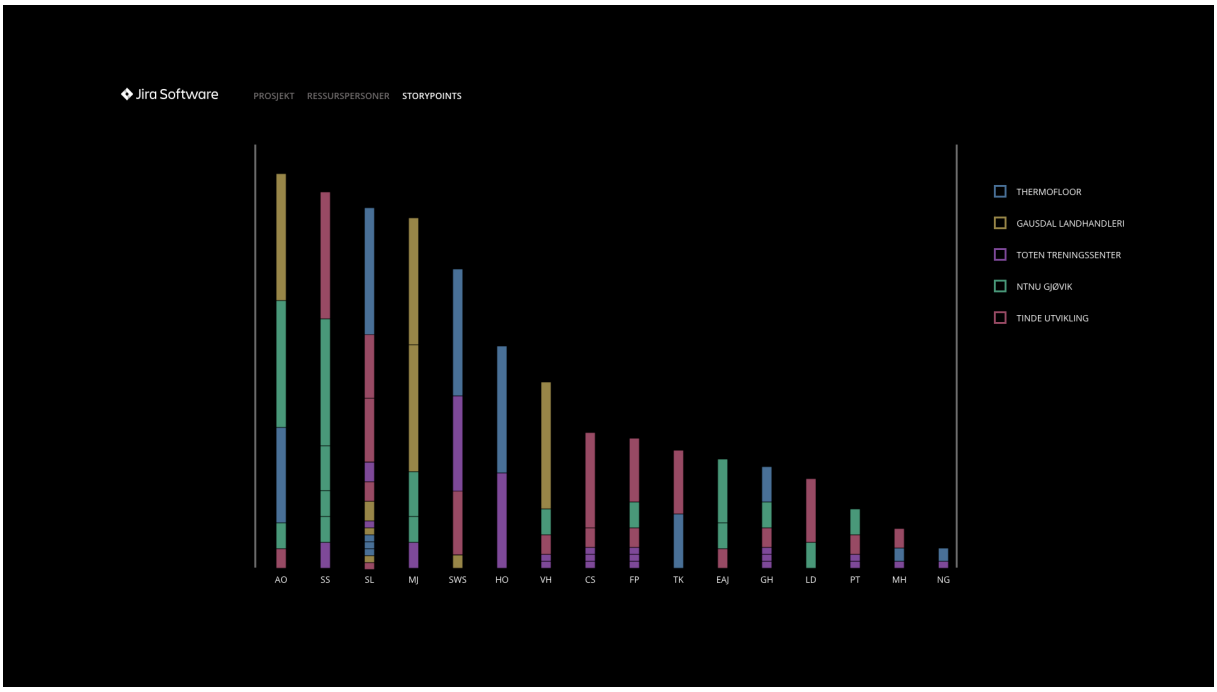


Figur 4.5 – Skjermbilde av første versjon "Ressursperson".

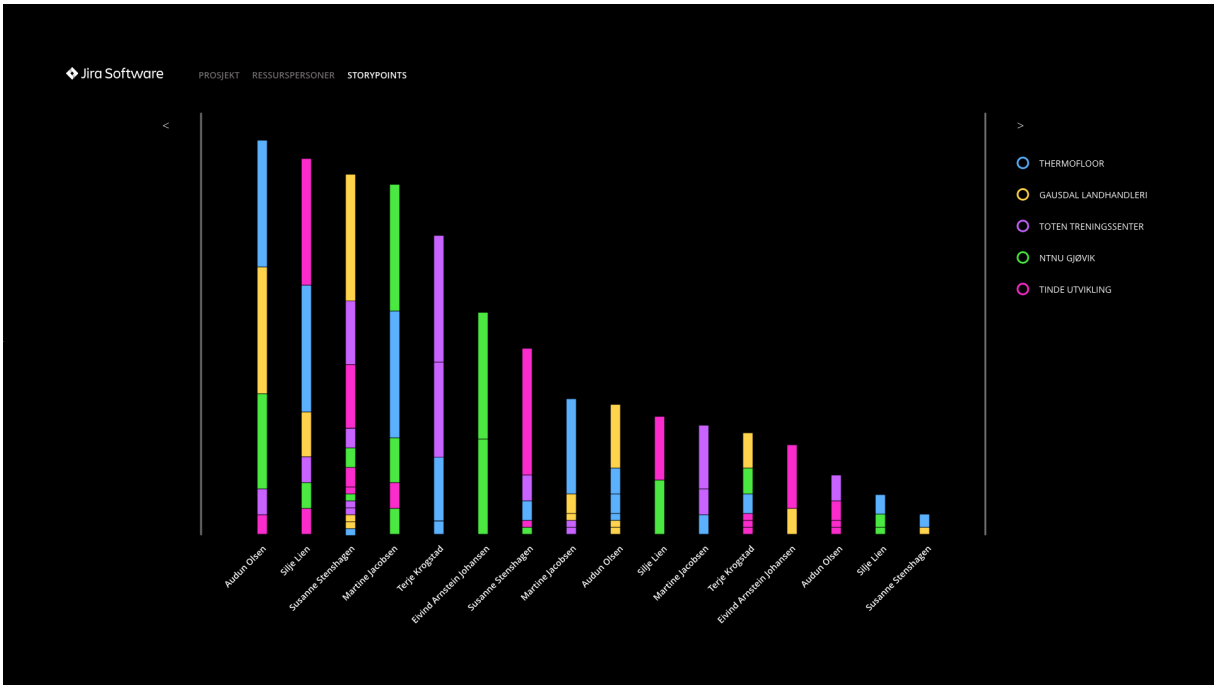


Figur 4.6 – Skjermbilde av andre versjon "Ressursperson".

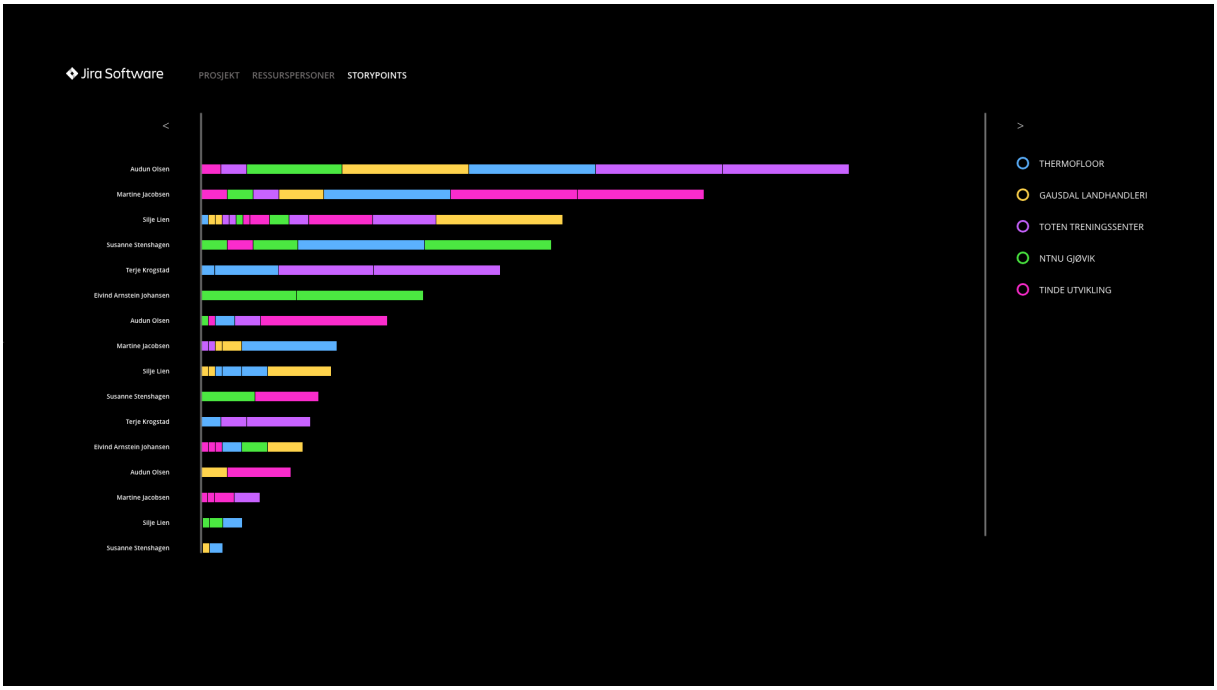
Deretter velger vi å utarbeide to ulike versjoner for storypoints, for å teste en vertikal og horisontal versjon. Her endres også avkrysningsbokser til radioknapper. Initialene i en vertikal versjon endres til fulle navn som skråstilles. Dette gjøres for å bedre leseligheten og unngå tilfeller der flere ansatte har like initialer.



Figur 4.7 – Skjermbilde av første versjon "Storypoints".



Figur 4.8 – Skjermbilde av andre versjon "Storypoints" vertikalt.



Figur 4.9 – Skjermbilde av andre versjon "Storypoints" horisontalt.

4.3.2 Andre brukstest

Den andre brukstesten gjennomføres på to testpersoner som er kjent med verktøyet Jira fra før, og som bruker det aktivt i sine prosjekter. Testpersonene får åtte scenarier, der ingen tilleggsinformasjon eller forklaringer gis i forkant. Årsaken til dette er for å få oppriktige tilbakemeldinger uten å lede frem til ønskelige svar fra vår side. Dette gjør også at vi kan registrere hvordan brukeren opplever løsningen basert på førsteinntrykk.

Scenarier

1. Du ønsker et generelt overblikk over dine arbeidere og deres belastning. Naviger til visningen best utrustet til dette.
2. Med utgangspunkt i storypoints-visningen ser du at *Audun Olsen* ser ut til å ha størst total av storypoints. Du ønsker ytterligere informasjon om Audun sine oppgaver.
3. Du er prosjektleder for *Gausdal Landhandleri* og ønsker kun å se oppgaver for dette prosjektet.
4. Du er prosjektleder for *Gausdal Landhandleri* og ønsker å se arbeidere sin storypoints-total for kun Gausdal Landhandleri.
5. Du ønsker å se storypoints-totalen for enhver arbeider og alle pågående prosjekter, men du ønsker å få en oversikt over "velocity"/fart aspektet på prosjektene.
6. Du vil få en oversikt over alle prosjekter og deres respektive sprinter.
7. Se på sprintene å forsøk å få en formening om hvordan det går med de ulike sprintene.
8. Gjør deg en formening om hvilke prosjekter som går godt og hvilke som går mindre godt.

Tilbakemeldinger og observasjoner under brukstest

Første tilbakemelding fra en testperson er et ønske om mer forklarende menynavn. Under andre scenario, kom det frem ønske om å se belastning på ressurspersoner tre uker frem i tid. For tredje scenario mener testpersonene at det ikke er tydelig nok hvilke sprinter som er overbelastet. Forslag som sterkere fargegradering eller et ekstra symbol foreslås.

En testperson forstår ikke fargegraderingen på de ulike prosjektene ved første øyekast. Det kommer forslag om en tooltip som inneholder ID til prosjekt og sprint, samt tittel på sprinten. Dette begrunnes i et ønske om mer informasjon om hver sprint, samt hver enkel oppgave i en sprint. Ukesperspektivet bør endres fra åtte uker til seks uker, ettersom de sjelden har mulighet til å planlegge så langt frem i tid.

I storypoints kommer det frem at en horisontal visning vil være mest optimalt for å tilpasse visningen til mindre enheter. Her blir det også påpekt at testpersonene synes det er nyttig med filtrering etter de ulike prosjektene. Testpersonene vil gjerne sette tidsintervallet selv under fjerde scenario, der 14 dager frem i tid er standardvisning.

Testpersonene poengterer at endringer kan oppstå etter hver fullførte sprint, så fremtidsaspektet kan være vanskelig å forutse. I tillegg settes kun start- og sluttdato for planlagte sprinter, ikke storypoints-totalen. En annen ting som testpersonene understreker er at oppgavers deadline følger sprintens start- og sluttidspunkt.

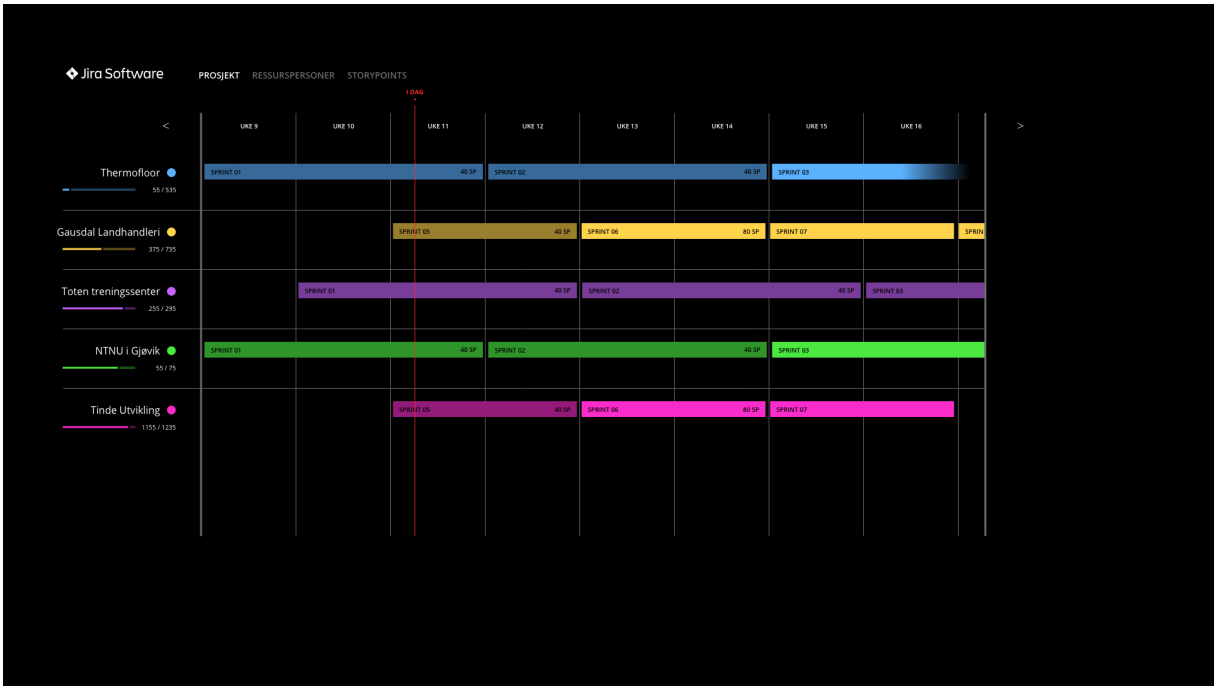
Evaluering etter brukstest

Vi er enige i hva testpersonene påpekte angående endring av menynavn, da navnene kan feiltolkes og er lite identifiserende om hva siden inneholder. Å legge inn muligheten for å endre tidsintervall under storypoints kan øke verdien av løsningen, ettersom en kan se mer nøyaktig belastning på ressurspersonene.

Vi mener også det er verdt å prøve en horisontal versjon av denne siden, for å se om dette kan fungere bedre. Det kan bety at en ikke opplever like stort tap av data for mindre skjermer, fordi en kan se flere personer på skjermen siden disse blir lagt i en vertikal liste på venstre side.

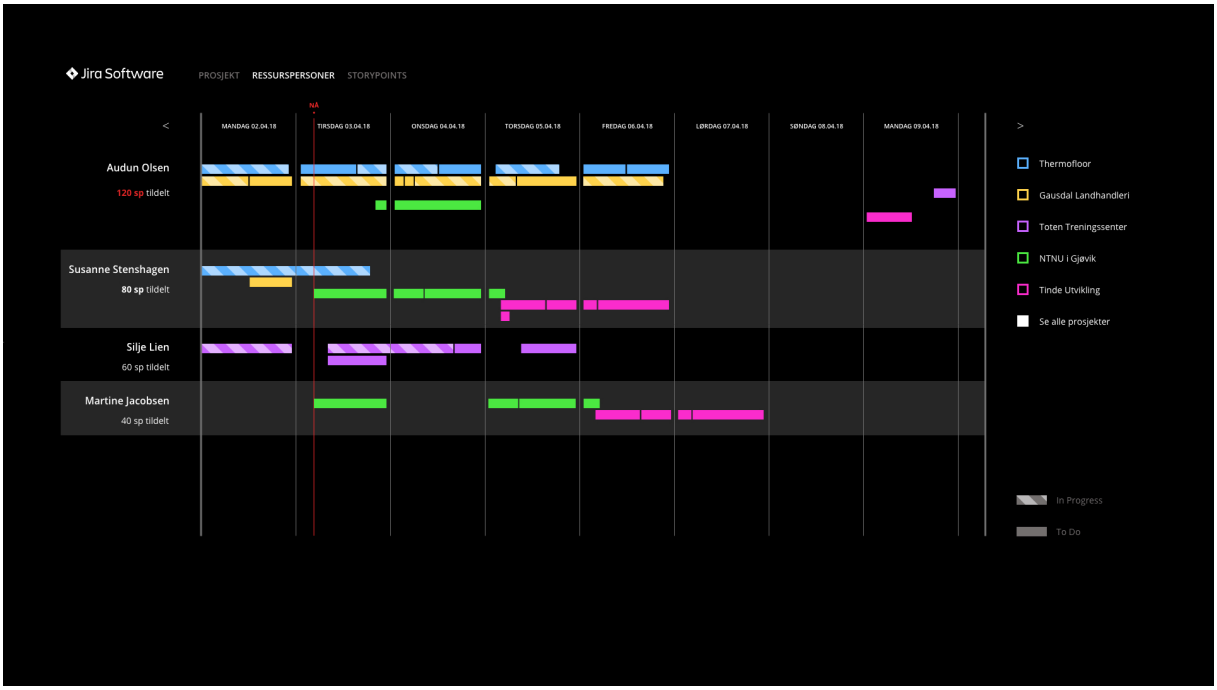
Iterering på prototype etter andre brukstest

Under prosjektsiden legger vi inn tydeligere fargegradering, der sterkere farger tydeliggjør for stor belastning. Ferdigstilte sprinter blir også fjernet, da de ikke lenger har verdi for løsningen. Sprinter som er fullført trenger en ikke vite status på, da oversikten gjelder belastning på pågående sprinter, og analyse av ferdigstilte sprinter er noe som kan hentes fra Jira. Sprinter som ikke har en sluttdato enda får en gradering til svart, da det ikke er mulig å markere et sluttpunkt uten en dato.

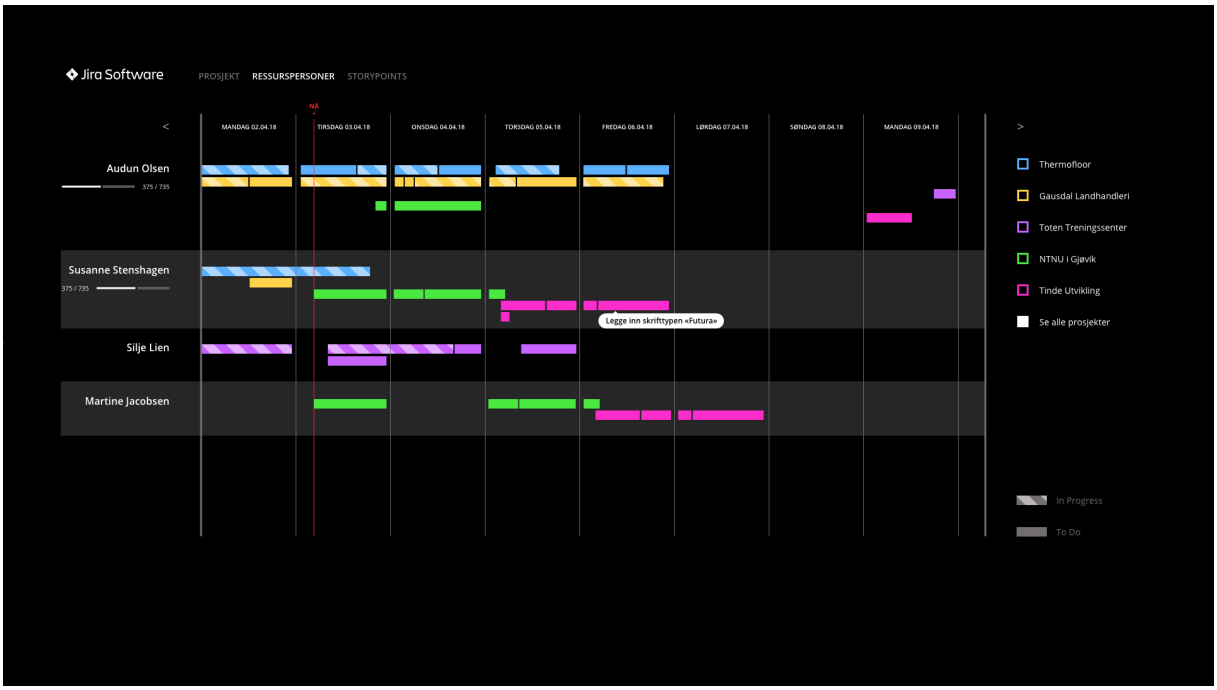


Figur 4.10 – Skjermbilde av tredje versjon "Prosjekt"-side.

I ressurspersoner prøver vi ut en versjon der oppgavene får hver sin boks. Det blir også lagt til en forklaring nederst til høyre. Denne forklaringen presiserer hva som menes med skravert og heldekkende farge på oppgavene. Ulike versjoner for status på ansatt prøves ut, men disse vurderes til å ikke være en god nok løsning. Årsaken til dette er at det ikke indikerer om det er et problem her og nå, eller om det vil være et problem fremover i tid. Tooltip legges til og testes, samtidig endres filtreringen fra radioknapper til avkrysningsbokser igjen, ettersom det kan være større verdi i muligheten til å se flere prosjekter samtidig. Avkrysningsbokser samsvarer med denne konvensjonen.

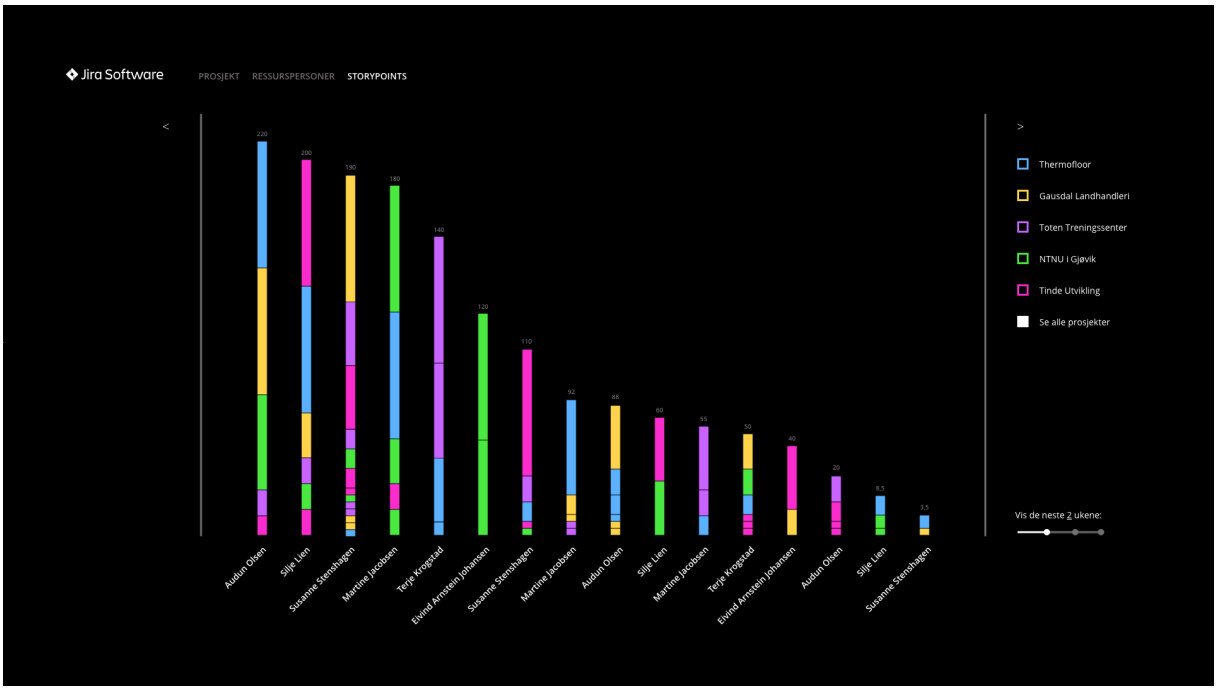


Figur 4.11 – Skjermbilde av tredje versjon "Ressurspersoner"-side.

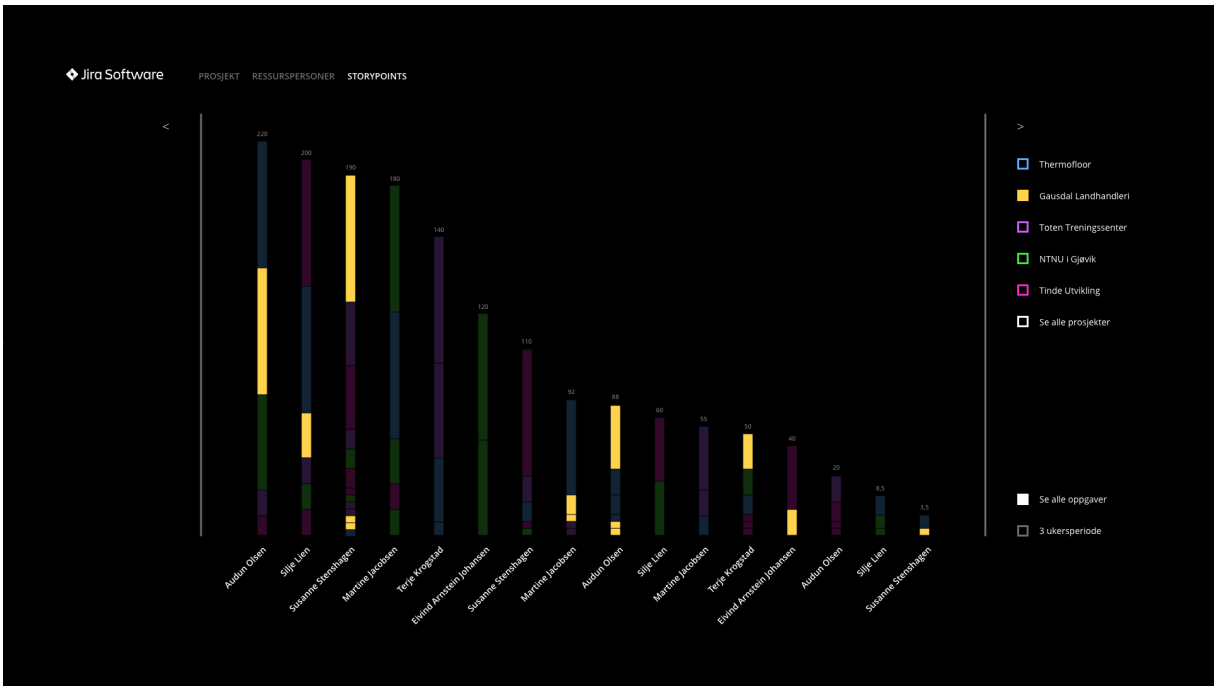


Figur 4.12 – Skjermbilde av tredje versjon "Ressurspersoner" med tooltip og progresjonsbar for ansatt.

I storypoints legges det inn filtrering på tidsintervall, der vi prøver ut to ulike versjoner. Dersom et prosjekt hukes av, blir de resterende mindre synlige. Slik kan en fortsatt se om en ansatt er overbelastet, uavhengig om en kanskje ikke har så mange oppgaver på det filtrerte prosjektet. Radioknapper blir også endret tilbake til avkrysningsbokser, av samme årsak som under ressurspersoner.



Figur 4.13 – Skjermbilde av tredje versjon "Storypoints".



Figur 4.14 – Skjermbilde av tredje versjon "Storypoints" der ett prosjekt er huket av, samtidig ble det testet en annen versjon med tidsintervall.

4.3.3 Tredje brukstest

Brukstest tre gjennomføres med oppdragsgiver og veileder, der brukstesten med oppdragsgiver gjennomføres i henhold til Sandnes sin metode, kalt en gonzotest (2011). Denne brukstesten utføres i oppdragsgivers lokaler, som er et naturlig miljø for oppdragsgiver. Det oppgis ingen scenarier, ettersom begge personene har god nok informasjon om prototypen. Iterering gjøres mellom veiledermøte og møtet med oppdragsgiver, samt etter begge møtene.

Tilbakemeldinger og observasjoner under brukstest

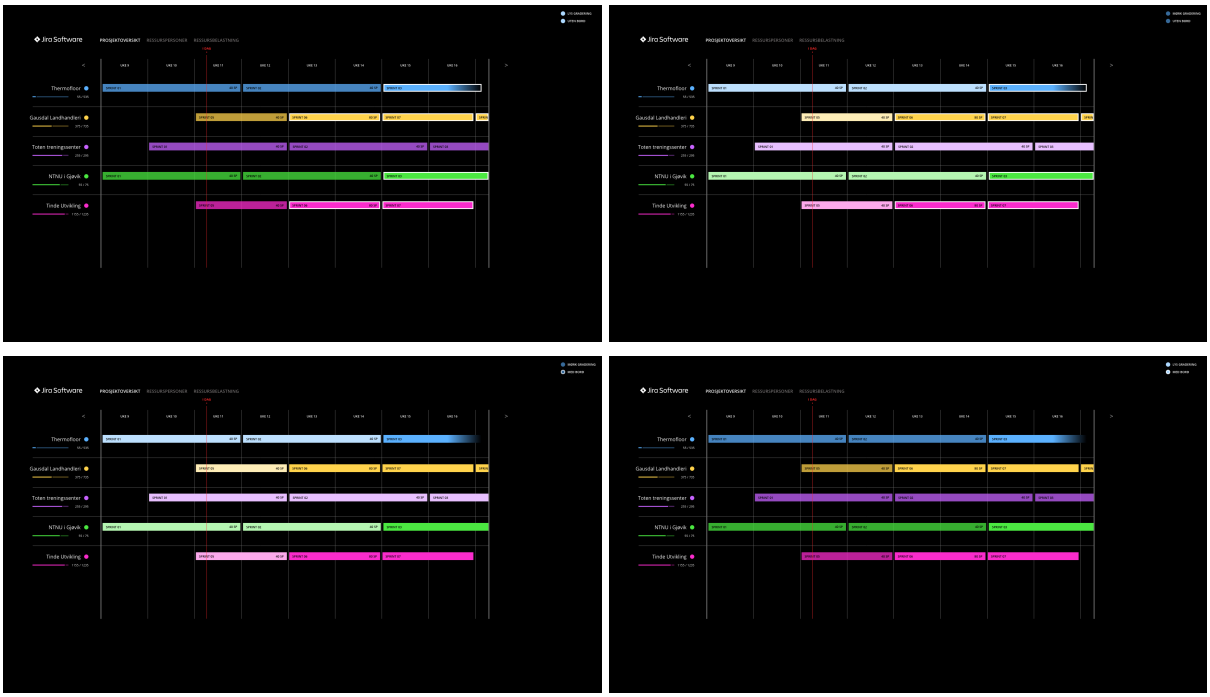
Den ene testpersonen ønsker at når det lenkes eksternt, åpnes det i et nytt vindu. Det nevnes at det bør ses på om det kan bli for mye med drop-down og tooltip på ressurspersoner. En annen idé som foreslås, er å slå sammen linjene på ressurspersoner til en linje. Linjene representere oppgaver på et prosjekt, gruppert etter status. På den måten sparer en plass. Argumentet er at to-do-linjen bør være borte etter at sprinten er borte. En kombinasjon en kan gjøre, er at dersom fem av syv oppgaver er in-progress, så kan 5/7 av linjen være skravert, mens resten har heltrukket farge. Dette gjør det også mer visuelt synlig. Et forslag fra testpersonen er å sidestille fremhevede oppgaver på ressursbelastning på venstre side. Dette kan bidra til å synliggjøre hvor store oppgaver en har i hvert prosjekt. Det foreslås også å flytte personer som ikke er deltakende nederst når en klikker på prosjektet.

Evaluering etter brukstest

I løpet av denne brukstesten kom nye og relevante ideer til mer markant tydeliggjøring av informasjon. Ideen om å slå sammen to linjer til én linje kan være veldig verdifull, da det både sparer plass, og gjør betydningsfull informasjon mer synlig. Testpersonens bekymring og eventuell irritasjon for å ha både drop-down og tooltip på ressurspersoner, mener vi må vike for at essensiell informasjon synliggjøres og er lett tilgjengelig til enhver tid. Dette kan også sikre at en slipper å åpne Jira for å se nødvendig informasjon som lett kan gjøres tilgjengelig i vår løsning. Motargumentet for flytting av personer når en klikker på prosjekter, er at det kan være forvirrende for brukeren dersom personer bytter plass. Slik bevegelse kan skape støy og forvirring, da logikken for en slik forandringer kan være vanskelig å forstå.

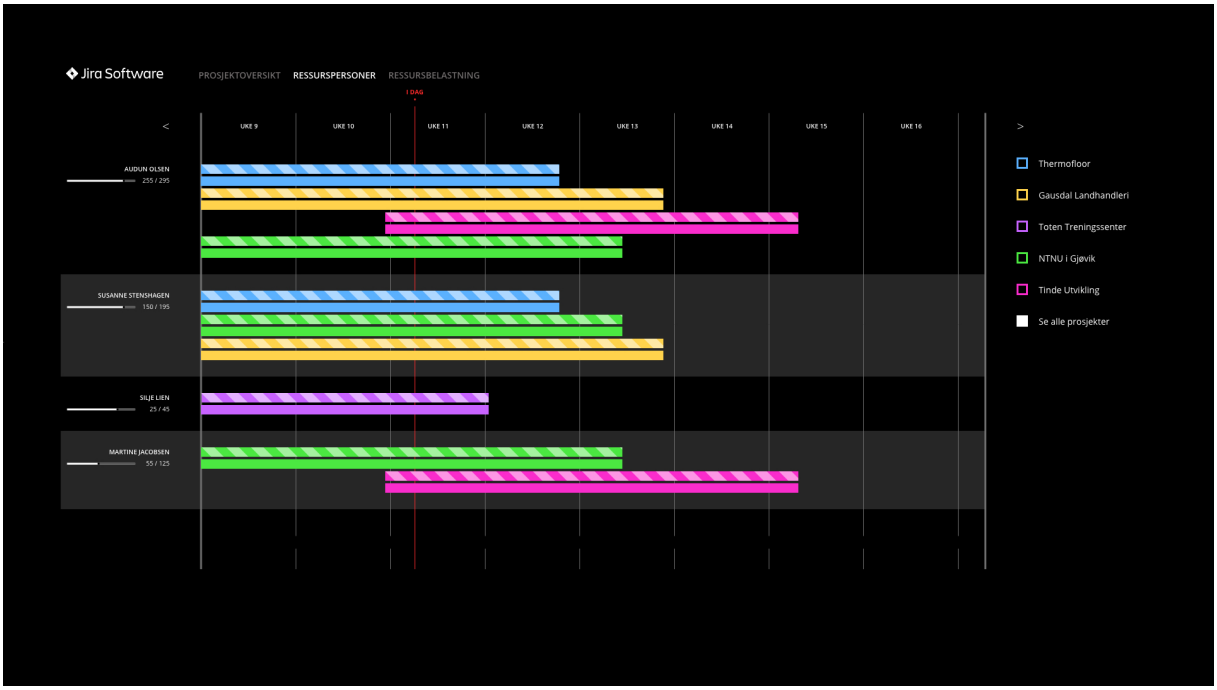
Iterering på prototype etter tredje brukstest

Det testes ut fire ulike fargenyanser på sprintene under prosjektoversikt, der noen også har stripe rundt for å vektlegge belastningen enda tydeligere. Menynavn blir endret fra prosjekt, ressurspersoner og storypoints til det vi mener er mer forklarende titler: prosjektoversikt, ressurspersoner og ressursbelastning. Etter brukstest tre gjøres prototypen fullstendig klikkbar i Adobe Xd.



Figur 4.15 – Skjermbilde av fjerde versjon "Prosjektoversikt" med fire ulike fargenyanser og med striper rundt.

Ressurspersoner får også flere endringer. For å unngå en lang liste som må scrolles gjennom dersom en bedrift har mange ansatte tildelt flere sprinter og oppgaver, velger vi å samle alle oppgaver i samme prosjekt og sprint i to progresjonsbarer – én for to-do og én for in-progress.

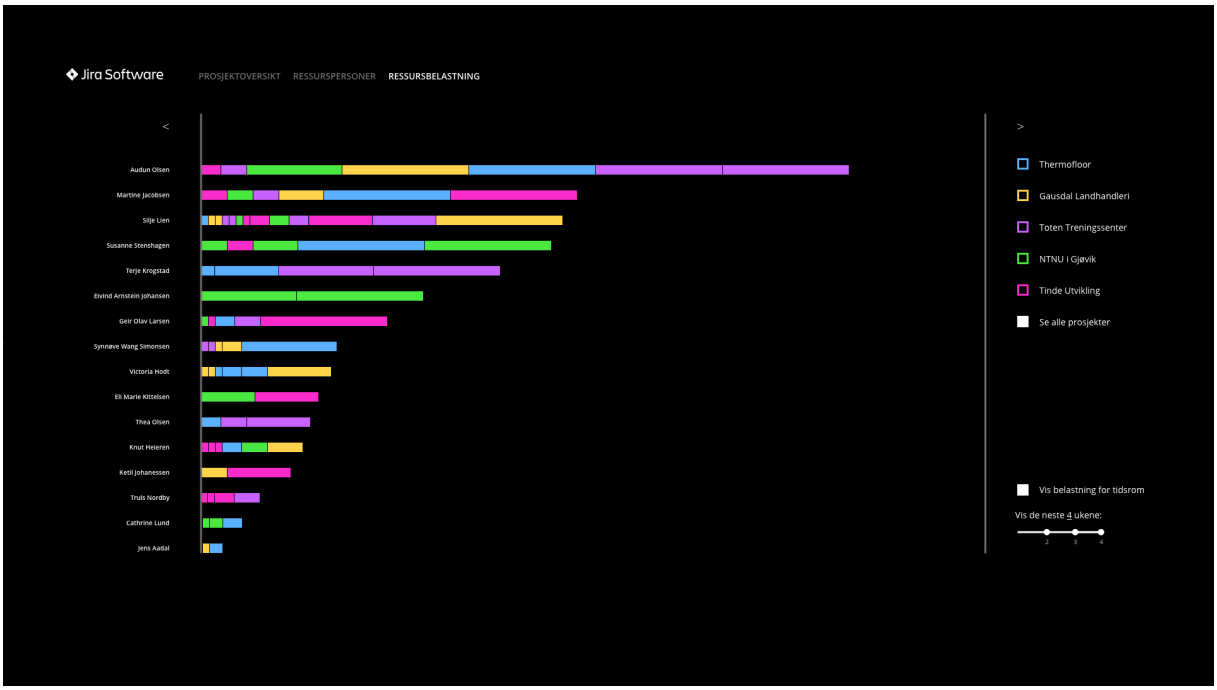


Figur 4.16 – Skjermbilde av fjerde versjon "Ressurspersoner" med kun to progesjonsbarer for hvert prosjekt.

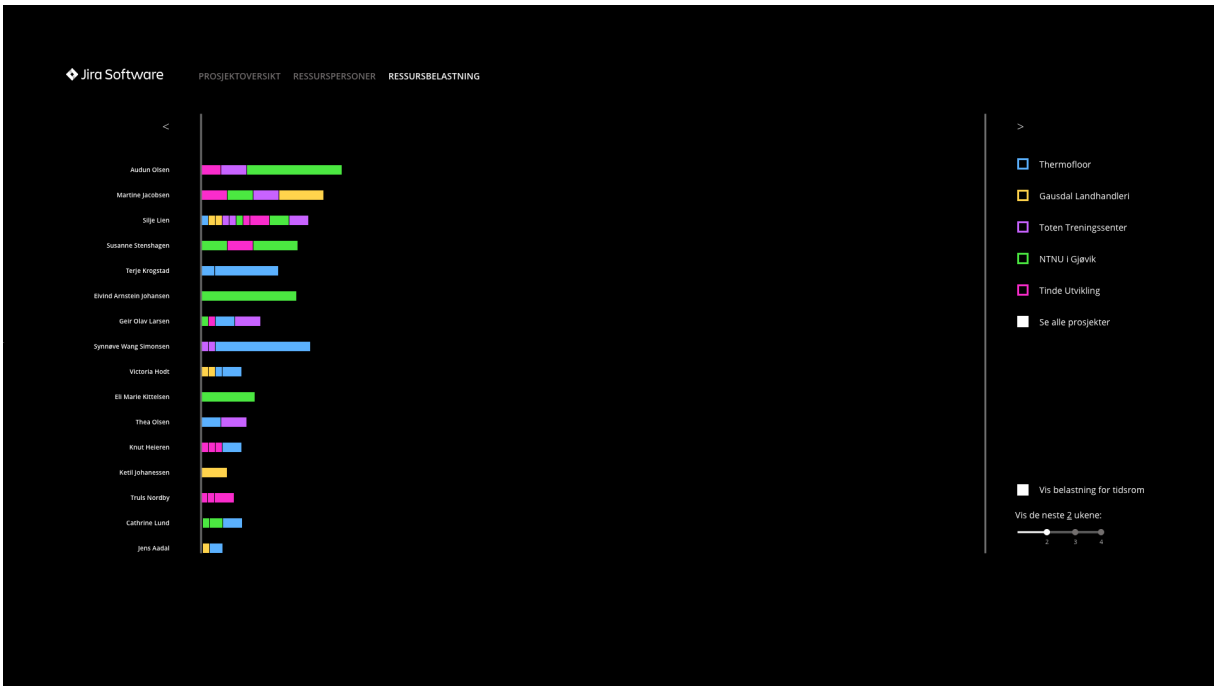
Tooltip fikk en designendring og ble plassert over progresjonsbaren. For å synliggjøre informasjon på en effektiv og plassbesparende måte, har vi med tooltip. Det legges også til en drop-down når en klikker på sprintene for å se den valgte ansattes oppgaver i aktuell sprint. På denne måten kan en få inngående informasjon ved kun ett klikk.

Figur 4.17 – Skjermbilde av fjerde versjon "Ressurspersoner" med tooltip og liste med oppgaver.

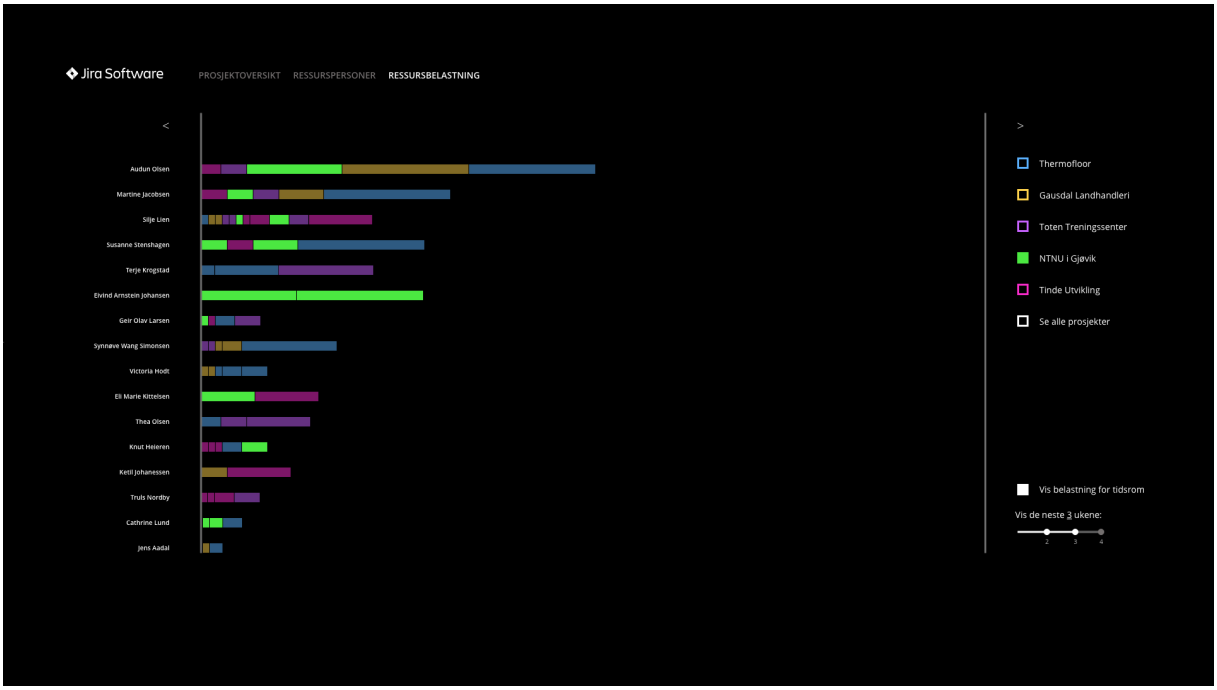
I ressursbelastning, den siste siden, tas en avgjørelse om å beholde kun den horisontale versjonen, for å unngå tap av data.



Figur 4.18 – Skjermbilde av fjerde versjon "Ressursbelastning" horisontalt.



Figur 4.19 – Skjermbilde av fjerde versjon "Ressursbelastning" der tidsintervall for to uker er satt.



Figur 4.20 – Skjermbilde av fjerde versjon "Ressursbelastning" der tidsintervall for tre uker er satt, samt grønt prosjekt er huket av.

4.3.4 Fjerde brukstest

Den siste brukstesten gjøres på en testperson som er kjent med verktøyet Jira fra før, og er ansatt i bedriften. Testpersonen har ikke rollen som prosjektleder eller Scrum Master, men er likevel en aktuell kandidat ettersom personen bruker Jira i prosjektarbeid. Det kan også være nyttig å få tilbakemeldinger av en person som ikke har like god kjennskap til oppgavebeskrivelse. Dette gjør at vi i siste brukstest kan avdekke ny informasjon som kan være viktig for sluttresultatet. Til denne brukstesten gis ingen spesifikke scenarier, da hensikten er å se brukerens generelle betraktninger og observasjoner, samt kartlegge om all viktig informasjon kommer frem på en tydelig måte.

Tilbakemeldinger og observasjoner under brukstest

I prosjektoversikt liker testpersonen at prosjektene er skilt med farger og opplever at innholdet tydelig er delt inn i rader. De visuelle indikasjonene for å avdekke sprintbelastning oppfattes ikke umiddelbart, men testpersonen forstår likevel at det var et varsel om noe som ikke er positivt. Testpersonen har et ønske om forklaringer på hvorfor noen sprinter har en sterkere farge enn andre.

Under ressurspersoner gir det ikke mening for testpersonen at to-do oppgavene har en heldekkende farge. Testpersonen oppfatter hel-fargen som en indikasjon på oppgaver som er fullførte. Den stripete linjen for in-progress fungerer godt, ettersom den følger konvensjonen om at noe er underveis. Testpersonen er usikker på hvor rotete det kan bli dersom det er mange prosjekter pågående, og begrunner dette med fargebruken.

Testpersonen ser tydelig hvilke personer som er overbelastet i ressursbelastning. Tidsrom som element bør være mer synlig fra start, fordi det påpekes en usikkerhet om

tidsrommet er for i år, hittil eller en annen intervall. Testpersonen foreslår å endre til radioknapper. Et ønske om mer beskrivende tekst om hva siden inneholder, som en tittel øverst, mener testpersonen kan være av verdi for brukeren.

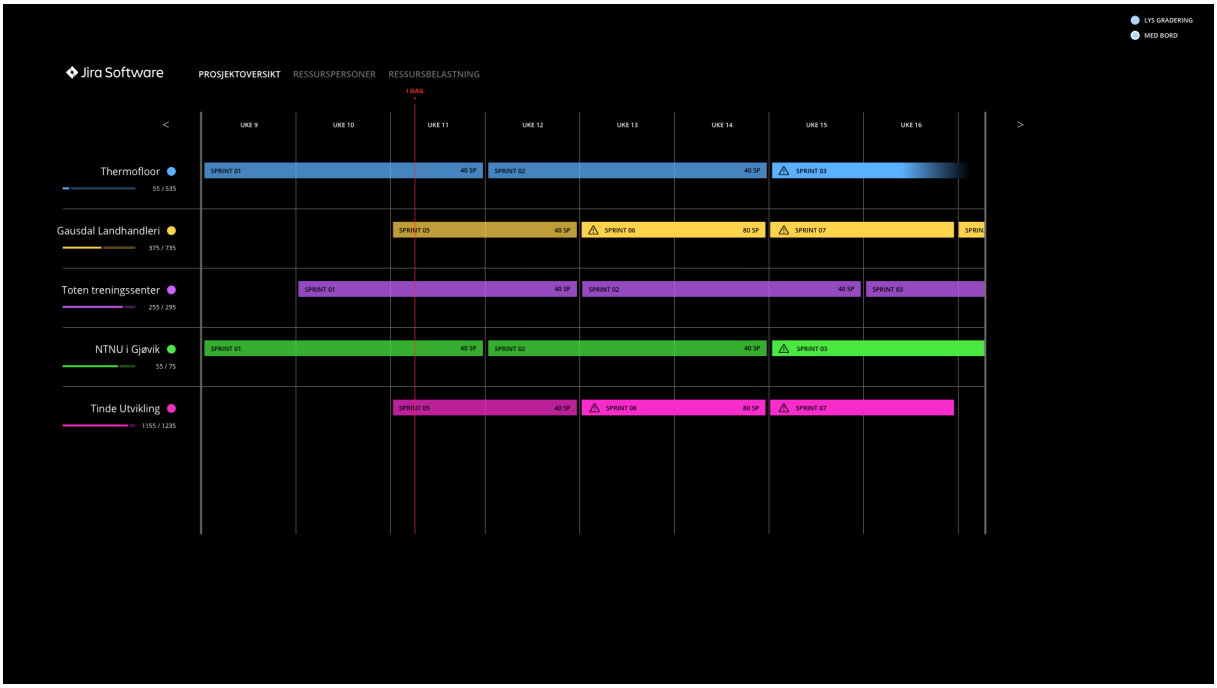
Andre ting som kommer fram er at testpersonen liker den mørke bakgrunnen, fordi det tydeliggjør fargene. Forslag om å sette egne farger til prosjektene, og drag-and-drop på oppgaver blir også nevnt.

Evaluering etter brukstest

Etter gjennomføring av siste brukstest, så vi verdien av å tydeliggjøre tidsintervaller på siden ressursbelastning. Testpersonen foreslo radioknapper, samtidig tenker vi det kan være logisk med en slags tidslinje, for å illustrere at det er for å se fremover i tid.

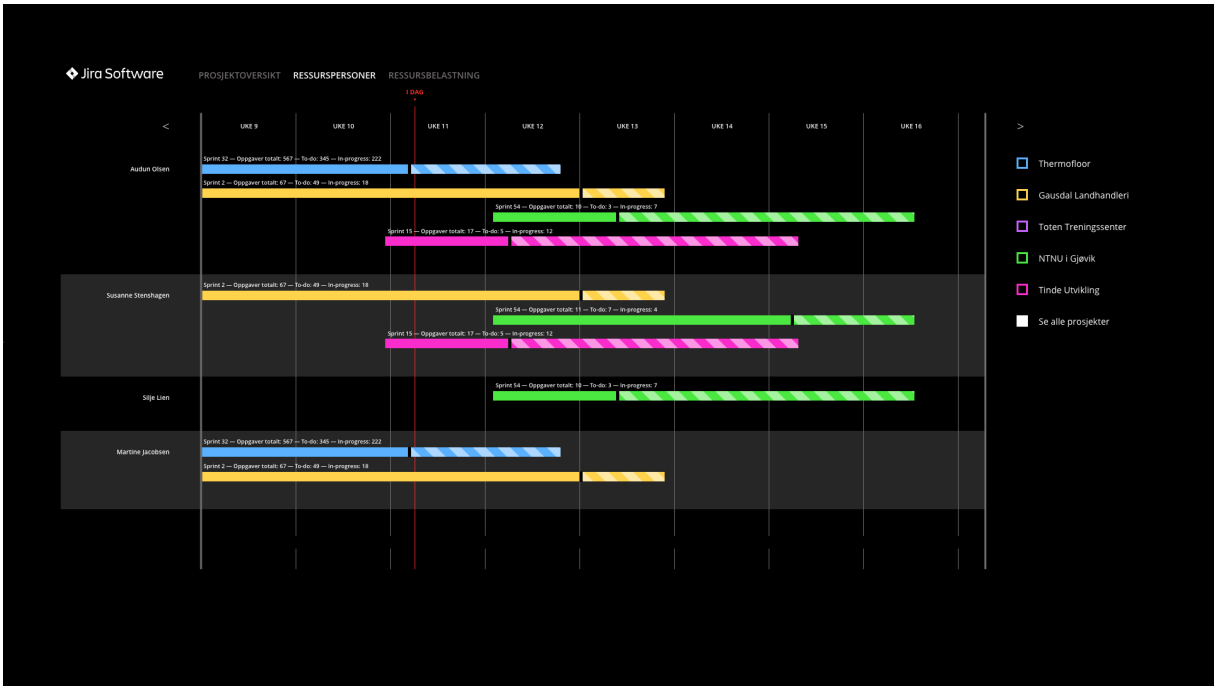
Iterering på prototype etter andre brukstest

Etter brukstest fire endres flere ting. Det legges til symboler i prosjektoversikt for raskere gjenkjenning (Cooper *et al.*, 2014) og tydeliggjøre overbelastning – i tillegg til den sterke fargen. Vi velger å holde symbolbruk til et minimum for å ikke skape støy i brukergrensesnittet (Sandnes, 2011).



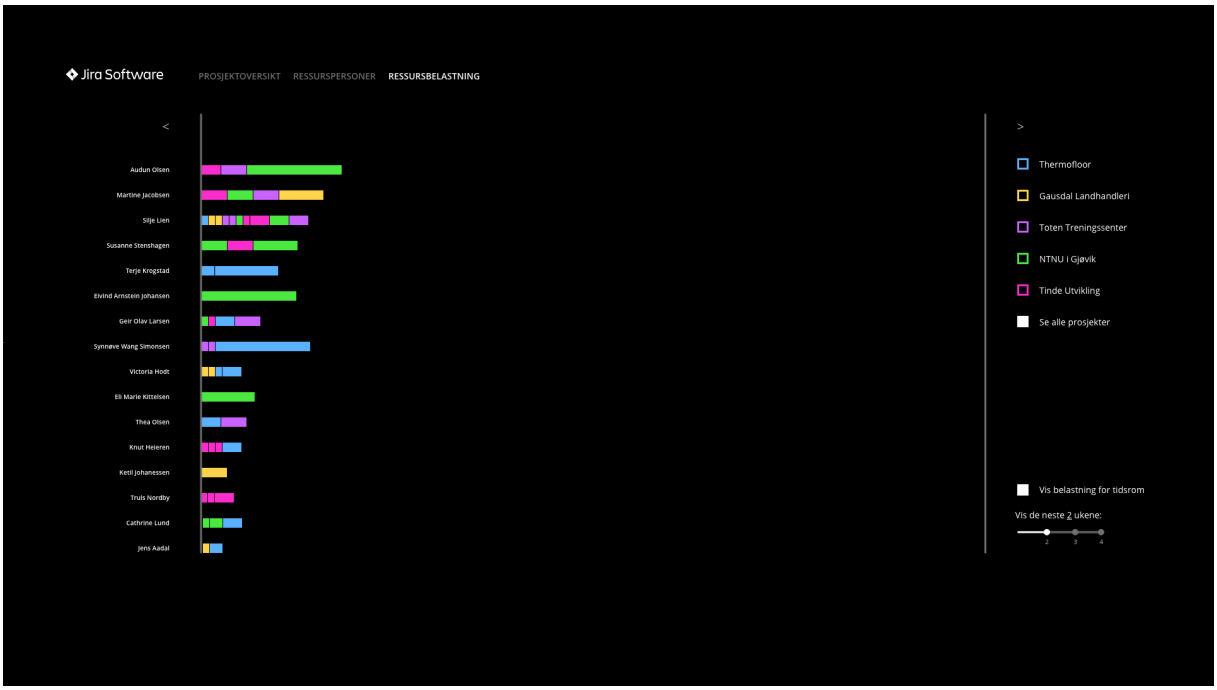
Figur 4.21 – Skjermbilde av femte versjon "Prosjektoversikt" med symboler for overbelastning.

I ressurspersoner får hvert prosjekt kun én linje, for å spare plass dersom en har mange ansatte i bedriften. To-do linjen kommer først, for å unngå konvensjonen om at progresjonsbaren for in-progress skal gå oppover. Samtidig som det indikerer at det første er det som er viktigst å se. Det ble lagt til tittel over sprintene.



Figur 4.22 – Skjermbilde av femte versjon "Ressurspersoner" med en progresjonsbar for hvert prosjekt, der to-do og in-progress er samlet til en enkel linje.

I ressursbelastning velger vi å beholde løsningen som ble presentert for testpersonen. Det settes et utgangspunkt i at itereringen av denne siden kan videreføres i implementeringsfasen, ettersom den fasen kan synliggjøre endringer som ikke kartlegges under brukstesting.



Figur 4.23 – Skjermbilde av femte versjon "Ressursbelastning" der to uker er krysset av.

4.4 Oppsummering

I dette kapitlet har vi vist hvor stor vekt vi har lagt på prototyping, brukstesting og iterasjon når det kommer til valgene vi har tatt underveis i prosessen. Det har gitt oss et grundig utgangspunkt for implementering av en kodet prototype. Et så detaljert grafisk bilde av løsningen mener vi vil bidra til å sette fokus på det tekniske, der den visuelle implementeringen vil gjøres mer effektivt. Videre iterering av prototype gjøres i implementeringsfasen, der vi gjør flere tekniske implementeringer for å få den endelige løsningen. Implementering av koding og script vil presenteres og forklares i neste kapittel.

5. Implementering

5.1 Introduksjon

Som nevnt avslutningsvis i foregående kapittel vil vi videre forklare gruppens hendelsesforløp ved implementering av løsningen. Det spesifiseres også hvorfor ulike teknologier brukes. Konsepter og teori for anvendt teknologi er presentert i *kapittel 2 – Teori, metoder og verktøy* og *kapittel 3 – Analyser* er lagt til grunn for hvorfor vi velger å anvende disse teknologiene. Fullstendig logg av progresjon i kodebasen finnes under vår Github katalog: www.github.com/audunolsen/simasu/commits.

5.2 Utvikling

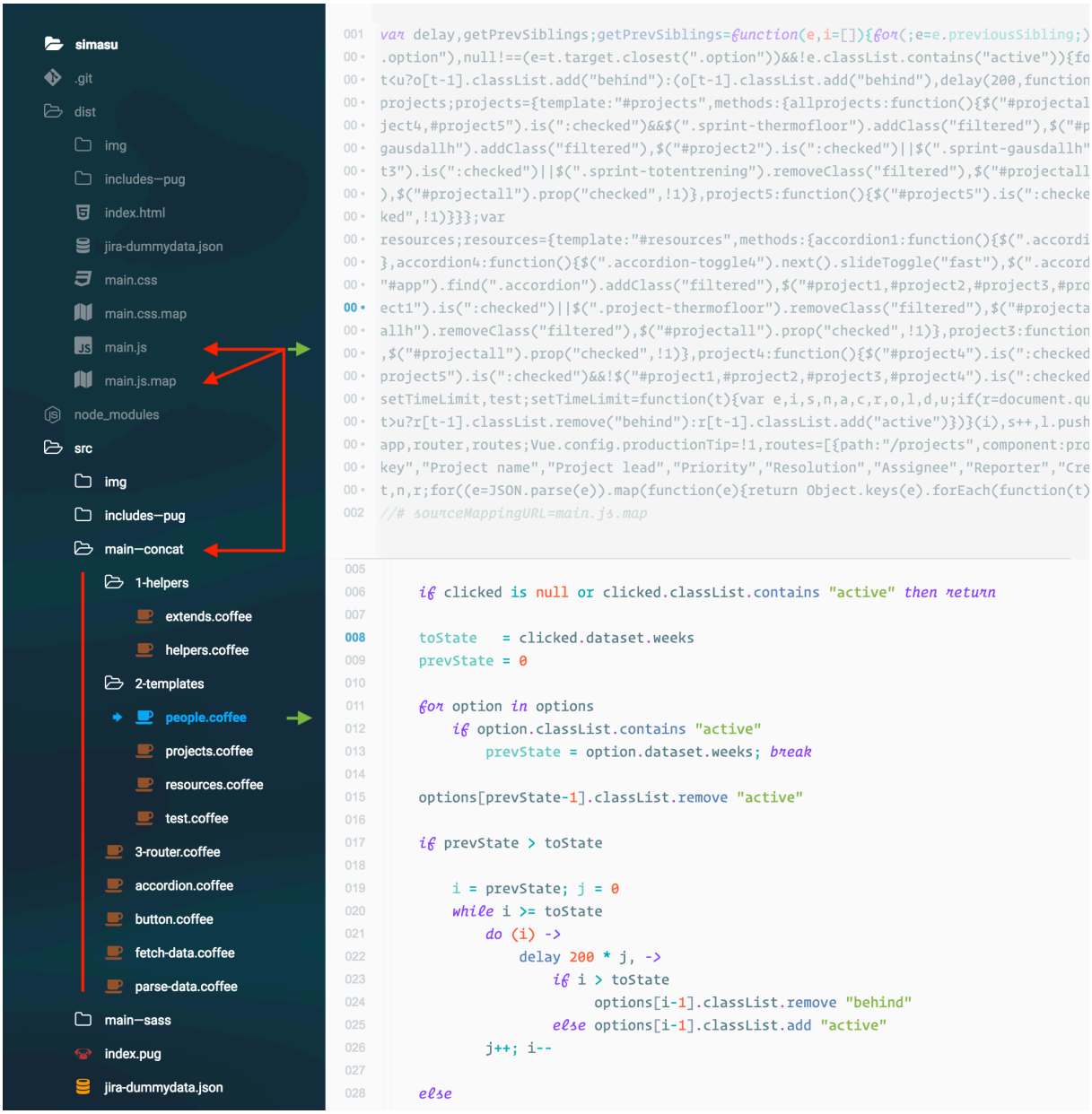
5.2.1 Utviklermiljø

Vi er opptatt av å sikre et godt fundamentet for videre utvikling. Derfor etablerer vi forholdsregler for å sikre et godt teknisk samarbeid mellom gruppemedlemmene. Dette antas å resultere i et prosjekt som er innbydende å jobbe videre med. For å manifestere slike forholdsregler i et teknisk verktøy, kan en bruke et utviklermiljø. Node.js og NPM implementeres for å etablere utviklermiljøet, ettersom bruk av Node.js har vært et ønske fra oppdragsgiver fra start.

CoffeeScript-, Node-, Sass- og Pug-modulene hentes inn fra start fra NPM sitt register. Slik kan vi skrive egne Node-skript som bruker deres respektive kompilatorer for å kompilere eksempelvis Pug-filer til HTML. Å implementere en whitespace sensitive syntaks for HTML, CSS og JavaScript – altså innhold, presentasjon og logikk – er vårt fundament for å etablere et miljø som fremmer ergonomisk og lesbar kode, samt kode som minsker gapet mellom programkode og pseudokode.

For å sikre et skille mellom programkode som skal være optimalisert for sluttbruker, og kode som er optimalisert for arbeid, definerer vi en mappestruktur som legger rette for dette. Katalogen for kildekode navngir vi *src*, kort for source, og *dist* for produksjonskode-katalogen, kort for distribution. Deretter skriver vi egne Node-skript som sørger for at kildekode som kompileres av preprosesserings-språkene sine respektive kompilatorer blir komprimert og distribuert videre til dist. For å jobbe i en semantisk oppdeling av filer – uten å forårsake mange *http-requests* for sluttbrukeren – lager vi skript som kan slå sammen selvstendige skript-filer med særegne formål til én større, komprimert fil. Ved å bruke en amerikansk tankestrek i katalognavnet forteller vi utvikler-miljøet hvilke kataloger den skal ta til betraktning for konkatenering. En katalog av CoffeeScript-filer kalt main—concat vil for eksempel bli til én JavaScript-fil kalt main.js. Alt bak

tankestreken blir erstattet med den aktuelle filtypen. Utviklermiljøet søker alle kataloger rekursivt i src for å finne privilegerte kataloger, altså de katalogene som inneholder en amerikansk tankestrek, for å gi de særegen behandling. Vi implementerer et eget skript for håndtering av alle filer og kataloger som ikke er privilegerte, det vil si filer som ikke er av filtypen .pug, .sass eller .coffee, samt kataloger som ikke inneholder en amerikansk tankestrek. Skriptet sin rolle er kun å rekursivt kopiere disse filene fra src til dist.



Figur 5.1 – Skjermbilde av konkateneringsmappe og komprimert fil i prosjektet.

For å kvalitetssikre koden som skrives, samt innføre konsekvente sedvaner for alle kollaboratører, er neste steg å inkorporere stilsjekkere for de respektive preprosesseringsspråkene. Her benytter vi oss av pug-lint, sass-lint og coffee-lint. Vi velger å skrive skript som lytter etter filendringer, og deretter sjekker filene for uoverensstemmelser. Dette gjør det enkelt å rette opp i koden fortløpende under utviklingen. Vi skriver også en egen klasse kalt *log* som brukes til å formattere output-en til stilsjekkerne, slik at advarsler og feil blir presentert tydelig og klart.

```
── SIMASU ▸ listening for changes… ──────────────────────────────

    11:43 ▸ file "index.pug" has been changed

···· WARNING ▸ Pug lint ····· ····· ····· ····· ····· ····· ····· ·····

○ src/index.pug

    4|         link( rel="stylesheet", href="main.css" )
    5|         meta( charset="utf-8" )
  > 6|         link( href="https://fonts.googleapis.com/css?family=Open+Sans:30
0,400,600", rel="stylesheet" )
    7|
    8|     body
    9|

Line length exceeds the maximum of 100 chars

····· ····· ····· ····· ····· ····· ····· ····· ····· ····· ····· ·····

    SUCCESS ▸ Pug
    ·····
    ✓ Wrote file "index.html"
```

Figur 5.2 – Skjermbilde av en advarsel fra pug-lint stilsjekkeren.

Videre implementerer vi en modul kalt *browser-sync* for å starte en server under arbeidsmaskinens localhost, slik at web-applikasjonen kan testes på andre enheter om nødvendig. Browser-sync implementeres også fordi modulen tillater automatisk injeksjon av endringer i kildekoden. Det vil si at en utvikler ikke behøver å laste inn applikasjonen manuelt for å få de siste endringene.

For å forenkle feilsøking under utvikling implementerer vi sourcemaps for å kartlegge innhold i kildekode til innholdet i produksjonskode. Dette tillater oss å se hvor runtime skriptfeil oppstår i våre CoffeeScript-filer, til tross for at CoffeeScript bruker en syntaks nettlesere ikke forstår. Med sourcemaps kan vi også i nettlesers konsoll se hvor i Sass-strukturen CSS -stiler kommer fra.

```
Console opened at 11:46:42 AM
▶ TypeError: null is not an object (evaluating 'document.querySelector("#jira-data").innerHTML=`${JSON.stringify(e[0])}`')   fetch-data.coffee:56
>
54    # Print to page TEST
55    document.querySelector "#jira-data"
56        .innerHTML = "#{JSON.stringify data[0]}"
57
```

Figur 5.3 – Skjermbilde av runtime feil som peker mot CoffeeScript kode.

5.2.2 Back-end

I vår dialog med oppdragsgiver ble vi tidlig informert om at kobling mot Jira har lav prioritet og at fokus i stedet bør være på reell test-data som de kan gi oss. Gjennomføring av en Jira-kobling er som tidligere nevnt et langsiktig mål, og ettersom denne koblingen representerer vår løsning sin server-funksjonalitet, blir ikke back-end et fokusområde i vårt prosjekt. For å simulere server-funksjonalitet for å hente data ble henting av test-data via en XMLHttpRequest et fokusområde.

Test-dataen vi får er i form av et øyeblikksbilde som viser data knyttet til alle oppgaver under deres Jira side i form av en *CSV-datafil*. CSV er et akronym for comma separated values. En CSV-fil viser data i en tabell der kolonner skilles med komma, og rader skilles med linjeskift. JavaScript har ingen funksjonalitet for å direkte interagere med CSV-formatet, og man må i så fall skrive egne funksjoner for å transformere CSV-data før den kan brukes. Ettersom vi arbeider i CoffeeScript – en variant av JavaScript – oppleves det som en utfordring å operere med CSV-filer, da en ikke kan bruke punktnotasjon for å aksessere data. Med JavaScript kan en enkelt hente data fra spesifikke nøkler som lagres i objekter ved å bruke punktnotasjon. Vi ønsker derfor å bruke *JSON* som filformat for å lagre test-dataen, da JSON er et format som lagrer data i JavaScript-objekter. Derfor velger vi å bruke NPM-modulen kalt CSVtoJSON for å transformere test-dataen fra et CSV-format, til et JSON-format, hvor den resulterende JSON-filen er omtrent 70 000 linjer lang. Filen brukes videre for lasting av data gjennom en egendefinert funksjon kalt *loadJSON* som bruker en instans av XMLHttpRequest-klassen for å åpne filen med data fra Jira. Deretter kalles en *callback*-funksjon som sender med Jira-dataen som et argument. Jira-dataen er nå tilgjengelig som en array av objekter en enkelt kan manipulere i videre skripting. Jira-dataen inneholder flere datanøkler enn det som er relevant for vår løsning, og oppdragsgiver har gitt en oversikt over de datanøklene vi bør forholde oss til. Derfor definerer vi en array som inneholder alle de relevante nøklene, og bruker dette som et filter. I vår løsning er oppgaver med status ferdig irrelevante og vi filtrerer også ut disse.

```
data.map (entry) ->
    Object.keys(entry).forEach (property) ->
        if property not in relevantKeys then delete entry[property]

data = data.filter (entry) => entry.Status isnt "Done"

console.log "\n#{JSON.stringify entry, null, 4}\n" for entry in data
```

Figur 5.4 – Skjermbilde av funksjon som filtrerer ut overflødig data.

Til tross for at vi får lastet inn dataen og brukt den i våre skripter, forblir det en utfordring å implementere slik data i et grafisk brukergrensesnitt, da flere dataverdier fra test-dataen er tomme. Eksempler på dette innebærer sprintnavn, storypoints og tidsfrister. Sprintnavnene ble tapt i konverteringen fra CSV til JSON, mens storypoints var sporadisk

definert, og tidsfrister var aldri definert. Ettersom størrelsen til JSON-filen er på omtrent 70 000 linjer anser vi det som lite fornuftig bruk av vår tid å manuelt endre på test-dataen før den kan brukes i et grensesnitt. Vi anser det også som unødvendig bruk av tid på å lage egendefinerte algoritmiske funksjoner for å forfalske dataverdier. Derfor velger vi å implementere statisk fiktiv data i grensesnittet, og heller benytte oss av test-dataen for å se hvilke parametere vi har tilgjengelig under utviklingen.

Selv om back-end funksjonalitet ikke blir – og aldri var – et stort fokusområde under vårt prosjektarbeid, vil det fortsatt være en vesentlig del av den endelige løsningen og derfor en betydelig del i videreutvikling av prosjektet. Dette går vi nærmere inn på i *kapittel 7 – Videreutvikling*.

5.2.3 Front-end

Som nevnt tidligere i kapitlet velger vi å plassere et sterkt fokus på språk som fremmer syntactic sugar og en whitespace sensitiv syntaks for å skape en kodebase som er semantisk, lesbar, minimal og håndterbar. Dette utgjør et godt utgangspunkt for et teknisk samarbeid på tvers av ulike kompetansenivå. Slike typer språk skaper ryddigere og mer strukturerte koder som antas å gjøre det enklere å ta over andres arbeid. Samtidig opplever vi at koden i helhet blir mer lesbar og lettere å navigere seg gjennom. De kodespråkene vi velger å bruke for dette, er de ovennevnte preprosesserings-språkene Pug, Sass og CoffeeScript. I tillegg til alle fordelene dette medbringer internt i gruppen, mener vi også at en slik kodebase bidrar til en inkluderende dialog med oppdragsgiver. Dette er på bakgrunn av ønske fra oppdragsgiver om å følge utviklingen av prosjektet – ikke bare på et konseptuelt nivå, men også et teknisk nivå.

Pug er et kodespråk som genererer HTML-markup for nettleseren. I tillegg til sin minimalistiske syntaks, har Pug ekstensiv tilleggsfunksjonalitet. Pug tillater blant annet å bruke maler, noe vi benytter oss av for å implementere den overordnede strukturen til løsningen. Resultatet er en body-tag som kun inneholder tre linjer, hvor hver tag laster inn en mal; en for navigasjonen, en for hovedinnholdet, og en for vår footer.

```
body: #app

    header: include ./includes-pug/menu.pug

    router-view.wrapper

    footer: include ./includes-pug/footer.pug
```

Figur 5.5 – Skjermbilde av innholdet i prosjektet sin body-tag.

For å partisjonere applikasjonen lager vi en mal for hver av våre tre undersider; prosjektoversikt, ressurspersoner og ressursbelastning.

For å strukturere flyten mellom data og grensesnitt i løsningen, velger vi å implementere et JavaScript-rammeverk. Det er per skrivende dato høy konkurranse mellom slike rammeverk, hvor Angular, React og Vue er mest utbredt. Oppdragsgiver har tidligere informert om at de har spisset sin kompetanse mot React og Angular, men at de ser stor verdi i nytt prosjekt som bygger på Vue i interesse av å utforske ny teknologi. Vi implementerer Vue.js via jsDelivr, et open source innholdsleveringsnettverk (CDN). For å skape en bedre brukeropplevelse bruker vi Vue.js og dens tilleggsmodul kalt Vue Router for å refaktorere koden til å bli en *single page application*. Nå lastes hver underside dynamisk inn i container-strukturen ved navigering av applikasjonen, slik at ikke hele siden behøver å laste inn på nytt. Vi lager så tre separate CoffeeScript-filer for hver underside, dette blir skript-filer som er ansvarlige for å definere relevant funksjonalitet som klikk-handlinger og lignende, i tillegg til å oppdatere Vue-instansen sin datatilstand.

Sass – et språk som kompileres ned til CSS – brukes for å stilsette løsningen. Sass – som Pug – tilbyr også ekstensiv ekstra funksjonalitet utover sitt originalspråk som vi benytter for å implementere vår løsning. Her benytter vi oss av blant annet variabler, nesting og mixins. Det gjør at vi enkelt kan sikre sammenhengende og gjennomgående stiler gjennom hele løsningen. Blant annet bruker vi dette på størrelser, farger og avstander. En annen svært nyttig fordel ved å bruke mulighetene Sass inkluderer, er for eksempel dersom vi bestemmer oss for å endre en hovedfarge, gjøres endringene kun ett sted i kodebasen. Måten vi velger å kode i front-end antar vi vil sikre effektivitet og raskere måte å takle endringer og utfordringer på. Dette sikrer vi blant annet ved å sette identifiserbare klasse- og ID-navn på alle elementer.

```
$sizes: small, medium, big
$sides: top, bottom, left, right

@each $size in $sizes
    @each $side in $sides

        %m-#{str-slice( $side, 0, 1 )}-#{str-slice( $size, 0, 1 )}
            @if $size == small
                margin-#{$side}: 12 / $rembase + rem
            @if $size == medium
                margin-#{$side}: 24 / $rembase + rem
            @if $size == big
                margin-#{$side}: 48 / $rembase + rem

        %p-#{str-slice( $side, 0, 1 )}-#{str-slice( $size, 0, 1 )}
            @if $size == small
                padding-#{$side}: 12 / $rembase + rem
            @if $size == medium
                padding-#{$side}: 24 / $rembase + rem
            @if $size == big
                padding-#{$side}: 48 / $rembase + rem
```

Figur 5.6 – Skjermbilde av en Sass mixin som gir rytmisk konsekvent avstandsforhold.

For å stile opp løsningens overordnede struktur kombinerer vi CSS flexbox og CSS grid. Det viser seg å være en uforutsett utfordring å få rader mellom sidebar og innhold til å samsvare med hverandre. Selv om løsningen illustrerer tenkt struktur, er ikke dette fleksibel og produksjonsverdig kode. Videre stiler vi alle våre komponenter, dette inkluderer blant annet avkrysningsbokser, radioknapper og navigasjonsknapper for å skape enkel gjenbruk av komponenter. For å holde presentasjon separat fra logikk, altså CSS separat fra JavaScript, kan vi bruke Sass sin eksklusive funksjonalitet for å designe våre komponenter. Et eksempel på dette er våre progresjonsbarer som implementeres med Sass-loops for å operere på verdier fra HTML-attributter for å unnlate å bruke JavaScript til dette formålet.

Figur 5.7 – Skjermbilde av Sass loop som bruker en HTML attributt for å stile et element.

For å gjøre siden interaktiv benytter vi oss av Vue sine klikk-handlinger som deklareres i markup. Vi definerer så metodene som deretter kalles i CoffeeScript konfigurasjonsfilene til undersidekomponentene. Vi bruker jQuery for å foreta DOM-manipulering slik at elementer på siden oppfører seg tilsvarende.

5.3 Kvalitetssikring

Som i nevnt i *delkapittel 5.2.1* som skildrer vår implementering av et utviklermiljø, bruker vi stilsjekkerne pug-, sass- og coffee-lint for å sikre en konsekvent og kvalitetssikret kode. Hver stilsjekker har sin egen konfigurasjonsfil med parametere en kan endre, slik at en selv må evaluere hva som anses som god praksis eller ikke.

Et eksempel på regler en stilsjekker kan innføre er hvor mange ordmellomrom et tabulatorhopp skal representere. For at kode skal være konsekvent og lesbar, bør hierarkiet i programkoden struktureres med konsekvente innrykk. Derfor velger vi at våre stilsjekkere skal varsle når noe annet enn et innrykk på fire ordmellomrom brukes. Dette gjør at kodeblokker i større grad blir gruppert etter gestaltloven om nærhet (Sandnes, 2011). En annen regel vi innfører er å advare om en linje overskrider 100 karakterer da vi anser linjer som strekker seg lengre enn dette å kompromittere leseligheten. En utfordring ved programvareutvikling er dypt nestet kode, der tomrommet etterlatt av tabulatorhoppene former en pyramide. Lange linjer og pyramidekode er begge karakteristikker som viser til rotete og lite lesbar kode.

> Web developers often find themselves having to deal with chains of asynchronous callbacks. These can lead to deeply nested code, termed a Pyramid of Doom (Eberhardt, 2014).

Kombinasjonen av linjelengde på 100 karakterer og tabulatorhopp med fire ordmellomrom vil motvirke pyramidekode, og heller oppfordre til alternative måter å uttrykke ønsket funksjonalitet på.

Størrelse på tabulatorhopp og linjelengde er språk-agnostiske regler. Hver stilsjekker har også mer språkspesifikke regler de kan innføre. Stilsjekkere har en mangfoldig liste over regler en kan innføre og mange regler kan ansees som mindre vesentlig. Videre i dette kapittelet forklares reglene vi har implementert som vi mener er mest viktige for både sluttbruker og utvikler.

For pug-lint lager vi to egendefinerte lister, en for attributter som ikke skal brukes, og en for tag-er som ikke skal brukes. Å definere slike lister gjør at en kan eliminere risikoen av å bruke tag-er eller attributter som er foreldet. Å bruke HTML-elementer som er foreldet betyr at det finnes nyere og bedre måter å skrive noe på. I tillegg risikerer en å bruke elementer som ikke lengre støttes av nettlesere. Å unngå å bruke foreldet markup sikrer et bedre grunnlag for utvikling av fremtidssikre løsninger. Som referansepunkt for hvilke attributter og tag-er vi skal unngå har vi konsultert ulike anerkjente ressurser på nettet som blant annet w3schools (www.w3schools.com) og CSS-Tricks (www.css-tricks.com).

Videre konfigurerer vi pug-lint for å innføre konsekvente sedvaner for hvordan markup skrives, blant annet ved å innføre en bestemt sekvens når en definerer elementer, der et eventuelt ID navn må komme først, etterfulgt av eventuelle klassenavn, og attributter til slutt.

Vi konfigurerer Sass-lint med samme tankesett som med pug-lint, og definerer regler for å innføre konsekvent skriving av Sass-kode. For å ikke forurense det globale navnerommet i CSS i større grad enn nødvendig, isolerer vi klasser slik at de samme klassenavnene kan ha ulik betydning avhengig av kontekst. Et godt eksempel på dette er ulike komponenter som alle kan ha en aktiv tilstand og behøver derfor en egen ekstra klasse oppkalt etter tilstanden. For å gjenbruke en tilstandsklasse som active på flere komponenter behøver en derfor å isolere klassen under ulike kontekster. Å isolere klasser i CSS – også kjent som nesting i Sass – kan være en byrde på sluttbruker da det er kostbart for en web-applikasjon sin ytelse å strukturere CSS på denne måten. Vi bruker Sass-lint for å innføre *the inception rule* for å unngå denne fallgruven. The inception rule er et begrep først brukt av Ricalde (2011) i hans artikkel kalt *Nested selectors: the inception rule* hvor han forklarer at en aldri skal gå dypere enn fire nivåer i Sass da noe annet vil kompromittere ytelse og gjenbruksverdi (Ricalde, 2011).

Vi konfigurer Coffee-lint for å sikre en konsekvent kodestil. For å sikre at løsningen kjører på JavaScript-kode som fremmer god ytelse for sluttbruker implementerer vi modulen Uglify-es for å komprimere JavaScript-koden slik at den tar minst mulig plass og bidrar til raskere nedlastingstid.

For å best mulig sikre nettleserkompatibilitet planlegger vi å implementere modulene *postCSS* og *Babel,* videre planlegger vi å bruke JavaScript unit-testing rammeverket *Jasmine* for å sikre at applikasjonen takler stress-testing og eventuelle edge-cases. Grunnet tidsbegrensninger blir dette en lav prioritet. Derfor forklarer vi nærmere hvordan vi ønsker å bruke disse modulene i *kapittel 7 – Videreutvikling.*

5.4 Sammendrag

I dette kapitlet har vi sett på hvordan vi gjennomgående tar hensyn til arbeidsmiljø og -flyt for selve kodingen. En primær årsak for å gjennomføre koding på denne måten, er for å tilrettelegge for samarbeid. Det viktigste aspektet anser vi å være muligheter for videreutvikling, ettersom vi sammen med oppdragsgiver så oss nødt til å redefinere oppgavens krav. Videre forklares de viktigste stegene i utbyggingen av løsningen. I tillegg forklares hvilke regler vi har innført under utviklingen for å sikre konsekvent kode som tjener sluttbrukeren best. I kommende kapittel vil vi drøfte den endelige løsningen, der vi også evaluerer det tekniske sluttresultatet.

6. Endelig løsning

6.1 Introduksjon

Dette kapitlet gir en oversiktlig beskrivelse, samt evaluering av endelig løsning med bakgrunn i foregående kapitler om prototyping *(kapittel 4)* og implementering *(kapittel 5)*. I dette kapitlet presenteres forklaring på sluttresultatet og vi går videre inn på styrker og svakheter ved løsningen.

6.2 Diskusjon og evaluering

6.2.1 Prototype

Det er utarbeidet to hi-fi prototyper; én visuell klikkbar prototype som er laget i Adobe Xd, og én kodet prototype som inneholder statisk data og enkle JavaScript-funksjoner for å simulere tenkt interaksjon. Den kodede prototypen er utformet som en webløsning der det senere kan implementeres dynamisk data. Når vi evaluerer endelig løsning, tar vi utgangspunkt i den kodede prototypen.

6.2.2 Innhold

Den endelige løsningen inneholder det vi har kommet frem til er viktig av informasjon og data for å skape et godt oversiktsbilde. Det gir oversikt over status på prosjekter, sprinter og ressurser, samt hvordan disse er belastet. Videre inkluderer vi sekundærinformasjon, som blant annet navn og beskrivelse på sprinter og oppgaver. Dette gjør vi for å sikre en optimal løsning som gir mest mulig verdi til bruker. Samtidig sikrer dette unødvendig hopping mellom Jira og vårt tilleggsverktøy.

Som nevnt tidligere inneholder sluttløsningen statisk data. Dette er fordi vi så det mer hensiktsmessig å løse det overordnede målet med oppgaven – å utvikle en løsning som gir ønsket oversikt for prosjektledere og derfor dekker behovet til oppdragsgiver. Derfor endre målet seg fra å utvikle en ferdig fungerende løsning, til å utvikle en prototype som kan videreutvikles med utgangspunkt i strukturen og grensesnittet vi har utviklet.

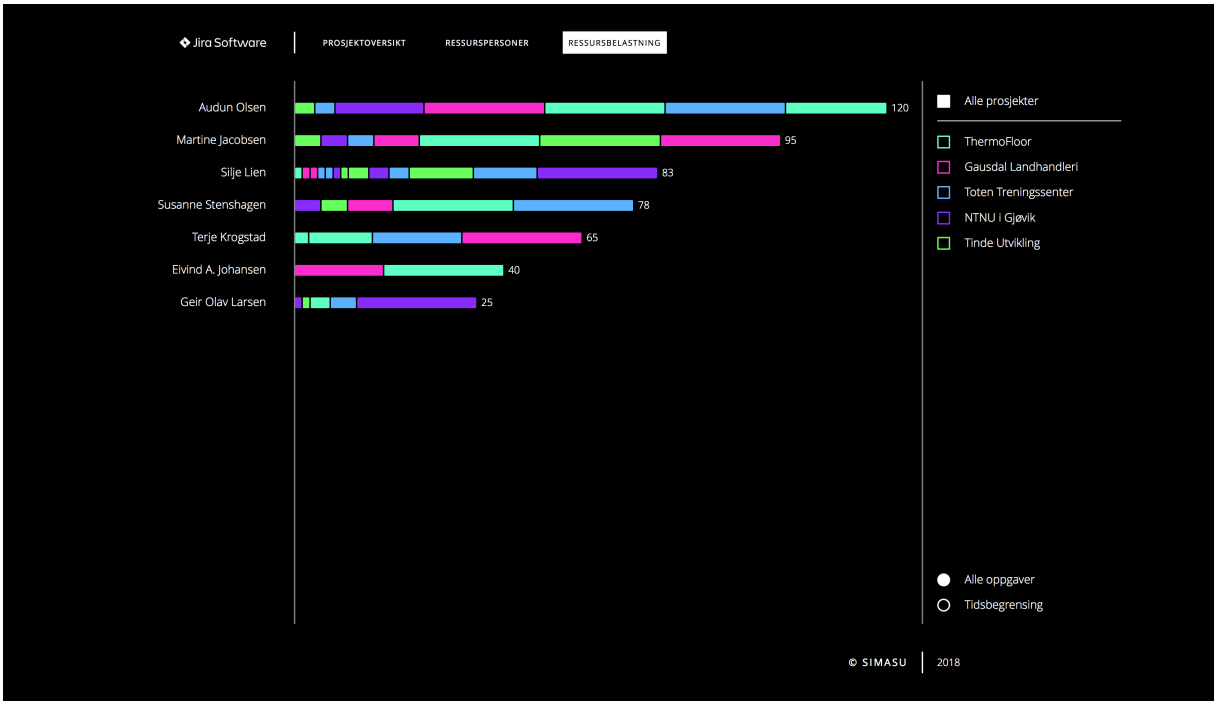
Løsningen inneholder flere fargeelementer som blir brukt som et visuelt hjelpemiddel for å tydeliggjøre informasjon, men det er aldri det eneste hjelpemiddelet. Prosjektene har egne farger, i tillegg til tekst for å opprettholde kravene innen universell utforming.

Under prosjektsiden er det viktig å påpeke at varsel om sprintbelastning er relativ i forhold til de ulike Scrum-teamene. For å avgjøre om en sprint er overbelastet eller ikke, vil en videreutvikling av løsningen se på foregående sprint, og hente ut *velocity*. Systemet skal sette antall fullførte storypoints mot antall planlagte storypoints og deretter regne ut

om kommende sprint har for mange storypoints. Symbolet om varsel vil dermed bli synliggjort før sprinten er i gang og overestimering av arbeid vil kunne unngås.

Figur 6.1 – Prosjektoversikt.

Figur 6.2 – Ressurspersoner.



Figur 6.3 – Ressursbelastning.

6.2.3 Teknologi og utvikling

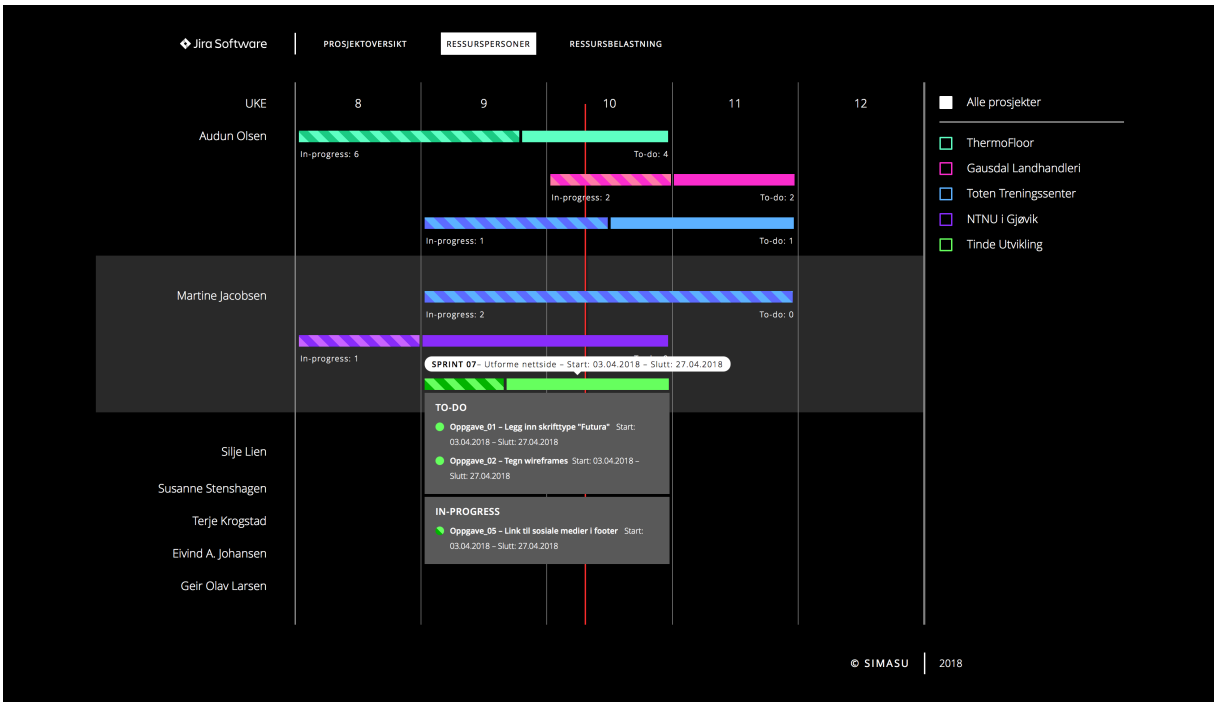
Node.js

Oppgaveteksten presiserer et ønske om en webapplikasjon som bygges på moderne webteknologier med Node.js på server-siden, samt et passende front-end JavaScript-rammeverk. Til tross for at oppgaveteksten presiserer at Node.js kan brukes som server-teknologi, kan Node.js også benyttes til å skrive byggeverktøy for å underbygge en solid kodebase, noe oppdragsgiver ser verdi i. Utviklermiljøet som er brukt har vi bygget fra bunn og skrevet selv i helhet – med Node.js. Vår anvendelse av tid sammen med oppgavens store omfang begrenset bruken av Node.js i den grad at vi ikke brukte Node.js på server, men anser dette som en viktig del av en fullt fungerende løsning. Node.js vil derfor bli en fundamental del av videreutviklingen.

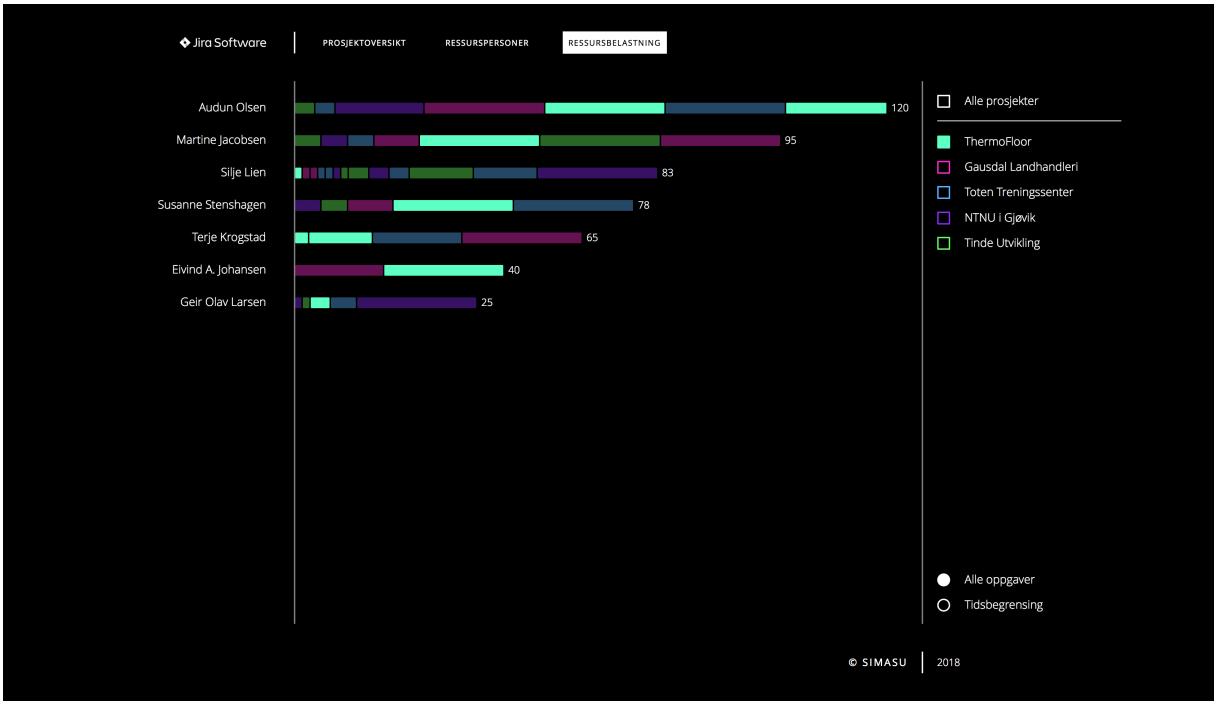
Funksjonalitet

Endelig løsning inneholder simulerte funksjoner som er kodet i JavaScript og jQuery. Disse funksjonene omhandler i hovedsak filtrering. Filtreringen er gjort i form av klikk-handlinger som viser og skjuler elementer ved hjelp av *opacity* og *Toggle*.

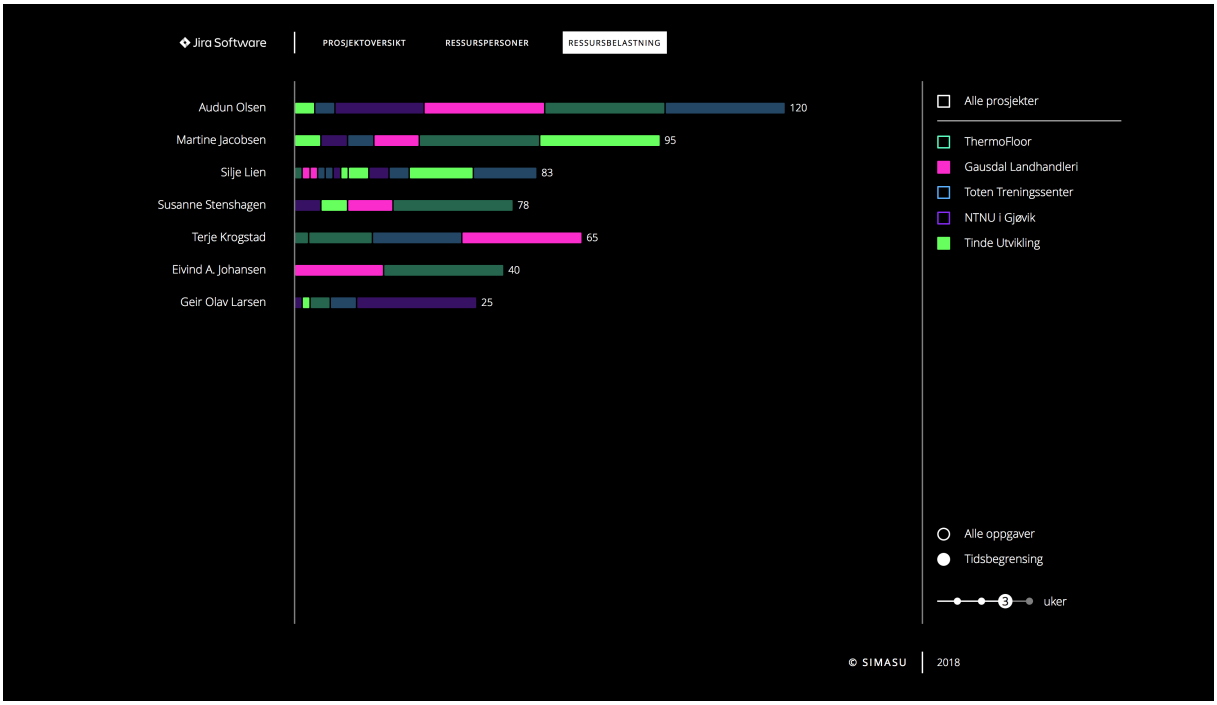
Figur 6.4 – Prosjektoversikt: Filtrering av tre prosjekter.



Figur 6.5 – Ressurspersoner: Tooltip og drop-down på én sprint for visning av oppgaver.



Figur 6.6 – Ressursbelastning: Filtrering etter ett prosjekt.



Figur 6.7 – Ressursbelastning: Visning av tidsbegrensning på tre uker og filtrering etter to prosjekter.

6.2.4 Styrker og svakheter

Styrker

Styrker ved løsningen vår er den grundige prosessen vi har gjennomgått. Den viser hvordan vi har kommet frem til et grensesnitt og informasjonshåndtering som vektlegger prosjektets problemstilling og illustrerer hvordan en ferdig implementert løsning vil fungere. Et aktivitetsforslag fra oppdragsgiver var planlegging av brukergrensesnitt. Siste brukstest antyder at vi har gjort gode valg som gjør løsningen brukervennlig og som dekker oppdragsgivers behov.

Visualisering av informasjon med grafikk bidrar til å gjøre løsningen intuitiv og gir et overblikk over status. Løsningen er lettvektig i form av inkorporering av webteknologier som forsørger minst mulig kodebase. Det gjør at applikasjonen har kort innlastningstid og er derfor rask i bruk. Visuelt er løsningen sammensatt ved konsis bruk av farger og elementer, som vi mener gjør at alle undersidene er sømløst knyttet sammen. Høy kontrast i form av fargebruk sikrer lesbarhet, fokus på viktig informasjon og tydeliggjøring av sammenkoblede elementer og funksjoner.

Svakheter

Den største svakheten ved løsningen er at det er en prototype. Det er ikke et fungerende system som behandler dynamisk data til å fremvise informasjon. Løsningen bygger som nevnt foreløpig kun på statisk eksempeldata og kan derfor ikke benyttes slik den er i dag. Fargebruk er eneste indikator på hvilke prosjekter ulike oppgaver tilhører, og det bør derfor vurderes om prosjektets navn skal legges til som et identifiserende element.

Griden som vises visuelt samsvarer ikke med slik den er kodet. Optimalt ville det vært en grid der kolonnehøyde baserer seg på radens høyde. Slik det er nå, er det en grid som er delt inn i tre deler; to sidekolonner og en midtseksjon, der midtseksjonen igjen inneholder en grid (ukesinndeling). Det betyr at vi har måtte benytte oss av margin og padding manuelt for å få en fullverdig visuell rad. Dette gjelder spesifikt undersiden *Ressurspersoner*. Bakgrunnsfargen som er tillagt annen hver person som separerer hver enkelt, er lagt til på en måte slik at den har en spesifikk plassering og ikke følger raden. Som forklart i foregående kapittel var dette en uforutsett utfordring. Ettersom løsningen illustrerer hva som er tenkt, mener vi likevel at det fungerer som en midlertidig løsning.

6.3 Konklusjon

Sluttproduktet samsvarer i stor grad med problemstillingen. Den sier at vi skal utvikle et system som skal gi oversikt, kartlegge ressursfordeling og foreslå løsninger for bedre allokering av ressurser for prosjekter i parallell. Det vi derimot ikke fikk tid til, og som raskt ble et langsiktig mål, var å implementere løsningen mot Jira. Sluttløsningen foreslår ikke løsninger for bedre allokering av ressurser for prosjekter i parallell, men den presenterer oppgaver som er mulig å allokere ved å merke disse som to-do.

Det som ble vektlagt i utviklingen var å få et brukergrensesnitt og sortering av informasjon slik at den ble mest mulig tilgjengelig og oversiktlig. Vi brukte mye tid på å kartlegge hva denne informasjonen skulle være og hvordan vise den på best mulig måte, med brukernes og oppdragsgivers behov i fokus.

Vi har opplevd utfordringer i bruken av Vue og innhenting av dynamisk data. Tidsmessig var Vue for nytt og krevende å lære seg for å kode gjennomgående i løsningen. Dette ble oppdaget for sent i prosessen, og kravene fra oppdragsgiver måtte reduseres. Derfor mener vi det er mest hensiktsmessig å levere et resultat som illustrerer tiltenkt struktur og funksjoner. Vi leverer derfor ikke en komplett og fungerende løsning, men en klikkbar prototype. Mye tid er brukt på konseptualisering og brukstesting, samt endring av prototype for å gi bedre brukeropplevelse. Vi valgte å gjøre dette for å være sikre på å ha en god grunnbase og vår vurdering er at løsningen er et godt utgangspunkt for videre utvikling og implementering.

6.4 Oppsummering

Som en foreløpig konklusjon kan vi si at vi ikke er kommet helt i mål med den opprinnelige oppgavens krav. Likevel viser vi evne til å reflektere over resultatet ved å presentere styrker og svakheter, samt vise forståelse over at tid var en viktig faktor i hvor langt vi har kommet. I påfølgende kapittel vil vi presentere vurderinger og forslag på videre utvikling og ferdigstilling av en fullstendig løsning.

7. Videreutvikling av løsningen

7.1 Introduksjon

Dette kapitlet tar utgangspunkt i resultater fra foregående kapittel og beskriver hva som gjenstår å implementere. Videre presenteres forslag til funksjonalitet som kan implementeres under en eventuell videreutvikling av løsningen.

7.2 Gjenstående arbeid

I denne delen gjennomgår vi gjenstående arbeid og forbedringer innen front- og back-end, for å kartlegge hva som må implementeres før et *minimum viable product* kan tilbys sluttbrukeren. Minimum viable product er minimumskravet for å ha en løsning som kan benyttes av en bruker.

7.2.1 Back-end

Å anvende Node.js for å implementere server-funksjonalitet var i utgangspunktet et ønske skildret i oppdragsgiver sin originale oppgavebeskrivelse. Som nevnt i denne rapporten viser tidlig dialog til gjensidig enighet om å avvente med funksjonalitet som krever en reell back-end løsning, og heller anse dette som et langsiktig mål å angripe skulle tiden strekke til. For å oppnå en fullt fungerende løsning forblir fortsatt Node.js en vesentlig del av prosessen videre. Node.js vil brukes for å tjene løsningen med reell data ved å gjøre spørringer opp mot Jira sin rest-API. Videre må vi implementere funksjonalitet som lytter til webhooks som initieres av Jira. Webhooks betyr å sende en varsling via HTTP-protokollen via post-metoden når en hendelse forekommer. Å bruke webhooks er nødvendig for at løsningen skal oppdatere sin data hentet fra Jira kun når faktiske endringer i Jira forekommer. Alternativet vil være å sette intervaller for når løsningen skal forsøke å hente oppdateringer, noe som ikke er gunstig for ytelse. Videre må server-funksjonaliteten kunne transformere og analysere data som hentes fra Jira. I vår løsning ønsker vi å identifisere i hvor stor grad en sprint kan anses som kritisk, noe som krever en egen formel. Om en sprint regnes som overbelastet eller ikke må være relativt til foregående sprinter sitt nivå av suksess, som målt av deres velocity. Denne dataen fra tidligere sprinter målt opp mot den planlagte totalen av storypoints i en aktiv sprint kan gi antydninger til om en sprint er i fare eller ikke. Med dette som grunnlag for en formel kan en kartlegge og identifisere hvilke data som er representative og nødvendige for å vise sammenheng mellom overbelastning, storypoints og aktuelle tidsrom.

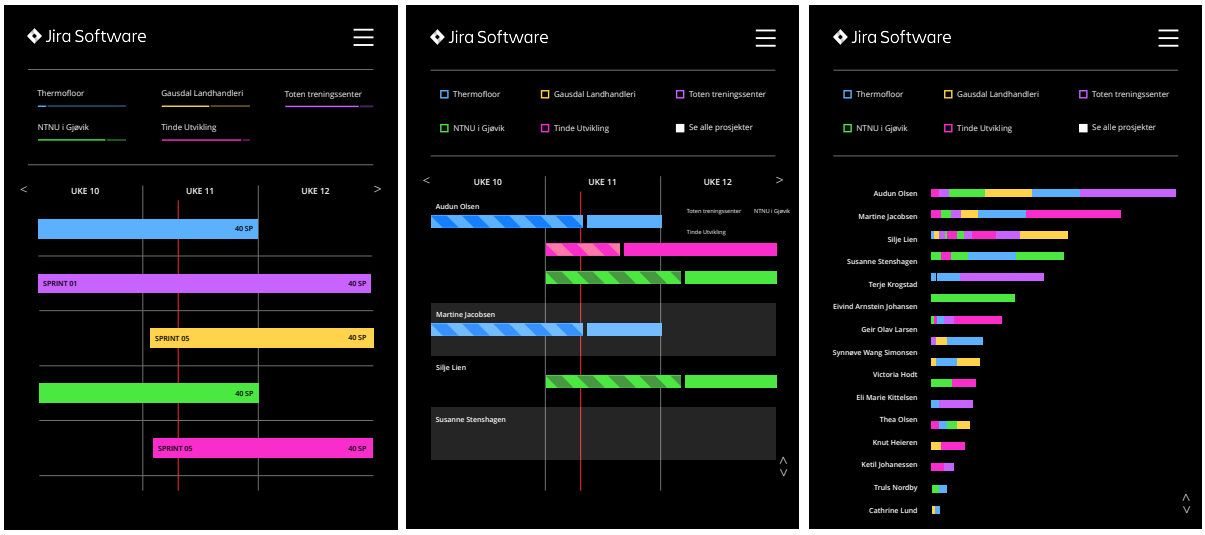
Løsningen må også integreres med Jira sine systemer, slik at den kan publiseres på Jira sin markedsplass som et tilleggsverktøy. Når brukere laster ned tilleggspakken må løsningen kunne aksesseres fra kontrollpanel-fanen i Jira sitt grafiske brukergrensesnitt.

7.2.2 Front-end

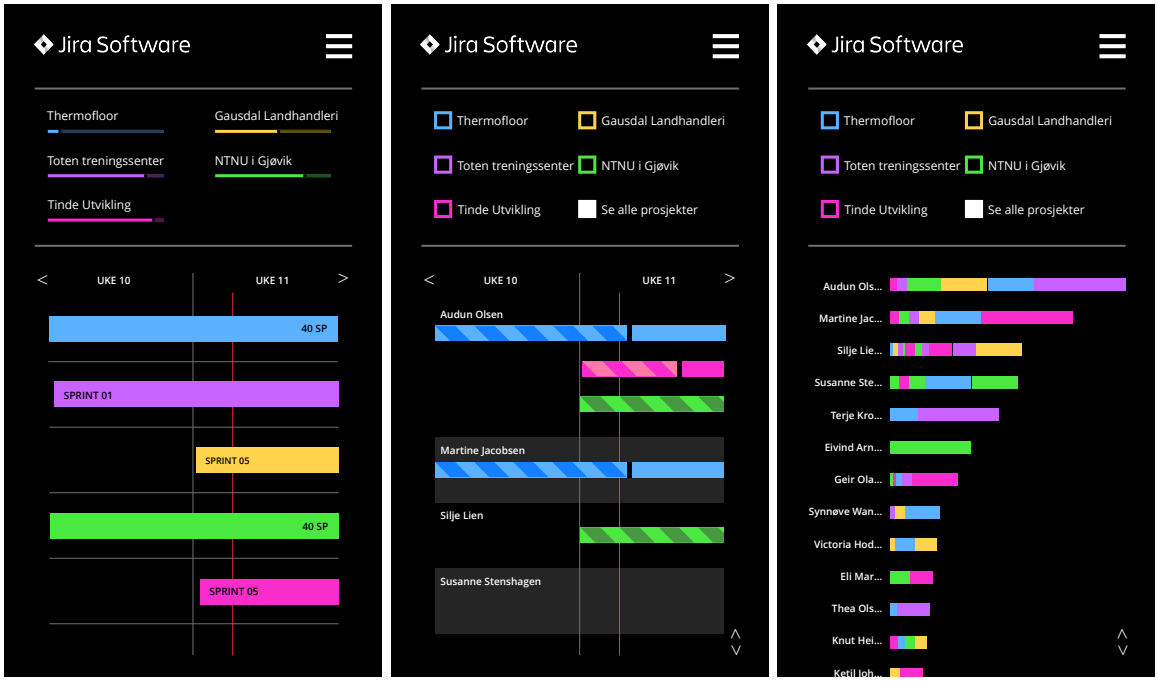
Et fokusområde vil være å gjøre nåværende front-end kode mer robust, med bedre integrasjon av rammeverket Vue.js og et MVC-mønster for å bedre strukturere flyt mellom applikasjonen sin datatilstand og brukerinteraksjoner. For øyeblikket har løsningen mye ad hoc-kode som ikke vil være kompatibel med dynamisk data. Derfor må vi refaktorere hardkodede elementer til å bli Vue-komponenter klare til gjenbruk. Funksjoner og komponenter bør også refaktoreres til å i større grad bruke data-attributter, da dette vil gjøre det enklere å skape samhandling mellom skripting og DOM-objekter.

Det neste steget vil være å fjerne jQuery fra vår kodebase. jQuery er et bibliotek som i utgangspunktet aldri var planlagt å bruke, men som ble innlemmet i prosjektet for å akselerere utviklingen ettersom alle gruppemedlemmer har kjennskap til biblioteket og gjorde det derfor enklere å programmere. Biblioteket brukes i vår løsning til å forenkle DOM-manipulering for å gjøre siden interaktiv for illustrerende formål. Med Vue.j kan dette gjøres direkte i HTML-markup med Vue-spesifikke attributter, noe som vil motvirke ad hoc-kode i tillegg til å harmonere bedre med en MVC-struktur. Vi anser også jQuery å være noe foreldet og redundant da vi bruker biblioteket til å hente og animere DOM-objekter, samt manipulere deres klasser. JavaScript har innebygd funksjonalitet for dette via querySelctor og classList-klassene. Animasjoner kan enkelt gjøres med CSS3-keyframes.

Designet må også implementeres for mobile enheter ved å implementere CSS mediaqueries. Dette sikrer en løsning som er tilpasningsdyktig for alle skjermstørrelser. Vi mener at en løsning sitt innhold i stor grad dikterer behovet for responsivitet. Selv om løsningen hovedsakelig er tiltenkt bruk på større skjermer, er vi bevisste på at anekdotisk bruk kan forekomme, og det er derfor viktig å være beredt med et responsivt design. Ved at løsningen fungerer uavhengig av skjermstørrelse er med å sikre god brukervennlighet uavhengig av enheter en har tilgjengelig. Vi har utformet forslag på et responsivt brukergrensesnitt, se figurene på neste side.



Figur 7.1 – Forslag til responsiv løsning for tablet.



Figur 7.2 – Forslag til responsiv løsning for mobilversjoner.

Ressurspersoner og ressursbelastning er to undersider som komplimenterer hverandre. Om en ser unormalt høy eller lav belastning hos en ressursperson under ressursbelastning, skal en ha enkel mulighet til å undersøke dette nærmere. I sidebaren hvor navnet til hver ressursperson er listet, anser vi det som ideelt at det linkes til ressurspersoner med en umiddelbar fremheving av den aktuelle personen. En slik markering vil være midlertidig med en bakgrunnsfarge som gradvis vil forsvinne. Med utgangspunkt i universell utforming bør det undersøkes hvordan dette kan kommuniseres til et skjermlesingsverktøy. Ved en slik markering mener vi det blir mer effektivt for en bruker å lokalisere aktuell person. Slik kan brukere utforske hvordan og hvilke oppgaver som kan tilegnes andre ressurspersoner, for å lette belastning på aktuelle

ressurspersoner. Hensikten er at oppgaver som er merket to-do er de der det kan være aktuelt å revurdere allokeringen.

Videre må informasjonsgrafikk plasseres i brukergrensesnittet, slik at den semantiske betydningen til innholdskomponenter kan ytterligere tydeliggjøres ved behov. Nederst i den høyre sidebaren til de respektive sidene vil et ikon som avbilder et spørsmålstegn vises. Ved å trykke på den vil en modal-boks vises som forklarer de ulike grafiske komponentene på siden. Et eksempel på dette vil være en modal-boks som forklarer risiko-ikonet og graderingen av farger belastning under prosjektoversikt.

7.2.3 Utviklermiljø

For å sikre en løsning som fungerer optimalt på tvers av ulike nettlesere, bør en kjøre koden gjennom en transkompilator. En transkompilator bruker kildekode for å produsere ny kildekode. *PostCSS* er en modul tilgjengelig fra NPM sitt register som brukes for å transformere CSS slik at den blir optimalisert for nettleserkompatibilitet. Det er derfor ønskelig å implementere postCSS i vårt utviklermiljø slik at CSS-koden som blir produsert av Sass-kompilatoren blir transpilert med postCSS før den er klar som endelig produksjonskode. Som et JavaScript-alternativ til postCSS, kan en bruke *Babel,* en modul som også er tilgjengelig fra NPM. CoffeeScript sin kompilator produserer ES6 JavaScript-kode, som per skrivende dato er den nyeste JavaScript-standarden. ES6 har mye funksjonalitet som er av stor verdi for utviklere, men som i dag ikke har blitt adoptert av alle nettlesere. For å kunne benytte seg av ES6-kode, kan en da bruke Babel for å transpilere koden, slik at koden optimaliseres for nettleserkompatibilitet. Vi ønsker derfor å bruke Babel som siste steg av JavaScript kompilasjoner før endelig produksjonskode er klar.

For å sikre at løsningen tåler eventuelle stress-tester hvor edge-cases kan oppstå, ønsker vi å benytte oss av et JavaScript *unit testing*-rammeverk. Slik blir utviklingsprosessen testdrevet. Vi ønsker å bruke *Jasmine* til dette formålet. Med Jasmine kan man bruke en fil kalt *spec* til å definere tester som forventer spesifikke resultater, der en blir varslet skulle en test feile.

For å forhindre potensielt feilaktig produksjonskode ønsker vi at ingen avvik fra våre stilsjekkere eller tester skal slippe gjennom fra en lokal versjon av kodebasen til den sentrale katalogen som ligger på Github. Derfor vil vi også implementere *Git hooks*. Git hooks tillater kjøring av egne skript før en spesifikk handling i Git forekommer. Resultatet av skriptet som kjøres kan avgjøre om Git skal avbryte eller gjennomføre handlingen. Vi ønsker å implementere et NPM-skript som kjører stilsjekkere og tester før en gjør en forpliktelse til sine endringer via *git commit*-kommandoen. Eventuelle tester og stilsjekkere som varsler om feil vil derfor nekte en utvikler å forplikte seg til sine endringer inntil feilene er rettet opp.

7.3 Videre utvikling

Selv når et minimum viable product er utviklet, ønsker vi fortsatt at nye iterasjoner av løsningen skal lanseres for å gi ny betydningsfull funksjonalitet til brukeren. I denne delen presenteres forslag til videreutvikling av løsningen.

7.3.1 Mulighet for endring av språk

Selv om det i denne oppgaven er en noe spesifikk målgruppe, skal løsningen være en utvidelse til Jira. Jira blir brukt av mange på tvers av landegrenser, og har støtte for et mangfold av språk. Det vil derfor være naturlig at løsningen benytter seg av det samme språket som et Jira-team bruker.

7.3.2 Mulighet for endring mellom lyst og mørkt tema

Vi ønsker også å gi brukeren muligheten til å endre mellom et lyst og mørkt tema i løsningens grensesnitt. I utgangspunktet har løsningen en svart bakgrunn med hvit skrift, men mulighet for å kunne endre dette til hvit bakgrunn med svart skrift er noe som kan tas med som en innstilling. En slik innstilling vil kunne tilpasse løsningen til brukerens preferanse, samt at det vil være mulighet for å tilpasse løsningen etter hvilke forhold den brukes i, der belysning spiller en rolle.

7.3.3 Omrokkering av oppgaver

Vår løsning setter status på oppgavene i fokus. Årsaken er at en oppgave med status to-do skal kunne flyttes fra en ressursperson med høy belastning over til en med lavere belastning. Å gå tilbake fra vår løsning til Jira sitt grensesnitt for å flytte en oppgave skaper en lang sekvens av handlinger før resultatet er oppnådd. Vi ønsker å implementere drag-and-drop funksjonalitet der en kan flytte oppgaver mellom ressurspersoner direkte i løsningens grensesnitt. For å forenkle omrokkering av oppgaver ytterligere, kan systemet komme med foreslåtte ressurspersoner til spesifikke oppgaver, ut fra hvilke kunnskaper det er registrert at de ansatte har. Omrokkeres en oppgave som omhandler endring i en database, bør personer med denne kompetansen bli foreslått. Jira støtter funksjonen ved å sette emneknagger til hver oppgave, men ikke til teammedlemmer. Derfor bør det implementeres en administratorside i løsningen, der en prosjektleder legger inn ansattes kompetanse. Slik kan systemet gi disse forslagene i samhold med Jira sine emneknagger.

7.3.4 Flux-arkitektur

Under prosjektarbeidet var vi ukritiske til bruk av Flux- og MVC-designmønstrene for å strukturere flyt, dette grunnet tidsrammer. Å hente data via server-funksjonalitet var ikke et fokusområde under prosjektarbeidet, derfor ble heller ikke et designmønster for å strukturere flyt mellom reell data på klientsiden et fokusområde. I *delkapittel 2.6.4 – Rammeverk* forklares de ulike strukturene, hvor Flux ble utviklet av Facebook for å kunne skalere bedre med komplekse web-applikasjoner (Salihefendic, 2015). Vi ønsker derfor å

ta høyde for at løsningen skal være beredt på å kunne øke i kompleksitet, og derfor strukturere flyt i applikasjonen gjennom Flux. For å strukturere løsningen etter en Flux-lignende arkitektur ønsker vi å implementere Vuex, et bibliotek for å sikre forutsigbar enveisflyt i Vue.js applikasjoner.

7.3.5 Kunstig intelligens

Et ønske fra oppdragsgiver var å se på muligheter for kunstig intelligens for å kunne gi brukeren tilbakemeldinger på potensielle risikoer og i tillegg presentere eventuelle løsningsforslag for bedre allokering. Å inkorporere kunstig intelligens krever spisset kompetanse som ingen i gruppen har, og det ble derfor for ambisiøst å sette av tid til å lære dette. Istedenfor å gjøre kunstig intelligens til et fokusområde, har gruppen valgt å presentere data fra Jira på en intuitiv måte som antas å gjøre det enklere for en prosjektleder å evaluere om nåværende allokering er bærekraftig, slik at hensiktsmessig handling kan utføres. Med vår løsning setter vi ikke arbitrære terskler for hva som anses å være god eller mindre god allokering, da det ikke ville ha skalert optimalt grunnet den varierende kapasiteten til ulike Jira-team. De ulike dataene settes istedenfor opp mot hverandre som grafiske elementer. Slik kan en prosjektleder se med lav kognitiv anstrengelse hvordan ulik belastning mellom ressurspersoner og prosjekter er relative til hverandre. Med kunstig intelligens kan vi gjennom maskinlæring avgjøre om et Jira-team over- eller underestimerer sin arbeidsmengde. Videre kan det synliggjøre hvordan de allokerer arbeidet godt eller ikke ved å bruke data fra tidligere prestasjoner. Vi har blitt oppmerksomme på verktøyet TensorFlow, et open source maskinlæringsrammeverk utviklet av Google som kan bidra til å forenkle implementeringsprosessen av kunstig intelligens i vår løsning. TensorFlow har også en JavaScript-variant som kan benyttes for å iverksette maskinlæringsmodeller i nettleser-baserte applikasjoner. Dette rammeverket kan derfor være en god kandidat å benytte seg av i videre utvikling.

7.4 Oppsummering

I dette kapittelet har vi sett på gjenstående arbeid, samt vist forslag til videreutvikling av løsningen hvor vi skildret hvilke funksjoner vi anser som naturlig å implementere i en fremtidig versjon. I neste kapittel konkluderes rapporten med å referere til problemstillingen, samt se på måloppnåelse.

8. Oppsummering og konklusjon

8.1 Introduksjon

I dette kapitlet avsluttes rapporten med en endelig konklusjon. Løsningen vår settes opp mot problemstillingen; å utvikle et system som skal gi oversikt, kartlegge ressursfordeling og foreslå løsninger for bedre allokering av ressurser for prosjekter i parallell. Systemet skal utvikles som en moderne web-applikasjon.

Vi vil evaluere om vi fullførte målene, og ser tilbake på prosessen og tidligere kapitler i rapporten, med en gjennomgang av våre metoder, aktiviteter og gruppens samarbeid i løpet av prosjektet. Til slutt går vi gjennom problemstillingen og konkluderer.

8.2 Sluttresultat

8.2.1 Hovedmål

Hovedmålet var å utarbeide en løsning for Escio som ga et informativt og oversiktlig innblikk i pågående prosjekter. Løsningen gir et forslag til prioriteringer og omrokkeringer, og gir prosjektleder et ryddig overblikk over status på de ulike prosjektene. Målet om å tilegne ferdigheter til ressurspersoner, samt vise tilgjengelige arbeidstimer ble nedprioritert, ettersom oppgaver vi anså som viktigere fikk større fokus. Denne vurderingen ble gjort i samråd med oppdragsgiver, da det var gjensidig enighet om hvilke oversikter som hadde størst verdi.

8.2.2 Effektmål

Ved gjennomføring av dette prosjektet har vi gitt oppdragsgiver et utgangspunkt for et verktøy som gir oversikt over ressurs- og prosjektbelastning. Dette vil gi prosjektledere mulighet til å allokere sine ressurspersoner på en smartere måte. Ved å gi en slik oversikt mener vi løsningen legger til rette for at prosjektleder kan ta gode valg i henhold til informasjonen som vises. Et ønsket effektmål av løsningen var å forebygge risikoen for at ansatte utsettes for uforsvarlige arbeidsmengder. Løsningen gir indikasjoner på risiko, men avhenger likevel av en prosjektleders evne til å evaluere best mulig allokering. Til slutt var målet med en ferdig utviklet løsning å styrke Atlassian Jira ved å tilby ny og verdifull funksjonalitet til tjenesten. Alle effektmål anses som oppnådd da også siste effektmål delvis er oppnådd ved at løsningen er et utgangspunkt for videre utvikling, og kan bidra til ny funksjonalitet ved at løsningen. Når løsningen er ferdig utviklet vil verktøyet være tilgjengelig for flere en kun oppdragsgiver. Det antas at ettersom en slik løsning er av verdi for oppdragsgiver, vil dette også være av verdi for andre.

8.2.3 Resultatmål

Etter gjennomført prosjekt har vi utviklet et utgangspunkt for å programmere en fremtidssikker løsning i Vue.js og Node.js som kartlegger ressursbruk for prosjektledere. Kodebasen vi har etablert gir Escio et godt grunnlag for videre utvikling. Målet om bruk av Node.js på serversiden ble ikke nådd da gruppen tidlig valgte et sterkere fokus på konseptualisering – med metoder som blant annet brukstesting og prototyping. Vi har brukt Vue.js til å forme en *single page application*, men ikke til å strukturere en god MVC-flyt. Det viste seg å være for tidkrevende å lære seg å bruke Vue.js til sin fulle potensiale innenfor prosjektets tidsramme. Node.js ble benyttet for å skrive egne skripter for å kjøre vårt eget utviklermiljø. Vårt sekundære mål var å utvikle en løsning som bidrar til å effektivisere tidsbruk i et Scrum-orientert selskap. Vi mener at vi har innfridd dette målet ved å konseptualisere en slik løsning. Måloppnåelse er basert på egne evalueringer og i samråd med oppdragsgiver.

8.2.4 Læringsmål

Vi har tilegnet oss kjennskap om ny teknologi – Vue.js og Node.js. Det vi anser som et av de største tekniske læringsutbyttene vi tar med oss fra prosjektet, er evnen til å samarbeide med flere personer i et større utviklermiljø. Dette inkluderer bruk av Git og GitHub på kommandolinjen, samt arbeide i en større kodebase med semantisk oppdeling av filer. Et viktig læringsmål for oss var prosjektgjennomføring og -ledelse av et større prosjekt. I løpet av denne perioden har vi opparbeidet oss verdifulle erfaringer vi kan dra stor nytte av senere i arbeidslivet.

8.3 Prosessevaluering

8.3.1 Metoder

Metodene vi valgte har vært svært viktige for vår prosess, fordi vi har tilegnet oss god forståelse av oppgavens krav, samt begrenset omfanget til å fungere innenfor prosjektets tidsplan og gruppens kunnskapsnivåer. Det viser hvordan vi kom frem til en god løsning som ivaretar oppdragsgivers ønsker og behov, ved hjelp av iterasjon og videreutvikling. Blant de metodene vi ønsker å fremheve er Scrum, brukerintervju, samt prototyping og brukstester. Flere av metoden som har blitt brukt bygger på hverandre og vi presenterer hvordan flere av metodene ble brukt. Vi legger frem hvilke fordeler dette ga oss, samt hvordan de ble brukt med – utgangspunkt i teori.

Scrum

Scrum har blitt fulgt kontinuerlig i løpet av prosjektet. Vi har brukt metoden i planleggings- og gjennomføringsfaser, samtidig som vi gjennomførte sprint retrospektiv (vedlegg 6) og sprint review (vedlegg 7) for å gjennomgå og evaluere fullførte sprinter.

Gruppen har jobbet etter Scrum tidligere i studiet, likevel brukte vi tid på å få inn gode rutiner ved bruk av Scrum. Dette gjelder spesifikt statusoppdatering i Jira. Gruppen benyttet seg av Jira for planlegging og utføring av prosjektet, med veiledning og oppfølging fra oppdragsgiver. Dette gjorde at vi fikk bedre forståelse og kunnskap om hvordan Jira fungerer som verktøy og hvilke muligheter som tilbys. En av reglene innen Scrum krever daglige møter, daily standups, noe vi ikke benyttet oss av ettersom ofte arbeidet sammen, samtidig som vi daglig kommuniserte via Facebook Messenger.

Brukerintervju

Brukerintervju var en av de første metodene vi benyttet oss av, og vi ser i ettertid at det var viktig å gjennomføre dette så tidlig i prosessen. Å gjennomføre brukerintervju ga oss dypere innsikt og forståelse av oppgavens krav og brukerens behov. Sandnes (2011) støtter opp om denne erfaringen, han forteller videre at en kan avdekke forventninger og hensikten med løsningen som skal utvikles. Det gjorde det enklere for oss å starte en skisse- og idéprosess. Samtidig fikk vi kartlagt oppdragsgivers mål og behov. Ettersom målgruppe var satt fra prosjektets begynnelse, slik vi har presentert i rapportens innledning, så vi det ikke som nødvendig å gå dypere inn på målgruppen. Brukerintervjuene mener vi ga oss informasjon som var mer verdifull for videre utvikling enn hva en målgruppeanalyse ville gitt. Årsaken til denne evaluering var oppdragsgivers ønsker og mål til en spesifikk løsning, og brukerintervju tillot oss å dypere inn på oppdragsgivers behov. Transkriptene fra brukerintervjuene var essensielle for å gjennomføre en detaljert PACT-analyse. Informasjonen fra disse transkriptene ble kategorisert og konkretisert gjennom analysen.

PACT-analyse

PACT-analysen bidro til å skape et tydeligere bilde på aktuelle handlinger og mulige funksjoner (Benyon, 2010). Det fikk oss til å tenke gjennom ulike brukssituasjoner og ga oss verdifull data som senere ble brukt for å utarbeide konkrete og realistiske personas og scenarier. Samtidig inkluderer analysen alle ønsker og krav oppdragsgiver hadde i utgangspunktet, og ble gjort før begrensninger ble satt. Det ga oss en forståelse av hva som var mulig, samtidig som vi fikk konkretisert hva som var viktigst å vise av informasjon.

Analysen, sammen med data vi fikk tilsendt av oppdragsgiver, var med å kartlegge hva som var viktig å registrere av data i Jira. Dette bidro også til å gi oppdragsgiver et innblikk i hva de bør sette som standard når planlegging blir gjort i Jira.

Ved at vi gjennomførte en detaljert PACT-analyse, la dette et godt grunnlag for utvikling av personas og scenarier, slik Benyon (2010) også hevder en slik analyse gjør. Dette var fordi informasjonen ga oss innsikt i ulike egenskaper personas kan ha, samt hvordan og i hvilke situasjoner løsningen kan bli brukt.

Personas og scenarier

Personas og scenarier ble utarbeidet med grunnlag i intervju (Cooper *et al.*, 2014) og PACT-analyse (Benyon, 2010). Scenariene bidro til dypere innsikt i løsningens funksjonelle innhold. Samtidig ga det oss et godt utgangspunkt for å utarbeide gode scenarier slik at vi fikk testet viktige aspekter ved løsningen. De gjorde det også mulig for oss å raskt teste små endringer vi gjorde underveis innad i gruppen. På denne måten unngikk vi å bruke mye tid på organisering og forberedelser til brukstester, da dette kunne vente til senere planlagte tester.

PACT-analyse og personas sikret at vi bearbeidet samme informasjon i flere steg av prosessen, og vi mener derfor det ga oss mer verdi enn analyse av en målgruppe som allerede var gitt. Målgruppe har likevel vært i fokus på den måten at oppdragsgiver har vært subjekt for brukstester og intervju. Samtidig har oppdragsgiver og veileder også gitt tilbakemeldinger underveis utenom planlagte tester.

Prototyping

I første del av prosjektet var skissering, som en tidlig fase av prototyping, viktig for å visualisere mulige utforminger av løsningen (Warfel, 2009). Det bidro til å skape en felles forståelse innad i gruppen og sammen med oppdragsgiver. Gruppen itererte på skisser og prototype som bidro til en jevn progresjon, og en løsning som inneholder elementer som er valgt med nøye utprøving. Gruppen har brukt skisser og prototyper for å visualisere detaljer som har blitt avdekket i arbeidet med PACT-analyse, samt personas og scenarier.

Brukstesting

Gjennom jevnlige brukstester, har vi sikret videreutvikling og utprøving av idéer og funksjoner. Brukstestene har vært svært avgjørende for valg vi har gjort. En av årsakene til at vi har basert valgene våre i stor grad på slike tester og tilbakemeldinger, er fokuset på brukersentrert design, også nevnt av Cooper *et al.* (2014). For oss har det vært viktig å designe et så optimalt brukergrensesnitt som mulig, og er derfor årsaken til at vi dedikerte mye tid til dette i prosessen.

Gantt-diagram

I startfasen benyttet vi oss av Gantt-diagram som et planleggingsverktøy. Se vedlegg 5. Dette bidro til bedre forståelse av prosjektets tidsperspektiv og omfang. Optimalt ville Gantt-diagrammet blitt oppdatert underveis for å representere det vi faktisk har gjort, for så å bruke dette i sammenligning og evaluering mot det som var planlagt. Som det er presentert tidligere i rapporten, kan en se hvor massivt diagrammet ble ettersom det er så detaljert. I tillegg lå dette på en annen plattform enn det vi har brukt underveis og ble derfor ikke hentet opp like ofte som først tiltenkt. Derfor valgte vi å fokusere på Scrum med backlog og sprintplanlegging – med utgangspunkt i Gantt-diagrammet. Vi fant vår egen metode for å sikre arbeidsflyt ved å lage en kalenderbasert oversikt med gjøremål for hver sprint, inkludert loggføring og rapportering. Dette fungerte godt, ettersom størrelsen

på Gantt-diagrammet gjorde at det opplevdes lite oversiktlig, og vi fikk separert gjøremål for rapport innen gjeldende sprint. Sprinter som allerede var gjennomført, ble derfor ikke et forstyrrende element i hver oversikt. Vi laget derfor flere tabeller med kortere tidsvisning som ble mer oversiktlig for den aktuelle tidsperioden, se vedlegg 8-12 for disse oversiktene.

Oppsummering av prosessevaluering

Verdien av metodene og utgangspunktene det har gitt oss videre i prosessen, har bidratt til gjennomgående og konsist arbeid innad i gruppen. Det har forenklet mange steg i prosessen fordi de bygger på hverandre, og gjør at informasjon vi har avdekket kan brukes videre og bearbeides. Det har derfor gitt oss et grunnlag for å sikre prosess og benytte oss av nye metoder.

8.3.2 Aktiviteter

Oppgaven var å utvikle et system som skulle gi oversikt, kartlegge ressursfordeling og foreslå løsninger for bedre allokering av ressurser for prosjekter i parallell. Systemet skulle utvikles som en moderne webapplikasjon. Ved siden av problemstillingen, har oppdragsgiver kommet med forslag til aktiviteter. Gruppen valgte seg ut noen av aktivitetene ettersom prosjektet ble avgrenset til å ikke inkludere alle ønsker og mål som presenteres i oppgaveteksten. Aktivitetene som ble gjennomført var:

- Dypdykk i Scrum som utviklingsmetodologi

Som forklart tidligere i dette kapittelet, benyttet vi Scrum som utviklingsmetode. Dette var et ønske fra oppdragsgiver, samtidig som det var et naturlig valg da dette er en metode som spesielt egner seg for system- og webutvikling der kravene til viss grad er uklare og kan endre seg underveis i prosessen.

- Detaljert behovsanalyse

Vi har ikke fulgt konkret metode og struktur for behovsanalyse, men gjennomføring av brukerintervju, PACT-analyse og utarbeidelse av personas og scenarier har fungert som en erstatter for en slik analyse. Det har gitt oss dyp innsikt i og god forståelse av oppdragsgivers brukerbehov.

- Planlegging av brukergrensesnitt. Målgruppe: daglig leder/prosjektleder

Da det ble klart at vi ikke skulle utvikle en ferdig løsning som implementerer alle krav til teknologi og funksjonalitet, ble denne aktiviteten et hovedfokus. Løsningen er utviklet basert på grundig arbeid som er direkte rettet mot brukergrensesnitt, brukervennlighet og -opplevelse.

- Utvikling av et system som kan:

  – avdekke svakheter/farer i planlagt ressursbruk

Løsningen vi har utviklet fokuserer, som tidligere nevnt i rapporten, på aspektet om informasjonsfremvisning. Ettersom vi ikke fikk utviklet et fungerende system, ble aktiviteten omformulert til *utvikling av forslag på løsning som avdekker svakheter/farer i planlagt ressursbruk*, slik at den samsvarer mer med problemstillingen.

- Brukertesting og evaluering av systemets resultater

Dette er en aktivitet vi har gjennomført på slik måte at flere brukstester er gjennomført. Og som nevnt i punktene over, fikk vi ikke muligheten til å teste et ferdig system. Brukstestene ble derfor gjort av itererte prototyper og brukt som grunnlag for videre iterering.

8.3.3 Gruppens samarbeid

Gjennom hele prosjektperioden har gruppen vært nøye med å føre timelogger og skrive møtereferater. Denne oppgaven ble tildelt et gruppemedlem, som da hadde hovedansvaret på å føre felles logg, samt skrive notater der det var nødvendig. Hvert enkelt gruppemedlem hadde selv ansvar for å loggføre timer ved selvstendig arbeid. Ved prosjektets start utarbeidet gruppen en detaljert prosjektplan, denne ble overført inn i et Gantt-skjema, som igjen ble brukt som utgangspunkt i videre produkt backlog og sprintplanlegging. Alle gruppens medlemmer var stort sett delaktige i sprint retrospektiv, men ikke alle har hatt mulighet til å delta hver gang. For hver retrospektiv har det vært minst tre gruppemedlemmer tilstede. Sprint retrospektiv var veldig nyttig, fordi det ga hver av oss en mulighet til å reflektere over arbeidet vi hadde lagt ned og hvordan vi kunne forbedre oss til neste sprint. Samtidig var en slik øvelse viktig med tanke på rapport og vår egen evne til å reflektere. En av de viktigste resultatene vi fikk med oss fra dette, var muligheten til å evaluere gruppens dynamikk og være bevisst på hvordan vi samarbeidet. Vi fikk avdekket hvorfor noe gikk bra og hvorfor konflikter oppstår.

Vi var tidlig bevisst på hvordan arbeidsfordelingen skulle være. Målet var å fordele oppgaver slik at alle jobbet med noe de ikke var like godt kjent med. På denne måten ville vi sikre best mulig læringsutbytte for alle. Det lot seg midlertidig ikke gjennomføre akkurat slik vi hadde tenkt. Tiden har vært avgjørende i valg av arbeidsfordeling. Derfor har flere oppgaver blitt tildelt gruppemedlemmer som vi antok ville utføre de effektivt og på best mulig måte.

Gruppen har hatt en gjennomgående god dynamikk. I det legger vi evnen til å samarbeide og kommunisere, i tillegg til jevn arbeidsinnsats. Vi ønsket et godt arbeidsmiljø, og utarbeidet derfor tydelige grupperegler. Se vedlegg 4. Her presiserte vi at det var viktig å avklare så tidlig som mulig at all åpenhet og ærlighet er essensielt for god gruppedynamikk. Men det er nødvendig at dette blir gjort med ydmykhet og respekt. En skal ikke angripe eller være nedlatende mot et annet gruppemedlem. Det kan skape dårlig stemning, samt påvirke både prosjektet og gruppen. Dette mener vi ble håndtert godt

ettersom det oppsto få uenigheter av større karakter. De konflikter som oppsto, ble avklart og løst innen rimelig tid. En viktig egenskap alle hadde var å være åpne for andre idéer samtidig som vi var trygge på å uttrykke egne meninger og synspunkter.

Noe vi erfarte underveis var viktigheten med oppdateringer på hva som var blitt gjort og hva hvert gruppemedlem arbeidet med. Fordi vi i starten ikke var så flinke til å oppdatere status på oppgaver i Jira, førte det til at det til tider var lett å miste oversikt over prosessen. Når en mister oversikt over hva de andre gjør og har gjort, kan det lett oppstå situasjoner der enkelte føler de gjør mer enn andre, og omvendt; at en ikke gjør nok. Det er sannsynlig at dette har vært tilfellet for vår gruppe også, men det er et tema vi har hatt svært få samtaler om. Det oppleves ikke som å ha vært et stort problem, da alle for det meste har vært i godt humør gjennom hele prosjektet, selv om det til tider har vært stressende.

Vi er en gruppe på fire personer, noe som kan sies å være en stor gruppe. Det har gjort at vi hatt økt fokus på kommunikasjon, da gruppestørrelsen gjør det mer utfordrende å holde hverandre oppdatert. Dette har vært aktuelt for vår gruppe, da en av medlemmene ikke bor i Gjøvik. For å sikre godt samarbeid, ble det satt opp faste tidspunkt der hele gruppen skulle samles fysisk. I utgangspunktet var dette to dager hver uke. Videre har det vært situasjonsbestemt når alle må møte. Dersom det en uke har vært flere viktige datoer hvor alle er tilstede, har det ikke vært nødvendig med møteplikt kommende uke, så lenge ingen viktige hendelser sto på planen. Dette gjelder i hovedsak gruppemedlemmet som ikke bor i Gjøvik. Grunnen til at vi har kunnet gjøre det slik, er nettopp fordi vi har hatt god rutine på å skrive konkrete møtereferater i Google Docs, og deretter diskutert dette via Facebook Messenger.

Alle medlemmene har tidligere jobbet sammen på lignende prosjekter. Vi var godt kjent med hverandres styrker og svakheter, i tillegg til bakgrunn og kompetanse. Det kan være en av de viktigste årsakene til at samarbeidet har gått så bra.

8.4 Konklusjon

I løpet av prosjektet har gruppen lært mye. Spesielt har vi hatt en stor læringskurve innen ny teknologi, der evnen til å samarbeide med flere personer i et større utviklermiljø og bruk av Git og GitHub på kommandolinjen anses som spesielt verdifullt. Innsikt i Jira og Escios arbeidsrutiner har vært essensielt for vår endelige løsning. Dyptgående forståelse for Scrum har stått sentralt i læringsprosessen. Metoden sikret god fremgang og struktur på prosjektet, i sammenheng med flere nyttige designmetoder. Vi har utviklet et forslag til et system som gir oppdragsgiver oversikt over allokeringen av ressurser for flere prosjekter i parallell, samt kartlegging av ressursbelastning, innad i bedriften. Ikke alle tekniske krav er oppfylt, men evalueringer og kartlegginger om videre utvikling tilrettelegger for at disse kan implementeres etter prosjektets slutt. Vi anser løsningen som et godt fundament for videre utvikling og integrering mot Jira. Ettersom løsningen

gir de oversikter og kartlegginger som problemstillingen tilsier, og oppgavens krav ble redusert, konkluderer vi med at problemstillingen er besvart.

Referanser

Allen, I. (2018) *The Brutal Lifecycle of JavaScript Frameworks*. Tilgjengelig fra: https://stackoverflow.blog/2018/01/11/brutal-lifecycle-javascript-frameworks/ (Hentet: 11. april 2018).

Beck, K. *et al.* (2001) *Manifestet for smidig programvareutvikling*. Tilgjengelig fra: http://agilemanifesto.org/iso/no/manifesto.html (Hentet: 11. april 2018).

Benitte, Greif og Rambeau (2018) *The State of JavaScript 2017*. Tilgjengelig fra: https://stateofjs.com/2017 (Hentet: 8. mai 2018).

Benyon, D. (2010) *Designing interactive systems : a comprehensive guide to HCI and interaction design*. 2nd ed. utg. Harlow: Pearson.

Chishkala, I. (2016) *The HTTP/2 Protocol: Its Pros & Cons and How to Start Using It*. Tilgjengelig fra: https://www.upwork.com/hiring/development/the-http2-protocol-its-pros-cons-and-how-to-start-using-it/ (Hentet: 10. april 2018).

Cooper, A. *et al.* (2014) *About face : the essentials of interaction design*. 4th ed. utg. Indianapolis, Ind: John Wiley & Sons.

Cuelogic Insights (2018) *Angular vs. React vs. Vue: A 2018 Comparison*. Tilgjengelig fra: http://www.cuelogic.com/blog/angular-vs-react-vs-vue-a-2018-comparison/ (Hentet: 6. mai 2018).

Cypriano, L. og Pinheiro, M. (2015) Prototyping and testing throughout all the design process as a methodology for developing interaction design projects (b. 9186, s. 157-166). doi: 10.1007/978-3-319-20886-2_16.

DeBill, E. (2018) Module Counts. Tilgjengelig fra: http://www.modulecounts.com/ (Hentet: 10. april 2018).

Eberhardt, C. (2014) *Tearing Down Swift's Optional Pyramid of Doom*. Tilgjengelig fra: http://blog.scottlogic.com/2014/12/08/swift-optional-pyramids-of-doom.html (Hentet: 10. mai 2018).

Escio (2018) *Om Escio*. Tilgjengelig fra: https://escio.no/om-escio/ (Hentet: 27. april 2018).

Gimeno, A. (2018) *The deepest reason why modern JavaScript frameworks exist*. Tilgjengelig fra: https://medium.com/dailyjs/the-deepest-reason-why-modern-javascript-frameworks-exist-933b86ebc445 (Hentet: 10. mai 2018).

Google (2017) *Open Sans – Google Fonts*. Tilgjengelig fra: https://fonts.google.com/specimen/Open+Sans (Hentet: 8. mai 2018).

Gyldendal (2018) *Philip Kotler*. Tilgjengelig fra: http://www.gyldendal.no/Forfattere/Kotler-Philip (Hentet: 4. mai 2018).

Hunt, P. (2013) *Why did we build React?* Tilgjengelig fra: https://reactjs.org/blog/2013/06/05/why-react.html (Hentet: 5. mai 2018).

Jongerius, P. (2012) *Get agile! : scrum for UX, design & development*. Amsterdam: BIS Publishers.

Kayes, I., Sarker, M. og Chakareski, J. (2016) Product backlog rating: a case study on measuring test quality in scrum, *Innovations in Systems and Software Engineering*, 12(4), s. 303-317. doi: 10.1007/s11334-016-0271-0.

Kearney, M. og Gaunt, M. (2018) *Set Up Your Build Tools*. Tilgjengelig fra: https://developers.google.com/web/tools/setup/setup-buildtools - what_is_a_build_process (Hentet: 10. april 2018).

Kerr, D. (2013) *An introduction to MVC frameworks*. Tilgjengelig fra: http://blog.scottlogic.com/2013/12/06/JavaScript-MVC-frameworks.html (Hentet: 4. mai 2018).

Kotler, P. (2005) *Markedsføringsledelse*. 3. utg. utg. Oslo: Gyldendal akademisk.

Krug, S. (2010) *Rocket surgery made easy : the do-it-yourself guide to finding and fixing usability problems*. Berkeley: New Riders.

Lacey, M. (2012) *The Scrum field guide : practical advice for your first year*. 1st ed. utg. Addison-Wesley Professional.

Lenz, J. og Fitzgerald, N. (2011) *Source Map Revision 3 Proposal*. Tilgjengelig fra: https://sourcemaps.info/spec.html (Hentet: 9. mai 2018).

Lovdata.no (2008) *Lov om forbud mot diskriminering på grunn av nedsatt funksjonsevne (diskriminerings- og tilgjengelighetsloven)*. Tilgjengelig fra: https://lovdata.no/dokument/LTI/lov/2008-06-20-42 (Hentet: 4. mai 2018).

Lovdata.no (2017) *Forskrift om universell utforming av informasjons- og kommunikasjonsteknologiske (IKT)-løsninger*. Tilgjengelig fra: https://lovdata.no/dokument/SF/forskrift/2013-06-21-732 (Hentet: 4. mai 2018).

Merriam-Webster (2018) *Pseudo | Definition of Pseudo by Merriam-Webster*. Tilgjengelig fra: https://www.merriam-webster.com/dictionary/pseudo (Hentet: 23. mars 2018).

NC State University, T. C. f. U. D. (1997) *The Principles Of Universal Design*. Tilgjengelig fra: https://projects.ncsu.edu/design/cud/about_ud/udprinciplestext.htm (Hentet: 11. april 2018).

Nolan, A. (2016) *The State of Front-End Tooling 2016 - Results*. Tilgjengelig fra: https://ashleynolan.co.uk/blog/frontend-tooling-survey-2016-results (Hentet: 10. april 2018).

Noll, J. *et al*. (2017) A Study of the Scrum Master's Role, i, *Cham*. Springer International Publishing, s. 307-323.

Norman, D. A. (2013) *The design of everyday things*. Rev. and exp. ed. utg. New York: Basic Books.

NTNU (2018) *Eivind Arnstein Johansen – NTNU*. Tilgjengelig fra: https://www.ntnu.no/ansatte/eivind.johansen (Hentet: 27. april 2018).

Pham, A. og Pham, P.-V. (2012) *Scrum in action : agile software project management and development*. Cengage Leraning.

Pick, C. (2015) *Why Babel Matters*. Tilgjengelig fra: https://codemix.com/blog/why-babel-matters/ (Hentet: 7. mai 2018).

Ricalde, M. (2011) *Nested selectors: the inception rule*. Tilgjengelig fra: http://thesassway.com/beginner/the-inception-rule (Hentet: 10. mai 2018).

Rosenberg, D., Stephens, M. og Collins-Cope, M. (2005) *Agile Development with ICONIX Process: People, Process, and Pragmatism*. Berkeley, CA: A-Press: Berkeley, CA.

Rosenfeld, L., Morville, P. og Arango, J. (2015) *Information Architecture*. O'Reilly Media, Inc.

Salihefendic, A. (2015) *Flux vs. MVC (Design Patterns)*. Tilgjengelig fra: https://medium.com/hacking-and-gonzo/flux-vs-mvc-design-patterns-57b28c0f71b7 (Hentet: 7. mai 2018).

Sandnes, F. E. (2011) *Universell utforming av IKT-systemer : brukergrensesnitt for alle*. Oslo: Universitetsforl.

Stack Overflow (2018) *Developer Survey Results 2018*. Tilgjengelig fra: https://insights.stackoverflow.com/survey/2018/ (Hentet: 22. mars 2018).

Stack Overflow (u.å.) *About - Stack Overflow*. Tilgjengelig fra: https://stackoverflow.com/company (Hentet: 22. mars 2018).

Toftøy-Andersen, E. og Wold, J. G. (2011) *Praktisk brukertesting*. Oslo: Cappelen Damm akademisk.

W3C (2008) *Web Content Accessibility Guidelines (WCAG) 2.0*. Tilgjengelig fra: https://www.w3.org/TR/WCAG20/ (Hentet: 9. mai 2018).

Warfel, T. Z. (2009) *Prototyping : a practitioner's guide*. Brooklyn, N.Y: Rosenfeld Media.

Webb, K. (1999) Raggett on HTML 4, Second Edition, *Internet Research*, 9(2). doi: 10.1108/intr.1999.17209baf.005.

Vedlegg

Vedlegg 1: Oppgavebeskrivelse fra oppdragsgiver

escio

Oppdragsgiver

Oppdragsgiver: Escio AS
Kontaktperson: Håvard Narvesen
Adresse: Jernbanesvingen 6, 2821 Gjøvik
Telefon: 990 35 053
Epost: hn@escio.no
Opplysninger om bedriften kan komme til campus og presentere oppgaven tirsdag 7.november: JA

System for oversikt over flere Scrum-prosjekter i parallell

Oppgaven

I "Vanilla scrum" skal prosjekter utføres av et team som er 100 % dedikert til ett prosjekt. I det virkelig liv er det ikke slik. Ressursene i teamet er ofte involvert i andre Scrum-prosjekter, helpdesk, salg eller annet.

I en organisasjon hvor det eksisterer flere Scrum-prosjekter i parallell er det behov for å lage en oversikt på "toppen" av alle Scrum-prosjekter som kan si noe om:

- Hvordan er belastningen på Team X eller ressurs Y?
- Er det konflikter i allokeringen? Er planlagt sprint realistisk basert på det som finnes av data?
- Kan systemet komme med anbefalinger til omdisponering av team basert på det vi vet om ressursene i firma og krav til faglig bidrag i prosjektene? (AI)

Det er ønskelig at systemet avdekker svakheter i planlagt ressursbruk og kan bidra til å forhindre forsinkelser og kostnadssprekker.

Forslag til aktiviteter som kan gjennomføres:

- Dypdykk i Scrum som utviklingsmetodologi
- Detaljert behovsanalyse
- Planlegging av brukergrensesnitt. Målgruppe: daglig leder / prosjektleder
- Fordypning i praktisk anvendelse av kunstig intelligens
- Utvikling av et system som kan:
    - Avdekke svakheter/farer i planlagt ressursbruk
    - Foreslå forbedringer
- Brukertesting og evaluering av systemets resultater

Ytterligere detaljer og informasjon deles med gruppen som velger oppgaven.

Teknologi og utvikling

Systemet ønskes utviklet som en moderne webapplikasjon med Node.js på serversiden og et passende frontend-rammeverk.

Data om Scrum-prosjekter og ressurser finnes i Atlassian Jira, og oppdragsgiver kan bistå med reell testdata.

Det er et stort pluss å ta sikte på at systemet senere kan benyttes som en "add-on" til Jira. Om løsningen blir god kan den tilbys på markedsplassen til Jira.

Atlassian Connect add-ons kan utvikles som separate webapplikasjoner der kommunikasjonen med Jira skjer over HTTP. Webapplikasjonen forholder seg til Jira på følgende, overordnede vis:

1. Mulighet for å plassere innhold/kode på gitte plasser i Jira sitt UI
2. Gjennomføre kall til Jira sitt REST API
3. Lytte og handle på WebHooks som initieres fra Jira

Vedlegg 2: Prosjektavtale

1 av 3

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato 25.01.2018 Vår referanse Martine Jacobsen

Prosjektavtale

mellom NTNU Institutt for design (ID) (utdanningsinstitusjon), og

ESCIO v/ Terje Krogstad (og Håvard Narvesen dersom Terje ikke har mulighet) (oppdragsgiver), og

Audun Meek Olsen, Silje Jeanette Fruseth Lien, Susanne Waaler Stenshagen og Martine Jacobsen (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 10.01.18 til 16.05.18 .

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU ID yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU ID står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, ID er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

Norges teknisk-naturvitenskapelige universitet
Institutt for design

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Eivind Arnstein Johansen

Oppdragsgivers kontaktperson (navn): Terje Krogstad

Student(er) (signatur): _____ dato 25.01.18

_____ dato 25.01.18

Silje Jeanette Fruseth Lien dato 25.01.18

_____ dato 25.01.18

Oppdragsgiver (signatur): Håvard Norvesen dato 30.01.18

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

3

Vedlegg 3: Prosjektplan

Prosjektplan

Tittel: System for oversikt over flere Scrum-prosjekter i parallell
Forfattere: Martine Jacobsen, Susanne Stenshagen, Audun Olsen, Silje Lien
Veileder: Eivind Arnstein Johansen
Oppdragsgiver: Escio
Antall sider: 16

1. MÅL OG RAMMER

1.1 Bakgrunn

Selskapet Escio organiserer sine prosjekter rundt samarbeidsverktøyet Jira som er en del av Atlassian sitt mangfoldige utvalg av team-orienterte verktøy. Jira er et verktøy som utmerker seg i blant annet smidige arbeidsmetoder. Escio, som utnytter seg av den smidige metodikken Scrum, ønsker en tilleggspakke til Jira de kan bruke til å visualisere allokeringen av ressurser og teammedlemmer som de har til disposisjon til de ulike oppgavene som inngår i et prosjekt. Tilleggspakken vil ha størst verdi for prosjektledere ettersom de har ansvaret å tilse at oppgaver er delegert rettferdig og fornuftig.

1.1.1 Dagens løsning

Per dags dato så benytter Escio seg av Atlassian Jira, som selv hevder å være det mest utbredte verktøyet for smidige team og brukes til å holde oversikt over de ulike faktorene som inngår i framdriften og problemer rundt et smidig prosjekt. Disse faktorene inkluderer da blant annet hvordan man disponerer sine tilgjengelige arbeidere, noe Escio ønsker å kunne se en mer helhetlig oversikt for. Escio ønsker ikke en alternativ løsning til Jira, men heller en tilleggspakke som avdekker svakheter ved allokeringen av ressurser i prosjektene sine, noe de mener er en av svakhetene til Jira.

1.2 Prosjektmål

1.2.1 Hovedmål

Vi vil utarbeide en løsning for Escio som gir et informativt og oversiktlig innblikk i pågående prosjekter, samt viser den enkelte ansattes tidsforbruk. Løsningen vil kunne gi forslag til prioriteringer og omrokkeringer, basert på den enkeltes ferdigheter og tilgjengelige arbeidstimer.

1.2.2 Effektmål

Ved gjennomføring av dette prosjektet vil vi oppnå følgende gevinster

- At prosjektledere vil kunne allokere sine ressurser smartere, med ressurser så menes de ansatte i et software utviklingsteam som prosjektledere har tilgjengelig til disposisjon
- Forebygge risikoen for at ansatte blir utsatt for urettferdige arbeidsmengder
- Styrke Atlassian Jira og tilby ny verdifull funksjonalitet til tjenesten

1.2.3 Resultatmål

Ved gjennomføring av dette prosjektet vil vi oppnå disse målene for løsningen

- Utvikle et oversiktlig organiseringsverktøy
- Utarbeide et brukervennlig system

1.3 Omfang

1.3.1 Oppgavebeskrivelse

Oppgaven er å utvikle et system som skal skaffe oversikt, kartlegge svak ressursbruk og foreslå løsninger for bedre allokering av ressurser for prosjekter i parallell. Systemet skal utvikles som en moderne webapplikasjon.

Kravspesifisering
Etter dialog med Escio har vi kommet til enighet om at vi står fritt til å utvikle løsningen etter egendefinerte krav til teknologier og design, Escio har kun preferanser, men ingen absolutte krav. Vi ønsker å fokusere på moderne teknologier på grunnlag av oppgavens store omfang, og at om oppgaven ikke ble ferdigstilt, så vil Escio få en fremtidssikker kodebase som er gunstig for videre utvikling. For å oppnå dette ønsker vi å bruke Node.js for å utforme et utviklermiljø samt backend funksjonalitet. Vi ønsker å bruke et passende front-end rammeverk sammen med fundamentale webteknologier som Javascript, HTML og CSS.

Når det kommer til visuell fremstilling av data, skal dette hovedsakelig være en nettside til visning på skjerm, men vi vil også lage den responsiv. Kodingen vil skrives i henhold til universell utforming.

1.3.2 Målgruppe

Målgruppen blir ansatte i Escio med rollen "scrum master", da vår løsning vil bli et internt arbeidsverktøy med størst verdi for de som behøver et overordnet blikk på effektiviteten rundt prosjekter sin framgang. Dersom løsningen blir vellykket, kan denne legges ut på markedet som en plugin til Atlassian Jira. Dette vil gi oss en større sekundær målgruppe som dekker software-team som utvikler etter smidige metoder.

1.4 Rammer og avgrensning

Prosjektet avgrenser seg til oppgavebeskrivelsens rammer og tidsperspektiv. Tid tilgjengelig er fra midten av januar til 16.mai. Løsningen skal ikke bli en ny versjon av Jira. Vi ble opplyst i begynnelsen at dette var en oppgave med et stort omfang, som krever bred kompetanse.

1.4.1 Kostnader

Escio nevnte i deres originale utkast av oppgaven at de ønsket at vi brukte web teknologien Node.js som grunnlaget for å utforme vår løsning. Javascript, et språk som før kun ble brukt til klientsiden av en webløsning, er nå med Node.js anvendt til å utvikle backend funksjonalitet. Vi har ikke lært Node.js i løpet av vår tilværelse på NTNU og er derfor enige som gruppe at vi skal være forberedt på kostnader rundt online kurs som dekker da Node.js og annen teknologi nødvendig for å fullføre prosjektet. Vi vil også kjøpe pensumbøker dersom det trengs. Vi har valgt å bruke Atlassian Jira for å komplimentere scrum arbeidsmetoden vi skal jobbe etter og har derfor blitt enige om å kjøpe abonnement til denne tjenesten.

1.4.2 Tidsramme

Prosjektet skal være fullført innen 16.mai 2018, og ulike faser er satt opp i et Gantt-diagram (se punkt 3.1.1). Vi har altså ca 18 uker til å fullføre prosjektet.

2. Prosjektorganisering

2.1 Ansvarsforhold og roller

OPPDRAGSGIVER
Escio ved Terje Krogstad

VEILEDER
Eivind Arnstein Johansen, universitetslektor ved Institutt for design på NTNU i Gjøvik

GRUPPELEDER | Martine Jacobsen
Gruppelederen sitt ansvar er å styre prosjektet, og passe på at de satte målene blir nådd. Ved diskusjoner og konflikter, har gruppeleder en overordnet rolle og kan ta endelige avgjørelser på vegne av gruppen.

SEKRETÆR | Susanne Stenshagen
Sekretæren har ansvar for loggføring av møter, samt booking av møterom og dokumentering av gruppens arbeidstimer.

DESIGNER | Silje Lien
Designeren har hovedansvar for det visuelle utseendet på løsningen, samt har fokus på brukervennlighet.

TEKNISK ANSVARLIG | Audun Olsen
Teknisk ansvarlig har ansvar for å sette opp utviklermiljø og vedlikeholde dette, samt begrunne avgjørelser rundt webteknologiene gruppen skal benytte seg av.

Til tross for våre tildelte roller, ønsker vi alle å bidra på tvers av disse ettersom alle gruppemedlemmer ønsker et læringsutbytte utenfor gruppemedlemmenes respektive komfort-soner.

2.2 Rutiner og regler i gruppa: Grupperegler

Se vedlegg 1.

3. Planlegging, oppfølging og rapportering

3.1 Prosjektfaser

FORPROSJEKT

- FASE 01 | Kartlegging
  UKE 2 – UKE 4: 10.01.2018–25.01.2018

  - Få innsikt i og forståelse av oppgaven og dens omfang
  - Kartlegge spesifikke tekniske krav
  - Oppstartsmøte med oppdragsgiver

- FASE 02 | Planlegging
  UKE 4: 25.01.2018–26.01.2018

  - Utforme utkast på prosjektplan
  - Utforme utkast på gruppeavtale
  - Innlevering av utkast på prosjektplan
  - Innlevering av utkast på gruppeavtale

- FASE 03 | Innlevering
  UKE 4 – UKE 5: *27.01.2018–01.02.2018*

  - Utforme endelig prosjektplan
  - Utforme endelig gruppeavtale
  - Innlevering av endelig prosjektplan
  - Innlevering av endelig gruppeavtale
  - Signering og innlevering av prosjektavtale

HOVEDPROSJEKT

- FASE 04 | Oppstart
  UKE 5: *02.02.2018–04.02.2018*

  - PLANLEGGING:
    - Ferdigstille oversikt/liste på kravspesifikasjoner (med tanke på product backlog)
    - Product backlog

  - AKTIVITETER:
    - Finne aktuelle artikler

  - RAPPORT:
    - Innhenting av informasjon fra artikler tilknyttet aktiv fase
    - Notater og refleksjoner
    - Logg

- FASE 05 | Forarbeid | *Sprint 1*
  UKE 6: *05.02.2018–11.02.2018*

  - PLANLEGGING:
    - Sprintplanlegging – sprint backlog

  - AKTIVITETER
    - Ferdigstille product backlog
    - Sette opp utviklermiljø
    - Workshop: gjennomgang og opplæring i Node og Vue

  - ANALYSE
    - Behovs-/brukerundersøkelse

  - RAPPORT:
    - Lese gjennom artikler
    - Utkast til rapportdisposisjon
    - Ferdigstille rapportdisposisjon
    - Sette mal / dokument for vedlegg
    - Innhenting av informasjon fra artikler tilknyttet aktiv fase
    - Notater og refleksjoner
    - Logg

- FASE 06 | Teknisk forarbeid | *Sprint 1*
  UKE 7 – UKE 8: *12.02.2018–25.02.2018*

- PLANLEGGING:
  - Innhente teknisk data til bruk (fra oppdragsgiver)
  - Kartlegge ønsket type data (fra oppdragsgiver/Jira)
  - Kartlegge tekniske løsninger for gjennomføring av utvikling
  - Kartlegge mulige løsninger for henting og visning av data

- UTVIKLING:
  - Utarbeide en teknisk prototype for henting og visning av data
  - Teknisk prototype: kun tekst og tall – ikke UX/UI og design

- RAPPORT:
  - Retrospektiv av sprint
  - Sette inn aktuelle vedlegg
  - Innhenting av informasjon fra artikler tilknyttet aktiv fase
  - Notater og refleksjoner
  - Logg

- FASE 07 | Analysering / Skisse / Prototype | *Sprint 2*
  UKE 9 – UKE 10: *26.02.2018–11.03.2018*

  - PLANLEGGING:
    - Sprintplanlegging – sprint backlog
    - Kartlegge muligheter for sammenligning og analysering av data

  - UTVIKLING:
    - Utforme tekniske analysemetoder
    - Skisse / wireframe på webapplikasjon for fremvisning av data
    - Low-fidelity prototype

  - RAPPORT:
    - Sette inn aktuelle vedlegg
    - Innhenting av informasjon fra artikler tilknyttet aktiv fase
    - Notater og refleksjoner
    - Logg

- FASE 08 | Testing | *Sprint 2*
  UKE 11: *12.03.2018–18.03.2018*

  - TESTING
    - Forarbeid til test – følge prosess for brukertesting
    - Brukertesting av low-fidelity prototype

  - ANALYSERING / EVALUERING:
    - Analysering og evaluering av innhentet data fra brukertesting

            - styrker, svakheter, forkaste, legge til
        - Kartlegge mulige endringer/løsninger
            - styrker og svakheter for disse
            - klargjøre disse til neste fase (lage oversikt/liste)

    - RAPPORT:
        - Retrospektiv av sprint
        - Sette inn aktuelle vedlegg
        - Innhenting av informasjon fra artikler tilknyttet aktiv fase
        - Notater og refleksjoner
        - Logg

- FASE 09 | Utvikling | *Sprint 3*
  UKE 12: *19.03.2018–25.03.2018*

    - PLANLEGGING:
        - Sprintplanlegging – sprint backlog

    - UTVIKLING:
        - Utvikle løsningen på bakgrunn av resultater fra brukertesting
        - Effektiv, systematisk og kort kode
        - Utforming av design
        - High-fidelity prototype

    - RAPPORT:
        - Retrospektiv av sprint
        - Sette inn aktuelle vedlegg
        - Innhenting av informasjon fra artikler tilknyttet aktiv fase
        - Notater og refleksjoner
        - Logg

- PÅSKEFERIE | Pause
  UKE 13: *26.03.2018–01.04.2018*

- FASE 10 | Videre testing | *Sprint 4*
  UKE 14: *02.04.2018–08.04.2018*

    - PLANLEGGING:
        - Sprintplanlegging – sprint backlog

    - TESTING:
        - Forarbeid til test – følge prosess for brukertesting
        - Test av brukervennlighet / design av high-fidelity prototype

- ANALYSE / EVALUERING:
  - Analysering og evaluering av innhentet data fra brukertesting
    - styrker, svakheter, forkaste, legge til
  - Kartlegge mulige endringer/løsninger
    - styrker og svakheter for disse
    - klargjøre disse til neste fase (lage oversikt/liste)

- RAPPORT:
  - Sette inn aktuelle vedlegg
  - Innhenting av informasjon fra artikler tilknyttet aktiv fase
  - Notater og refleksjoner
  - Logg

- FASE 11 | Videre utvikling | *Sprint 4*
  UKE 15: *09.04.2018–15.04.2018*

  - UTVIKLING
    - Utvikle/endre løsningen på bakgrunn av resultater fra brukertesting
    - Unngå å gjøre for store endringer
    - Unngå å legge til nye elementer/funksjoner

  - RAPPORT:
    - Sette inn aktuelle vedlegg
    - Innhenting av informasjon fra artikler tilknyttet aktiv fase
    - Notater og refleksjoner
    - Logg

- FASE 12 | Siste test | *Sprint 4*
  UKE 16: *16.04.2018–22.04.2018*

  - TESTING:
    - Forarbeid til test – følge prosess for brukertesting
    - Test av endringer fra forrige brukertesting

  - ANALYSE / EVALUERING:
    - Analysering og evaluering av innhentet data fra brukertesting
      - styrker, svakheter
    - Kartlegge mulige endringer/løsninger
      - styrker og svakheter for disse
      - klargjøre disse til neste fase (lage oversikt/liste)

  - RAPPORT:
    - Retrospektiv av sprint

        - Sette inn aktuelle vedlegg
        - Innhenting av informasjon fra artikler tilknyttet aktiv fase
        - Notater og refleksjoner
        - Logg

- FASE 13 | Ferdigstille | *Sprint 5*
  UKE 17: *23.04.2018–29.04.2018*

    - PLANLEGGING:
        - Sprintplanlegging – sprint backlog

    - UTVIKLING
        - Siste endringer på løsningen på bakgrunn av resultater fra brukertesting
        - Kun bugfix, små og konkrete endringer fra brukertesting

    - EVALUERING
        - Kontroller løsningen
        - Eventuelt rette opp i dersom løsningen – eller deler av den – ikke fungerer

    - RAPPORT:
        - Retrospektiv av sprint
        - Sette inn aktuelle vedlegg
        - Innhenting av informasjon fra artikler tilknyttet aktiv fase
        - Notater og refleksjoner
        - Logg

SLUTTPROSJEKT

- FASE 14 | Rapport
  UKE 18: *30.04.2018–06.05.2018*
    - FERDIGSTILLE:
        - Utdype/skrive om punkter som ikke er ferdigstilt underveis

    - KONTROLL
        - Kontrollere at rapporten er skrevet på en sammenhengende måte
        - Kontrollere at rapporten samsvarer med akademisk standard

    - VEDLEGG:
        - Kontrollere at alle vedlegg er lagt inn
        - Eventuelt samle og legge inn de resterende vedleggene

- PAUSE | Nullstille
  UKE 19: *07.05.2018–09.05.2018*

    - Hvile hodet og øynene for å kunne korrekturlese rapporten med friske øyne

- FASE 15 | Korrekturlesing
  UKE 20: *10.05.2018–15.05.2018*

    - Lese korrektur
    - Kontrollere referanselisten, sjekke at alle er brukt og henvist til
    - Finpusse oppsett
    - Evaluering

- INNLEVERING | Prosjekt og rapport
  UKE 21: *16.05.2018*

    - Levering av bachelorprosjektet

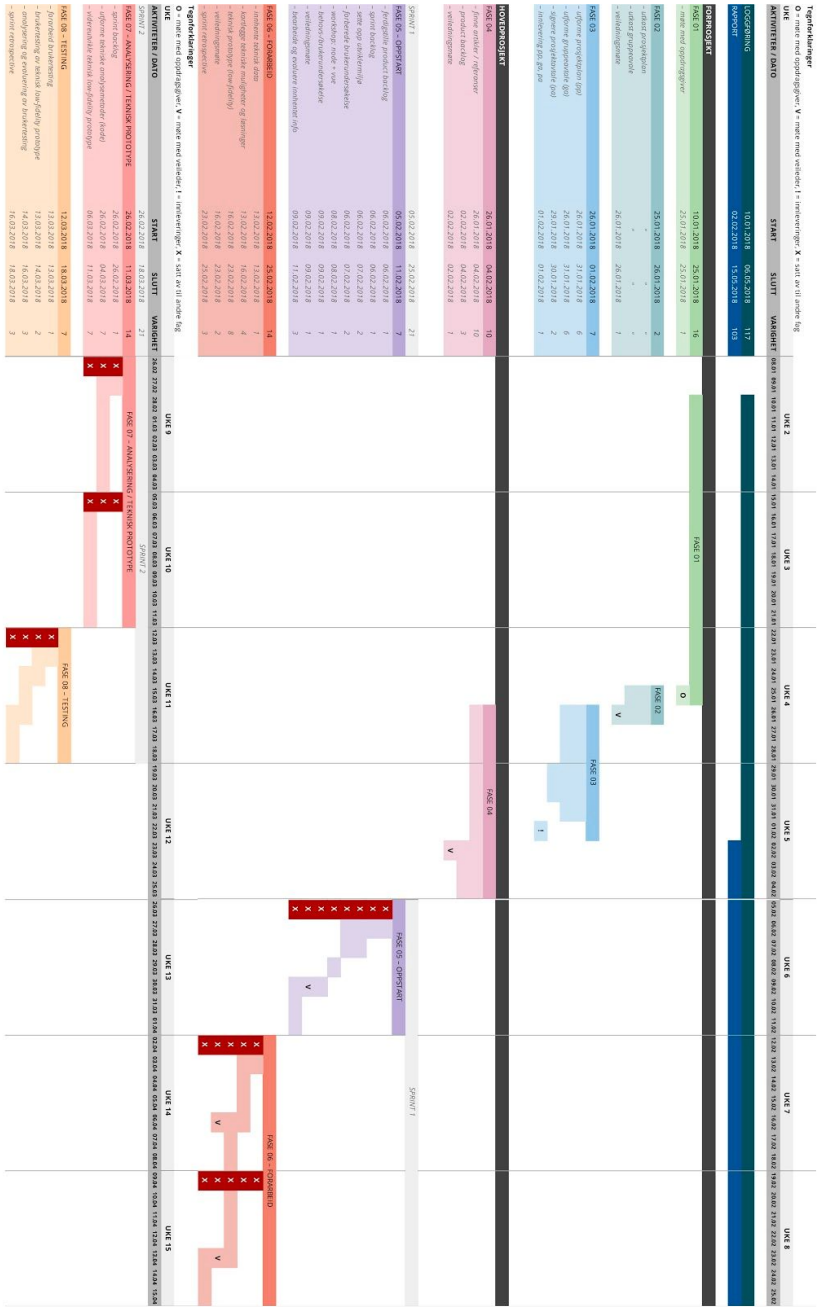
- FASE 16 | Presentasjon
  UKE 12: *18.05.2018–04.06.2018*

    - PLANLEGGING
        - Disposisjon av presentasjonen - hva må med, hva kan utelates?
        - Utarbeide presentasjonen - PDF / PPT
        - Utarbeide plakat for prosjektet

    - FORBEREDELSER
        - Fordele presentasjonen på medlemmene - hvem sier hva?
        - Hver person lager stikkord til sin del
        - Øve inn hva man skal si (individuelt)
        - Gjennomkjøring av presentasjonen sammen (gjerne flere ganger)
            - For å sikre at alle vet når de skal prate

    - PRESENTASJON
        - Ansvarsområder:
            - Teknisk – mac, kabler o.l. tyring av PDF / PPT – hvem?
            - Hvem svarer på hvilke spørsmål?
                - Inndeling i tema/forhold til hvem som presenterte hva?

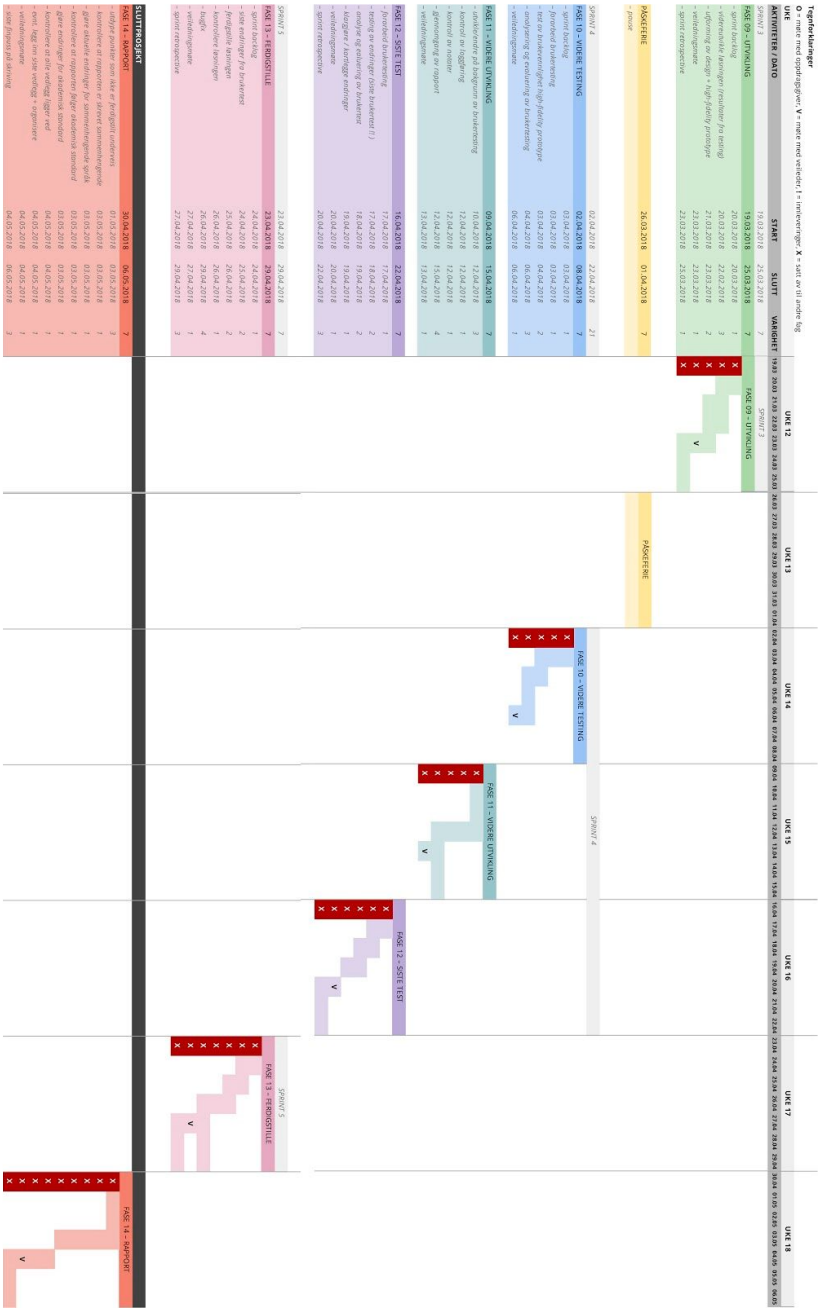
3.1.1 GANTT-diagram



Tegnforklaringer

O = møte med oppfølgingsfest, V = møte med veiledet, I = mineveringer, X = satt av til andre dag

UKE	ARTIFIETER / DATO	START	SLUTT	VARIGHET
	SPRINT 3	*19.02.2018*	*25.02.2018*	*7*
	FASE 09 – UTVIKLING	19.02.2018	25.02.2018	7
	– sprint backlog	20.02.2018	20.02.2018	1
	– informasjon til deltagere (resultater fra testing)	20.02.2018	22.02.2018	3
	– utforming av design + high-fidelity prototype	21.02.2018	23.02.2018	2
	– verifiseringsmøte	23.02.2018	23.02.2018	1
	– sprint retrospective	25.02.2018	25.02.2018	1
	PÅSKEFERIE	26.03.2018	01.04.2018	7
	Ferie			
	SPRINT 4	*02.04.2018*	*22.04.2018*	*21*
	FASE 10 – VIDERE TESTING	02.04.2018	08.04.2018	7
	– sprint backlog	03.04.2018	03.04.2018	1
	– forberede brukertesting	03.04.2018	03.04.2018	1
	– test av brukervennlighet high-fidelity prototype	03.04.2018	04.04.2018	2
	– analysering og evaluering av brukertesting	04.04.2018	06.04.2018	3
	– verifiseringsmøte	06.04.2018	06.04.2018	1
	FASE 11 – VIDERE UTVIKLING	09.04.2018	15.04.2018	7
	– implementere på bakgrunn av brukertesting	09.04.2018	11.04.2018	3
	– sprint backlog	12.04.2018	12.04.2018	1
	– kontroll av resultater	12.04.2018	12.04.2018	1
	– gjennomgang av rapport	12.04.2018	15.04.2018	4
	– verifiseringsmøte	15.04.2018	15.04.2018	1
	FASE 12 – SISTE TEST	16.04.2018	22.04.2018	7
	– forberede brukertesting	17.04.2018	17.04.2018	1
	– testing av endringer (siste brukertest !)	17.04.2018	18.04.2018	2
	– analyse og evaluering av brukertest	18.04.2018	19.04.2018	2
	– detaljere i kommende endringer	19.04.2018	19.04.2018	1
	– verifiseringsmøte	20.04.2018	20.04.2018	1
	– sprint retrospective	20.04.2018	22.04.2018	3
	SPRINT 5	*23.04.2018*	*29.04.2018*	*7*
	FASE 13 – FERDIGSTILLE	23.04.2018	29.04.2018	7
	– sprint backlog	24.04.2018	24.04.2018	1
	– siste endringer fra brukertest	24.04.2018	25.04.2018	2
	– ferdigstille løsningen	25.04.2018	26.04.2018	2
	– kontrollere løsningen	26.04.2018	26.04.2018	1
	– bugfix	26.04.2018	29.04.2018	4
	– verifiseringsmøte	27.04.2018	27.04.2018	1
	– sprint retrospective	27.04.2018	29.04.2018	3
	SLUTTFASE/SPRINT	30.04.2018	06.05.2018	3
	FASE 14 – RAPPORT	01.05.2018	03.05.2018	3
	– ferdigstillelse av rapporten (ikke er ferdigstilt i endtermin)	01.05.2018	03.05.2018	3
	– kontrollere at rapporten er skrevet sammenhengende	03.05.2018	03.05.2018	1
	– gjøre aktuelle endringer for sammenhengende språk	03.05.2018	03.05.2018	1
	– kontrollere at rapporten følger akademisk standard	03.05.2018	03.05.2018	1
	– gjøre endringer for akademisk standard	03.05.2018	03.05.2018	1
	– kontrollere om alle verifiseringer fungerer ved	04.05.2018	04.05.2018	1
	– event. legge inn siste verifiseringer + organisere	04.05.2018	04.05.2018	1
	– verifiseringsmøte	04.05.2018	04.05.2018	1
	– siste finpuss på innleveringen	04.05.2018	06.05.2018	3

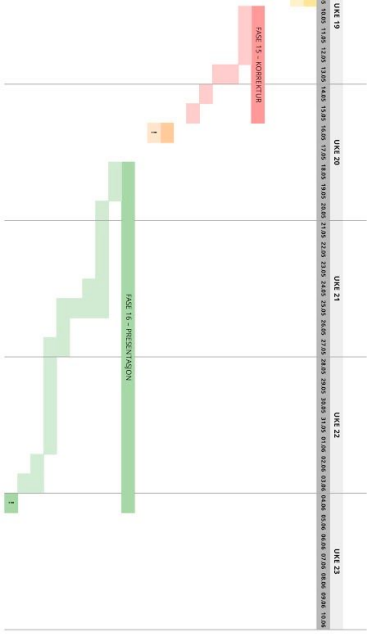


12 av 16

Tegnforklaringer

O = møte med oppdragsgiver; **V** = møte med veileder; **I** = innleveringer; **X** = satt av til andre fag

UKE					
AKTIVITETER / DATO	**START**	**SLUTT**	**VARIGHET**		
PAUSE	07.05.2018	09.05.2018	3	PAUSE	
– nullsime					
FASE 15 – KORREKTUR		**10.05.2018**	**15.05.2018**	**6**	
– *ikke korrektur*		10.05.2018	12.05.2018	3	
– *tekstkorrektur sjøfarmsklasen*		13.05.2018	13.05.2018	1	
– *finskriver sjøfarmsklasse*		13.05.2018	13.05.2018	1	
– *skriver ferdiskriftet*		14.05.2018	14.05.2018	1	
– *siste korrektur*		15.05.2018	15.05.2018	1	
INNLEVERING		**16.05.2018**	**16.05.2018**	**1**	
– *innleverring av bachelorprosjekt og rapport*					
FASE 16 – PRESENTASJON		**18.05.2018**	**04.06.2018**	**18**	
– *disposisjon av presentasjon*		18.05.2018	19.05.2018	2	
– *utarbeide presentasjon*		20.05.2018	25.05.2018	6	
– *utforme prosjektplakat*		24.05.2018	25.05.2018	2	
– *øve om og forbrite presentasjonen*		25.05.2018	25.05.2018	1	
– *forberede egen del*		25.05.2018	27.05.2018	3	
– *øve presentasjon*		25.05.2018	27.05.2018	3	
– *øve med gjennomgang av presentasjon*		27.05.2018	01.06.2018	4	
– *øve med gjennomgang av presentasjon*		02.06.2018	03.06.2018	2	
– *innøving av forskjellige teknikk emnemuligheter*		03.06.2018	03.06.2018	1	
– **PRESENTASJON**		**04.06.2018**	**04.06.2018**	**1**	



3.2 Valg av utviklingsmodell

Vi velger å bruke Scrum som utviklingsmodell. Alle gruppemedlemmene har god kjennskap til Scrum fra tidligere prosjekter, og en del av prosjektet vårt vil også innebære å ta et dypdykk i Scrum som utviklingsmetode. Jira bygges opp av en slik smidig metode, så for å få en god forståelse innad i gruppen, vil det være nødvendig å bruke Scrum under dette prosjektet.

3.3 Plan for statusmøter

Statusmøter vil bli holdt i etterkant av hver sprint, og i forkant av oppstart på ny sprint. Veiledermøter avholdes fast hver fredag, og møte med oppdragsgiver blir ca hver 14. dag.

4. Organisering av kvalitetssikring

4.1 Kvalitetssikring

For å kvalitetssikre løsningen vår, benytter vi oss av de nyeste versjonene av kodespråkene, og lager et utviklermiljø med JSLint for å se etter feil. Vi vil også benytte oss av JS Unit Testing for å sørge for at kodene fungerer som de skal.

4.2 Dokumentasjon (organisering og lagring)

Google Drive

Google Drive brukes til samskriving av dokumenter som ikke er kode-relaterte. Dette verktøyet er bra til bruk av backup, i tilfelle noe skulle skje med en av gruppemedlemmenes datamaskiner. Her loggfører vi også møter og bruk av timer på oppgaver i prosjektet. Dokumenter kan også deles med spesifikke interessenter.

Git og Github

Git, et DVCS verktøy (Distributed Version Control System) brukes til å dokumentere et prosjekt sin kodebase underveis. Git tilbyr muligheten til å vise en tidslinje med en oversikt over alle iterasjonene i en kodebase og forandringenes respektive beskrivelser. Samkjøring mellom Git og Github – en serverløsning for å hoste kildekode – gjør et godt utgangspunkt for kollaborasjon hvor alle gruppemedlemmer kan jobbe lokalt og laste opp og ned hverandres endringer.

Atlassian Jira

For å holde oversikt over planmål for enhver sprint så har vi valgt å bruke Atlassian Jira, det samme verktøyet som oppgaven omhandler. Vi skal bruke denne tjenesten primært til å holde en oversikt over vår backlog og hvordan den fordeles på oppgavens respektive sprinter. Vi ønsker også å bruke Jira for blant annet å bli kjent med Jira sine styrker og svakheter for å få ytterligere innsikt rundt utfordringene skildret i oppgavebeskrivelsen.

I tillegg så utmerker Jira seg for å være et verktøy som godt fasiliterer scrum som arbeidsmetode, en metode vi skal ta i bruk i løpet av dette prosjektet.

Facebook Messenger
Vi har en egen gruppechat på Facebook som vil fungere som en kommunikasjonskanal. Vi bruker Facebook Messenger fordi dette er en applikasjon som er lett tilgjengelig på både mobil og datamaskin. Messenger egner seg på å sende hyppige beskjeder som gjør det enkelt for enhver gruppemedlem å holde seg oppdatert

4.3 Risikoanalyse

	Ubetydelig	Alvorlig	Svært alvorlig
Svært usannsynlig		5	7 | 8
Sannsynlig	6	1 | 4	
Svært sannsynlig		2	3

Potensielle risikomomenter og potensielle tiltak

1. Vanskeligheter med å få avtalte møter med oppdragsgiver
    - Ha åpen dialog med oppdragsgiver, diskutere ulike tider og komme til enighet, inngå kompromiss om nødvendig; f.eks. ikke være fulltallig gruppe
    - Organisere møter etter oppdragsgiver sine ønsker, ikke motsatt
2. Feildisponering av tid
    - Re-evaluere og omprioritere oppgaver og fullføre de viktigste først
3. Tidsfrister opprettholdes ikke
    - Kom til bunns i problemet og diskutér ulike måter for å opprettholde neste frist
4. Manglende kompetanse
    - Søk kompetansen: Spør fagfolk, eller finn informasjon på Internett
5. Kommunikasjonssvikt blant gruppemedlemmer
    - Samle gruppen, diskutere problemet og deretter løse det via dialog
6. Sykdom på enkeltmedlemmer
    - De resterende gruppemedlemmene fordeler oppgavene seg imellom om mulig
7. Tap av data
    - Definere forhåndsregler for å forebygge denne risikoen; f.eks. ta i bruk mest mulig cloud baserte tjenester fra anerkjente aktører.
    - Let i arkivene og finn seneste versjon om tilgjengelig
8. Et gruppemedlem slutter
    - Finne kompromiss med oppdragsgiver rundt nedskalering av oppgaven slik at arbeidsmengde reflekterer antall gruppemedlemmer igjen

Vedlegg 1: Gruppeavtale

Gruppeavtale for bachelorprosjekt

Forventinger til hverandre og samarbeidet

- Jobb etter beste evne
- Ikke vær redd for å spørre om hjelp
- Husk å jobbe med flere oppgaver i parallell
- Delegér oppgaver ikke bare etter kompetanse, men også ønske om å lære
- Hvert enkelt medlem har selv ansvar for å ta back-up av eget arbeid

Forsentkomming eller ikke gjort tilegnet oppgave

- Åpen dialog om forsinkelser og uforutsette hendelser
    - Si ifra så fort du vet (via Messenger)
- Straff ved forsentkomming uten reell årsak (15 min)
    - Prikkesystem – Ved tre prikker skal vedkommende kjøpe forfriskninger/snacks til neste sammenkomst
- Straff ved ugjorte oppgaver uten reell årsak
    - Intervensjon/konfrontér vedkommende
    - Må lese en ekstra vitenskapelig artikkel eller få en ekstra tildelt oppgave
    - Alternativt kan en prikk tildeles

Kommunikasjon

- Hyppige oppdateringer på Messenger
- Høflig og sivil diskurs, men fortsatt ærlig og villig til å ytre seg om sine uenigheter
- Diplomatisk
- Ikke vær passiv-aggressiv og ikke insinuér negativitet, vær rett frem
- ALLTID bruk Messenger-gruppechatten, selv om det bare er en samtale mellom to medlemmer for å være mer transparent og holde hverandre oppdatert

Konfliktløsing

- Ikke forvirr konflikt med diskusjoner
- La alle få ordet og si sitt
    - legg alt annet til side for å komme til bunns i konflikten
    - bruk fornuft og ikke la konflikten vare lenger enn nødvendig
- Den minst partiske får rollen som diplomat

Avtale oppmøte- og samarbeidstid

- Veiledermøter hver fredag kl. 09.00
- Møtes hver dag vi ikke har valgfag som skiller gruppen
- Torsdag ettermiddag samles alle for å forberede til veiledermøte
- Fredag etter veiledermøte er det obligatorisk statusmøte for å se hva som er gjort og hva som skal gjøres den neste uken – siste frist for ferdigstilling er søndagen

Vedlegg 4: Gruppeavtale

Gruppeavtale for bachelorprosjekt

Forventinger til hverandre og samarbeidet

- Jobb etter beste evne
- Ikke vær redd for å spørre om hjelp
- Husk å jobbe med flere oppgaver i parallell
- Delegér oppgaver ikke bare etter kompetanse, men også ønske om å lære
- Hvert enkelt medlem har selv ansvar for å ta back-up av eget arbeid

Forsentkomming eller ikke gjort tilegnet oppgave

- Åpen dialog om forsinkelser og uforutsette hendelser
    - Si ifra så fort du vet (via Messenger)
- Straff ved forsentkomming uten reell årsak (15 min)
    - Prikkesystem – Ved tre prikker skal vedkommende kjøpe forfriskninger/snacks til neste sammenkomst
- Straff ved ugjorte oppgaver uten reell årsak
    - Intervensjon/konfrontér vedkommende
    - Må lese en ekstra vitenskapelig artikkel eller få en ekstra tildelt oppgave
    - Alternativt kan en prikk tildeles

Kommunikasjon

- Hyppige oppdateringer på Messenger
- Høflig og sivil diskurs, men fortsatt ærlig og villig til å ytre seg om sine uenigheter
- Diplomatisk
- Ikke vær passiv-aggressiv og ikke insinuér negativitet, vær rett frem
- ALLTID bruk Messenger-gruppechatten, selv om det bare er en samtale mellom to medlemmer for å være mer transparent og holde hverandre oppdatert

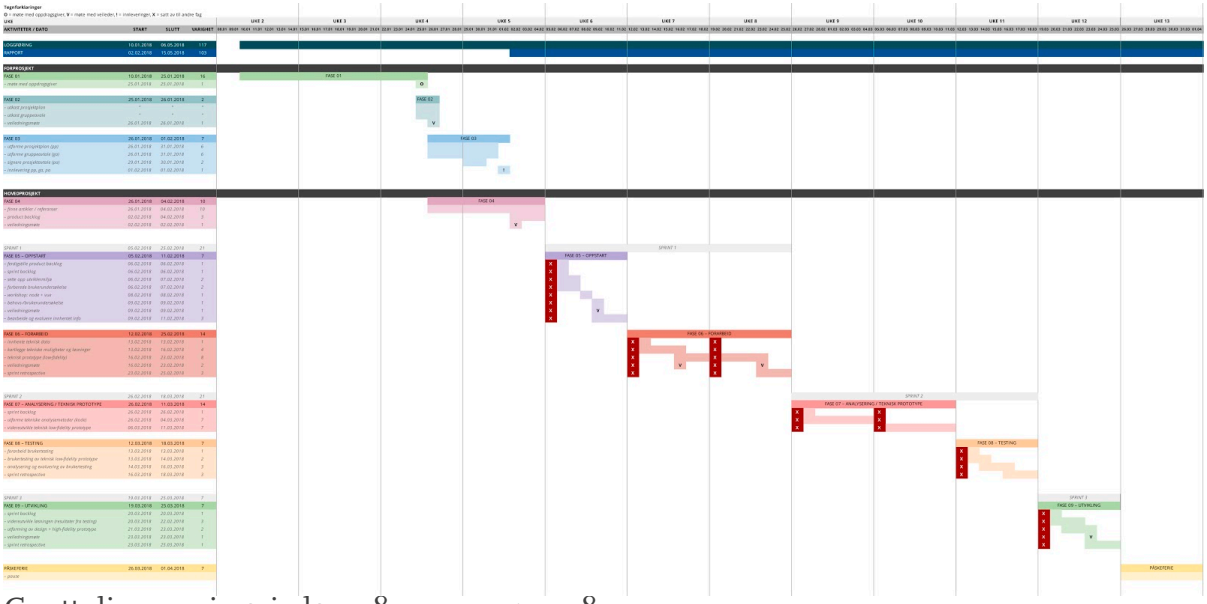
Konfliktløsing

- Ikke forvirr konflikt med diskusjoner
- La alle få ordet og si sitt
    - legg alt annet til side for å komme til bunns i konflikten
    - bruk fornuft og ikke la konflikten vare lenger enn nødvendig
- Den minst partiske får rollen som diplomat

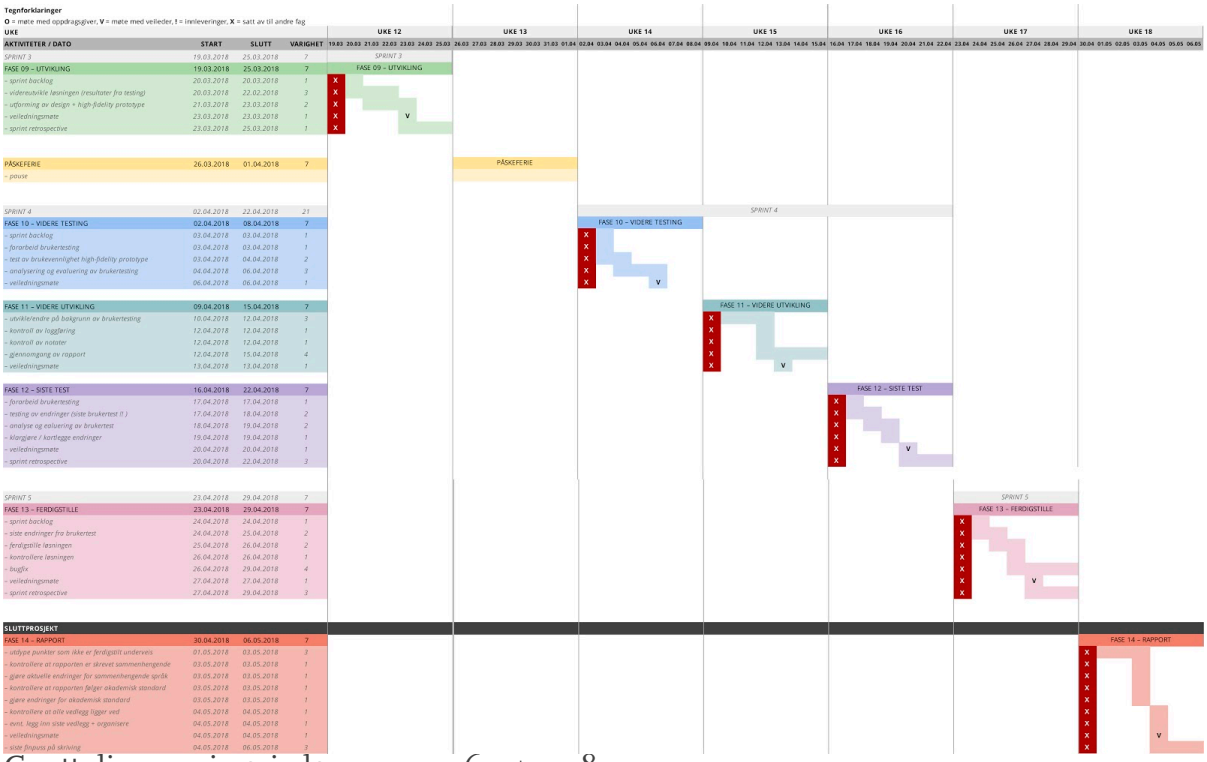
Avtale oppmøte- og samarbeidstid

- Veiledermøter hver fredag kl. 09.00
- Møtes hver dag vi ikke har valgfag som skiller gruppen
- Torsdag ettermiddag samles alle for å forberede til veiledermøte
- Fredag etter veiledermøte er det obligatorisk statusmøte for å se hva som er gjort og hva som skal gjøres den neste uken – siste frist for ferdigstilling er søndagen

Vedlegg 5: Gantt-diagram



Gantt-diagram i perioden 08.01–01.04.2018



Gantt-diagram i perioden 19.03–06.05.2018

Tegnforklaringer
O = møte med oppdragsgiver, **V** = møte med veileder, **I** = innleveringer, **X** = satt av til andre fag

UKE				UKE 19					UKE 20					UKE 21					UKE 22					UKE 23				
AKTIVITETER / DATO	**START**	**SLUTT**	**VARIGHET**	07.05	08.05	09.05	10.05	11.05	12.05	13.05	14.05	15.05	16.05	17.05	18.05	19.05	20.05	21.05	22.05	23.05	24.05	25.05	26.05	27.05	28.05	29.05	30.05	31.05	01.06	02.06	03.06	04.06	05.06	06.06	07.06	08.06	09.06	10.06
PAUSE	07.05.2018	09.05.2018	3	PAUSE																																		
– nullfilie																																						
FASE 15 – KORREKTUR	10.05.2018	15.05.2018	6				FASE 15 – KORREKTUR																															
– lese korrektur	10.05.2018	12.05.2018	3																																			
– kontrollere referanselisten	13.05.2018	13.05.2018	1																																			
– finpuse oppsett	13.05.2018	13.05.2018	1																																			
– skrive evalueringen	14.05.2018	14.05.2018	1																																			
– siste korrektur	15.05.2018	15.05.2018	1																																			
INNLEVERING	16.05.2018	16.05.2018	1																																			
– innlevering av bachelorprosjekt og rapport														I																								
FASE 16 – PRESENTASJON	18.05.2018	04.06.2018	18																																			
– disposisjon av presentasjon	18.05.2018	19.05.2018	2																																			
– utarbeide presentasjon	20.05.2018	25.05.2018	6																																			
– utforme prosjektplakat	24.05.2018	25.05.2018	2																																			
– dele inn og fordele presentasjonen	25.05.2018	25.05.2018	1																																			
– forberede egen del	25.05.2018	27.05.2018	3																																			
– individuell øving	27.05.2018	01.06.2018	6																																			
– felles gjennomkjøring av presentasjon	02.06.2018	03.06.2018	2																																			
– inndeling av forhåpentlige tekniske ansvarsområder	02.06.2018	02.06.2018	1																																			
– PRESENTASJON	04.06.2018	04.06.2018	1																																			

Gantt-diagram i perioden 07.05–10.06.2018

Vedlegg 6: Sprint retrospektiv

SPRINT RETROSPECTIVE

Samling av alle sprint retrospective

SPRINT 1: Retrospective

tirsdag 20.02.2018

Hva gikk bra under sprinten?

- samarbeid og kommunikasjon
- oppdatere hverandre på prosess underveis
- relativt flinke til å loggføre timer
- fikk til en samlet kodedag hvor vi har lært å bruke utviklermiljøet
- funnet flere aktuelle artikler
- fikk gjennomført to intervjuer der vi fikk innhentet mye informasjon som ble notert/arkivert slik at vi kan bruke dette videre
- gjennomført behovsanalyse fra intervjuene
- hatt jevnlige møter, både gruppen og med veileder
- lagde en arbeidsfordeling uke for uke - "sprintkalender oppdelt i uke"
- flinke til å planlegge og gjøre forarbeid

Hva gikk galt under sprinten?

- litt sent ute, blant annet ble kodedagen gjennomført litt senere i sprinten enn det som først var ønskelig
- ikke fått lest gjennom og notert fra artiklene som er innhentet
- ikke fått begynt på utkast på rapport
- ikke like flinke til å følge planen vi har satt opp, tidsfrister vi har satt selv blir ikke like godt fulgt opp eller overholdt

Hva kan vi gjøre annerledes for å sikre forbedringer til neste sprint?

- ha konkrete tidsfrister innad
- ha en person som følger opp tidsfristen for de andre (Susanne)
- sette opp ukeplan for hver uke (på søndagen før eller tidlig på mandag)

SPRINT 2: Retrospective

torsdag 15.03.2018

Hva gikk bra under sprinten?

- godt og balansert diskusjonsmiljø
- ideelt fordi folk tør å si sine egne meninger, det blir respektert og hørt, selv om man kan være uenige
- holdt tidsfristen (ikke stress på slutten)
- har individuelle oppgaver som gjøre at vi ikke må sitte sammen for å arbeide
- alltid fremgang pga individuell jobbing

Hva gikk galt under sprinten?

- demotiverende sprint pga utfordringer, vanskelig å finne gode løsninger som har gjort at utfordringene har føltes enda større og vanskeligere å overkomme
- fått følelsen av at prosessen har stagnert hver gang vi har møtt på en utfordring
- data fra oppdragsgiver er ufullstendig, derfor må vi sette av mye tid for å skrive algoritmer som fyller disse datahullene, dette er noe som kan anses som arbeid uten særlig verdi for endelig løsning – vår løsning skal hente data fra Jira (data skal ikke legges inn via løsningen)
- Data var ufullstendig i den forstand at det manglet verdier som er essensielle for vår løsning

Hva kan vi gjøre annerledes for å sikre forbedringer til neste sprint?

- Når vi kommer over utfordringer som i øyeblikket føles svært vanskelig å løse, bør vi nullstille oss: rydde bordet fullstendig, eller sette oss ned et sted med kun penn og papir (uten macer og telefoner) og drøfte utfordringen (kanskje med en 5-minutters pause først

Andre konkrete kommentarer?

- En periode har vi vært uten veileder - dette skjedde i en periode vi følte var vanskelig, der vi burde ha benyttet andre eksterne f.eks. til å komme med forslag på løsninger

SPRINT 3: Retrospective

onsdag 04.04.2018

Hva gikk bra under sprinten?

- videreutviklet prototypen
- testet prototypen
- fått litt mer fokus på å skrive i rapporten
- innad i gruppen har kommunikasjonen gått bra
- diskusjoner har vært fornuftige og konstruktive, vært gode for å komme videre i prosessen
- synspunkter har blitt forstått og respektert

Hva gikk galt under sprinten?

- ikke fått kodet like mye som planlagt

Hva kan vi gjøre annerledes for å sikre forbedringer til neste sprint?

- mer parallell jobbing med koding og rapport
- (Martine og Silje) jobbe mer regelmessig og sammenhengende
- følge Jira backlog mer slavisk

Andre konkrete kommentarer?

- kort sprint, der påskeferien gjorde at den ble veldig splittet opp
- andre emner (valgemner) skaper støy og tar mye tid og energi som i en viss grad påvirker arbeidet med bacheloren

SPRINT 4: Retrospective

fredag 27.04.2018

Hva gikk bra under sprinten?

- Alle kodet og bidro
- Kom i mål av hva som var realistisk med tanke på koding
- Poker Planning ga greit tidsestimat på oppgavene
-

Hva gikk galt under sprinten?

- Noen elementer tok lenger tid å kode enn planlagt, samt at det var vanskeligere enn opprinnelig tenkt
- Det ble litt mye "stress-koding", slik at kodedelen ikke ble så gøy

Hva kan vi gjøre annerledes for å sikre forbedringer til neste sprint?

- Spørre om mer hjelp (?) på de oppgavene der vi slet
- Har man et problem i koding der samtalen er langvarig, er det bedre å ringe

Andre konkrete kommentarer?

- Burde tatt koding og rapport mer parallelt (både i denne sprinten og tidligere)
    - Veldig blokkbasert der vi bytter hovedfokus hver uke, kan gjøre arbeidsflyten litt tregere

Vedlegg 7: Sprint reviews

Sprint reviews

sammen med oppdragsgiver, Terje Krogstad

Sprint review 1: 23. februar 2018

Første sprint review ble inkorporert i et brukerintervju, som senere ble transkribert. Dette kan ikke offentliggjøres etter samtykkeskjema.

Sprint review 2: 20. mars 2018

User stories kan gå over flere sprinter, og derfor er det greit å angi storypoints der. Men det er kun der man kan angi storypoints. Bruk av storypoints varierer veldig fra prosjekt til prosjekt.

Vi har poker-planning sesjoner der vi estimerer storypoints, og der kan det sprike veldig. Eller vi kan være veldig samkjørte. Så det er en fin måte å få en felles forståelse på, samt at vi er enige om størrelsen på oppgaven.

Ser på første hi-fi:

- Kunne hatt fargekoding på belastning

Ser på ny hi-fi:

- Likte at vi droppet "done"
- Synes det var bra vi flyttet progressbar til venstre (prosjekt)
- Får ikke noe sprint-indikasjon på ressurspersoner (ser uker, men ikke sprint)
- Kanskje ikke radiobuttons, men checkbokser (på ressurspersoner)
- Tooltip er interessant
- Her kunne dere iterert på dataintervall, kunne se belastningen på vedkommende i de neste 3 ukene, for eksempel (på storypoints)
- Vil ha tilleggsinformasjon (fargekoding) for å se hvilke sprinter det er overbelastning på
- Kan ha en prikk eller indikator for å se overbelastning
    - Belastning på sprinter – varseltrekant
    - tar for lang tid med bare tall
- Ha mulighet til å se belastning ved å velge en checkboks på siden
- Bruke farge med ulik intensitet for å markere overbelastning
- Vil anbefale å gå for horisontal visning i storypoints-siden kun på grunn av tilpasning til mindre enheter
- Den vil mest bli brukt internt, men er absolutt relevant å kunne bruke som et verktøy for å vise tidsestimat og allokering ut mot kunder
- Ressurspersoner - HUSK knapp se alle
- Den bortover er bedre med responsivitet

- Velge ut 4 prosjekter som vi manipulerer data til

Trenger subtasks på brukerhistorier for å få fram koding, få kommet ordentlig igang med det så fort som mulig

Plukk ut noen sprinter som man setter datoer på. Så kan oppgavens sluttdato bli satt til slutten av sprinten som den er i. Må bli flinkere til å definere user stories med et tidsestimat på. Prøver å ha vanlige tasks og user stories så små som mulig for å gjøre det håndterbart innenfor en sprint.

100-poengere kommer ikke inn i Jiraen vår, da setter vi oss heller ned å plukker oppgaven fra hverandre til mindre oppgaver. Det gjelder også oppgaver på 80 og 40, for det er stories som vi ikke ønsker å håndtere, det er for store oppgaver og bør deles opp.

Appen planning poker løser det med storypoints-størrelsen, og der brukes fibonacci-tallrekken. Man velger et tall i appen og viser det til de andre. Så ser man om de har valgt å se en lik størrelse på oppgaven. Så blir man enige om et scope, og tar en ny runde på storypoints.

En utfordring er user stories som går over flere sprinter. Det kan være et problem som vår løsning kan oppfatte og løse.

Noe viktig å se på når man skal definere oppgaver, er å aldri la dem være større enn at man kan løse dem på en dag. Blir det større enn det, bør de deles opp i flere mindre oppgaver.

Sprint review 3: 6.april 2018

God hjelp av pokerplanning - prøv det i neste sprint
Spikre sprinten etter pokerplanningen - prøv å ha pokerplanning på mandag eller tirsdag
Ta høyde for storypoints - legg dere lavt, lær av tidligere "feil", altså beregning av hva man klarer
Kom med reelle tiltak for hvordan man skal gjøre det bedre i neste sprint, ikke bare "*vi må planlegge bedre, vi må prioritere bedre*". Konkrete tiltak må komme etter pokerplanning.

Atomic design er verdt å lese seg opp på.

Når man lenker eksternt, så åpne i nytt vindu, men når man lenker internt så åpner man i samme fane (sjekk research på det)

Kan være plagsomt med dropdown/tooltip (på ressurspersoner)

Kan man slå sammen linjene til en linje (på ressurspersoner)? Argumentet er at to-do-linjen bør være borte etter at sprinten er borte

En kombinasjon man kan gjøre er at dersom 5 av 7 oppgaver er gjort, så kan 5/7 av linjen være skravert, men resten er ikke skravert. Gjør det også mer visuelt synlig.

Kanskje oppgavene som er opplyste skal flyttes innerst (slik at man ser hvor store oppgaver man har i det spesifikke prosjektet) (på ressursbelastning)

De som ikke er med på et prosjekt når man klikker inn på et (på ressursbelastning), kan flyttes nederst. Motargumentet der er at det kan være forvirrende for brukeren dersom personer plutselig bytter plass.

Liker "vis belastning for tidsrom" der man viser uker. Kan være relevant å ha den opp til 6 uker.

Kan beregne neste sprint ut ifra lengden på forrige sprint, da de pleier å være like lange per prosjekt.

Ser at denne add-onen kan tvinge de til å bli mer strukturerte (det er positivt)

Sprint review 4: 27. april 2018

PROSJEKTOVERSIKT
- Husk link til eksterne sprinter i Jira

RESSURSPERSONER
- Ikke ha overskrifter repeterende
- I en sprint backlog er det motsatt i Jira
- Vil ha in-progress først
- Legge inn flere oppgaver på personer
- Fikse filtrering

RESSURSBELASTNING
- Filtrering på et prosjekt: Flytt oppgavene til venstre
- json-objekter må legges til

Vedlegg 8: Sprintplan 1

man. 12.02	tir. 13.02	ons. 14.02	tor. 15.02	fre. 16.02	lør. 17.02	søn. 18.02	
	SKYPEMØTE – kl 11.00 med Håvard + transkribering	transkribering møte med Håvard					MARTINE
	Lese bachelorbok + ta notater	Lese bachelorbok + ta notater	Lese bachelorbok + ta notater	Lese bachelorbok + ta notater	Lese bachelorbok + ta notater	FERDIG: Lese bachelorbok + ta notater + renskrive notatene	
		RAPPORT: – teori om intervjuprosess (bruker, målgruppe, behov) – ~~teori om behovsanalyse~~ - gruppens fremgangsmåte – utkast til rapport	RAPPORT: – teori om intervjuprosess (bruker, målgruppe, behov) - gruppens fremgangsmåte – utkast til rapport				
	utviklermiljø	utviklermiljø	utviklermiljø	utviklermiljø			AUDUN
		Wireframes					
		Purre på Terje om data ikke er mottatt					SUSANNE
		Wireframes	EndNote	Wireframes/ Brukerreiser&Scenario			
		Notater fra grafikkbok	Notater fra grafikkbok	Starte å se på Terjes data, hvis tid			
			Sende påmindelse til Eivind om å se på rapportdisposisjon				
	transkribering møte med Terje	transkribering møte med Terje	transkribering møte med Terje	transkribering møte med Terje			SILJE
		~~Wireframes~~		Wireframes/ Brukerreiser&Scenario			
	se video om node + vue	se video om node + vue	se video om node + vue	se video om node + vue	se video om node + vue	se video om node + vue	ALLE
	Motta data fra Terje	Se på rapportdisposisjonen	Se på 2-3 andre rapporter, noter ned punkter som er relevante i forhold til rapportdisposisjon				
		Lage punktliste over hva vi skal bruke kodedagen (tirsdag) til					
man. 19.02	**tir. 20.02**	**ons. 21.02**	**tor. 22.02**	**fre. 23.02**	**lør. 24.02**	**søn. 25.02**	
		FERDIG:	SKIDAG				
		behovsanalyse: kritiske punkter om intervju/spørsmål/prosess					MARTINE
		RAPPORT: – teori om intervjuprosess bruker, målgruppe, behov) - gruppens fremgangsmåte – utkast til rapport					
Teknisk prototype	utviklermiljø på alle maskiner	utviklermiljø på alle maskiner					AUDUN
		WORD-dokument med EndNote					SUSANNE
							SILJE
	Kodedag i Vue + Node	rapportdisposisjon					ALLE
		Sprint retrospective + forberede til sprint review					
		Ferdigstille wireframes, behovsanalyse (PACT/fra bok), brukerhistorier (personas og scenarier?)					

Vedlegg 9: Sprintplan 2

man. 26.02	tir. 27.02	ons. 28.02	tor. 01.03	fre. 02.03	lør. 03.03	søn. 04.03	
	Notere fra bok?	Notere fra bok?	Kodedag + EndNote	Forarbeid til hi-fi prototype? Rapportskriving!!			MARTINE
	Video om Vue	Video om Vue					
	Teknisk prototype	Teknisk prototype					AUDUN
	Tegne wireframes	Samle referanser i EndNote					SUSANNE
	Tegne wireframes	Lese artikler					SILJE
	Ferdigstille lo-fi wireframes (Papirskisser) til en samlet versjon	Skrive rapport					ALLE
		Artikkel + referanser = notater	Sende lo-fi /wireframes til Terje?				
man. 05.03	**tir. 06.03**	**ons. 07.03**	**tor. 08.03**	**fre. 09.03**	**lør. 10.03**	**søn. 11.03**	
		Klargjøring brukertest 1	Kodedag + Klargjøring brukertest 1	Brukertest 1			MARTINE
	Teknisk prototype	Teknisk prototype					AUDUN
		Klargjøring brukertest 1					SUSANNE
		Klargjøring brukertest 1					SILJE
	Lage hi-fi prototype	Lage hi-fi prototype					ALLE
		Artikkel + referanser = notater	Ferdigstille hi-fi prototype				
man. 12.03	**tir. 13.03**	**ons. 14.03**	**tor. 15.03**	**fre. 16.03**	**lør. 17.03**	**søn. 18.03**	
	Evaluering brukertest 1	Rapportskriving	Kodedag + se på design/iterasjon av hifi	Veiledermøte + Ferdigstillelse sprint 2 + Sprint retrospektiv 2			MARTINE
	Teknisk prototype	Teknisk prototype					AUDUN
	Evaluering brukertest 1	Rapportskriving					SUSANNE
	Evaluering brukertest 1	Rapportskriving					SILJE
							ALLE
	tir. 20.03						
	Sprint review 2 med Escio kl. 10.30 til 12.30						

Vedlegg 10: Sprintplan 3

man. 19.03	tir. 20.03	ons. 21.03	tor. 22.03	fre. 23.03	lør. 24.03	søn. 25.03	
		KODEDAG					MARTINE
							AUDUN
							SUSANNE
							SILJE
		Lage designmal / designregler (med notater for bruk i rapport)	Se på tidligere pensumbøker og skrive rapport	Veiledningsmøte			ALLE
		Implementere designmal	Skrivekveld	Ferdigstille hi-fi klar til brukertest			
man. 26.03	**tir. 27.03**	**ons. 28.03**	**tor. 29.03**	**fre. 30.03**	**lør. 31.03**	**søn. 01.04**	
	Påskeferie – kos og avslapning!						
	RAPPORT: Scrum						MARTINE
	RAPPORT: Kode / teknologi						AUDUN
	RAPPORT: Scrum						SUSANNE
	RAPPORT: Universell utforming						SILJE
	Ferdigstille hi-fi klar til brukertest						ALLE
man. 02.04	**tir. 03.04**	**ons. 04.04**	**tor. 05.04**	**fre. 06.04**	**lør. 07.04**	**søn. 08.04**	
	Brukertest Escio		Implementere grid-visning av uker				MARTINE
			Implementere progressbar til hvert prosjekt				
			Sprint backlog				AUDUN
			Sprint backlog				SUSANNE
			Sprint backlog				SILJE
		Rapportskriving	Sprint retrospektiv og ferdigstillelse av sprint 3	Veiledningsmøte + Oppdragsgiver møte Escio om sprint 3 + 4			ALLE
	Analysere brukertest 2						

Vedlegg 11: Sprintplan 4

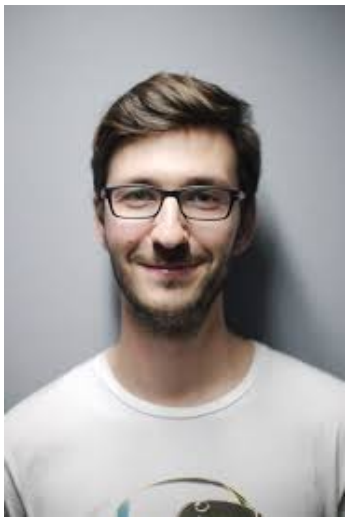
man. 09.04	tir. 10.04	ons. 11.04	tor. 12.04	fre. 13.04	lør. 14.04	søn. 15.04	
	Skrive om prototype	Skrive om prototype	Skrive om designprinsipper, PACT-analyse, Usecases	Skrive om designprinsipper, PACT-analyse, Usecases			MARTINE
Rapport	Rapport	Rapport	Rapport	Rapport + Veiledermøte			AUDUN
							SUSANNE
							SILJE
					Rapport	Rapport	ALLE
man. 16.04	**tir. 17.04**	**ons. 18.04**	**tor. 19.04**	**fre. 20.04**	**lør. 21.04**	**søn. 22.04**	
Fullføre PACT i teoridelen, skrive inn PACT-analysen i analysedelen, skrive om Use Cases	Fullføre PACT i teoridelen, skrive inn PACT-analysen i analysedelen, skrive om Use Cases	skrive inn PACT-analysen i analysedelen, skrive om Use Cases	Skrive om designprinsipper (Don Norman)	Koding			MARTINE
	Koding	Koding	Koding				AUDUN
							SUSANNE
							SILJE
				Veiledermøte + Statusmøte med Terje + Koding			ALLE
man. 23.04	**tir. 24.04**	**ons. 25.04**	**tor. 26.04**	**fre. 27.04**	**lør. 28.04**	**søn. 29.04**	
Koding	Koding	Koding	Sprint retrospektiv og ferdigstillelse sprint 4	Veiledermøte + Sprint review 4			MARTINE
							AUDUN
							SUSANNE
							SILJE
							ALLE

Vedlegg 12: Plan over sluttfase

man. 30.04	tir. 01.05	ons. 02.05	tor. 03.05	fre. 04.05	lør. 05.05	søn. 06.05	
1. Prosjektbeskrivelse	Leveringskrav + utskrift av oppgave (sende mail til Iriskeby Media) 2. Prosjektorganisering	2. Universell utforming	3. Risikoanalyse → flyttet til del 1	4. Prototype	*Ferdigstille alle oppgaver*	*Ferdigstille alle oppgaver*	MARTINE
Egen forside som forklarer gruppenavn + bakgrunn kap. 1	2. Samhandlingsverktøy (in progress)	2. Informasjonsarkitektur	2. Webteknologier	5. Implementering	5. Implementering	5. Implementering	AUDUN
1. Øvrige roller + bakgrunn og kompetanse	2. Prosjektstyringsverktøy	2. Designprinsipper	3. Brukerinnsikt: Målgruppe- og konkurrentanalyse	4. Prototype	*Ferdigstille alle oppgaver*	*Ferdigstille alle oppgaver*	SUSANNE
Finne krav på forside + lage forside (in progress)	2. Mental modell	2. Universell utforming + Personas og scenarier	2. Brukertesting: evaluering	4. Prototype	3. Brukerinnsikt: Brukerintervju / analyse	*Ferdigstille alle oppgaver*	SILJE
Rapport	Rapport	Rapport	Rapport	Rapport + Veiledermøte	Rapport	Rapport	ALLE
Levere kode til Escio							
man. 07.05	**tir. 08.05**	**ons. 09.05**	**tor. 10.05**	**fre. 11.05**	**lør. 12.05**	**søn. 13.05**	
6. Endelig løsning	Konklusjon + 1.8 Bakgrunn og kompetanse	Universell utforming (Lover) + 5. Implementering Frond-End	Lage forslag til responsiv design i Xd + 5. Implementering Frond-End	Korrekturlesing (spesielt kap. 4 om brukstest)			MARTINE
7. Videreutvikling	Implementering + Videreutvikling + Evaluering av Node (6. endelig løsning)	Ferdigstillelse web+infoarkitektur kap.2	Site Map + Korrekturlesing	Ferdigstille kap. 5 + Avsnitt i kap. 6 + Skrive ferdig kap. 7 + Korrekturlesing + Printe ut			AUDUN
6. Endelig løsning + Konklusjon	Konklusjon + 1.9 Terminologi	*4.2.7 Design prototype + Sette inn vedlegg: Logg, prototype, skisser, gantt+++ Vedlegg kap. 2, nr.xx osv*	4.2 Idéfase + 4. bruksmønster + 4. Ikoner + SiteMap +Korrektur + Word	Skrive om Site Map kap.4 + Fylle ut alle innledninger og sammendrag + Sjekke at problembeskrivelse er nevnt overalt + Korrekturlesing + Printe ut	Lese igjennom hele rapporten	Lese igjennom hele rapporten	SUSANNE
7. Videreutvikling	Teori i kap.2 om intervjuf! + 1.3.6 Gjenstående arbeid	Kap. 7	4.2 Idéfase + 4. bruksmønster + 4. Ikoner + SiteMap + 2. Gestaltprinspier + Korrekturlesing	Skrive om Site Map kap.4 + Fylle ut alle innledninger og sammendrag + Sjekke at problembeskrivelse er nevnt overalt + Korrekturlesing + Printe ut			SILJE
6. Endelig løsning		Ferdigstille rapport	Lese igjennom hele rapporten	Veiledermøte			ALLE
	Martine back in G-town						
Publiseringsavtale biblioteket?							
man. 14.05	**tir. 15.05**	**ons. 16.05**	**tor. 17.05**	**fre. 18.05**	**lør. 19.05**	**søn. 20.05**	
							MARTINE
Korrekturlesing + skrive sammendrag	Ferdigstillelse av rapporten	LEVERE Bachelor kl. 16.00 + lunsj	NONONONONONONONONONONONONOSpis digg mat, drikk masse sjampis og kos dere<3<3<3				AUDUN
							SUSANNE
							SILJE
Lese igjennom hele rapporten							ALLE
	Martine back in G-town						

Vedlegg 13: Personas

Primær personas



Navn: Erik Haugland
Alder: 33
Bosted: Nordbyen, Gjøvik
Utdannelse: Medie- og dokumentasjonsvitenskap, Master på UiT
Sivilstatus: Forlovet med Iselin Ludvigsen (31)

(Bilde hentet fra: https://www.pexels.com/photo/adult-beard-boy-casual-220453/)

Erik er prosjektleder i bedriften. Han har alltid hatt en interesse for teknologi og nye trender i miljøet. Han er nylig ansatt, men har allerede funnet seg godt til rette. I tillegg til å være lidenskapelig opptatt av teknologiske duppeditter, er han glad i å holde seg i form ved hjelp av langrenn og løping.

Mål:
Erik har nettopp fått et nytt oppdrag for bedriften. Utfordringen er at han allerede er leder for tre andre pågående prosjekter. Målet er å holde oversikten over alle, slik at ingen møter krasjer eller at arbeidsoppgaver for teamet overskrider hverandre.

Scenario:
Erik har et oppstartsmøte med bedriften i mars, og kunden ønsker at nettsiden skal være ferdig innen utgangen av mai. Når han har møte med teamet, finner han ut av to av fem ansatte allerede er for opptatt med de andre prosjektene. Han må derfor finne en måte å allokere ressurspersonene på.

Sekundær personas

Navn: Nina Fredriksen
Alder: 37
Bosted: Hamar
Utdannelse: Industriell økonomi og teknologiledelse, Master på NTNU i Trondheim
Sivilstatus: Gift med Andreas Fredriksen

(Bilde hentet fra: https://www.pexels.com/photo/close-up-photography-of-a-woman-wearing-formal-coat-785667/)

Nina er en teknologi-entusiast, til tross for at hun ikke er like hands-on som mange av sine kolleger, egner hun seg godt til å holde et overordnet blikk over IT-landskapet. Hun vet hva som er in, og liker å overse om prosjekter sin utvikling promoterer hva hun anser som fremtidssikre fremgangsmåter. På fritiden liker hun å dra ut på eksklusive restauranter med sine venninner.

Mål:
Hun er opptatt av teammedlemmene sitt velvære og ønsker at alle skal kunne trives samtidig som de jobber etter beste evne. Hun må få et oversiktlig overblikk over alle aspekter av bedriften – ansatte, prosjekter og oppdragsgivere – for å kunne utføre sin lederrolle godt.

Scenario:
Nina, som representerer et viktig kommunikasjonsledd i sitt IT-konsulentselskap, har ansvaret for å føre en behagelig og produktiv dialog med sine oppdragsgivere. Hun må sørge for at oppdragsgivere respekterer deadlines og arbeidsbelastningen på utviklerne. Hun trenger derfor et verktøy som gjør at man kan presentere data fra Jira på en enkel og delikat måte, slik at oppdragsgivere kan få et klart innblikk i utviklerne sin hverdag, slik at de ikke presser unødvendig på deadlines.

Sekundær personas (ansatt i bedrift)

Navn: Kenneth Larsen
Alder: 24
Bosted: Kopperud, Gjøvik
Utdannelse: Webutvikling, Bachelor på NTNU i Gjøvik
Sivilstatus: Enslig

(Bilde hentet fra: https://www.pexels.com/photo/adult-attractive-blue-casual-415326/)

Kenneth er nyutdannet, og har nettopp tatt en bachelorgrad i Webutvikling. På fritiden programmerer han mye, og har generelt stor interesse for ny teknologi. Han er introvert, og derfor noe konfliktsky. På fritiden liker han å konsumere media i form av YouTube-videoer, i tillegg til at han følger pop-kultur-verdenen og gleder seg til de nyeste kino-lanseringene.

Mål:
Kenneth vil ha en god oversikt over sin egen arbeidsdag og når oppgaver skal gjøres. Han liker å se hele teamet sin fremgang for å kunne føle en tilhørighet og se prosjektet i sin helhet.

Scenario:
Det er fredag morgen og Kenneth ser på planen at han har for mye arbeid igjen til å kunne fullføre det denne uken. Han er ikke særlig utadvendt og blir ubekvem av tanken på å si fra til Scrum master ettersom at han er konfliktsky og ikke vil opptre som en negativ arbeider. Kenneth håper at det eksisterer verktøy som bedre avdekker urettferdige eller tunge arbeidsbelastninger.

Vedlegg 14: Brukerhistorier

Som bruker av løsningen ønsker jeg å se belastningen på hver ressursperson slik at jeg kan analysere allokering

Som bruker av løsningen ønsker jeg å se belastningen på hver sprint slik at vi får oversikt over sprintbelastningen

Som bruker av løsningen ønsker jeg å se belastningen på alle prosjekter slik at jeg kan sammenligne ressursbruken

Vedlegg 15: Brukerreiser

> Tildeling av oppgaver

1. Prosjektleder har definert nye oppgaver i et prosjekt som må gjennomføres.
2. Prosjektleder er noe motvillig til å tildele nye oppgaver til sine medarbeidere da de kan være overbelastet fra andre prosjekter.
3. Prosjektleder vil derfor undersøke hvilke potensielle medarbeidere han har til disposisjon med kapasiteten til å utføre de nye oppgavene.
4. Prosjektleder åpner applikasjonen fra Jira sitt kontrollpanel, og navigerer seg til en underside som viser arbeidere sine storypoints-totaler.
5. Prosjektleder identifiserer en medarbeider med få storypoints, og vil undersøke nærmere hvordan arbeideren sine oppgaver er fordelt over tidsrom.
6. Prosjektleder navigerer seg til en visning for oppgaveoversikt, og bekrefter at medarbeideren ikke har andre oppgaver med snar frist, og har derfor kapasitet til å utføre oppgaven.
7. Oppgaven blir tildelt medarbeideren med kapasitet, og prosjektleder er nå tryggere på en jevn flyt i prosjektet.

> Sprint planlegging

1. Prosjektleder skal bevege prosjektarbeidet over i ny sprint.
2. Prosjektlederen henter derfor oppgaver fra backlog og inn i ny aktiv sprint.
3. Prosjektleder vil se om sprint risikerer å bli større enn hva prosjekt-teamet har kapasitet til å gjennomføre.
4. Prosjektleder åpner applikasjonen fra Jira sitt kontrollpanel, og navigerer seg til en underside som viser en prosjektoversikt.
5. Applikasjonen varsler i grensesnittet om hvilke sprinter som kan anses som risikable.
6. Prosjektleder ser at sprinten er tungt belastet, og handler derfor hensiktsmessig i form av evaluere hvilke oppgaver som kan flyttes tilbake til backlog.

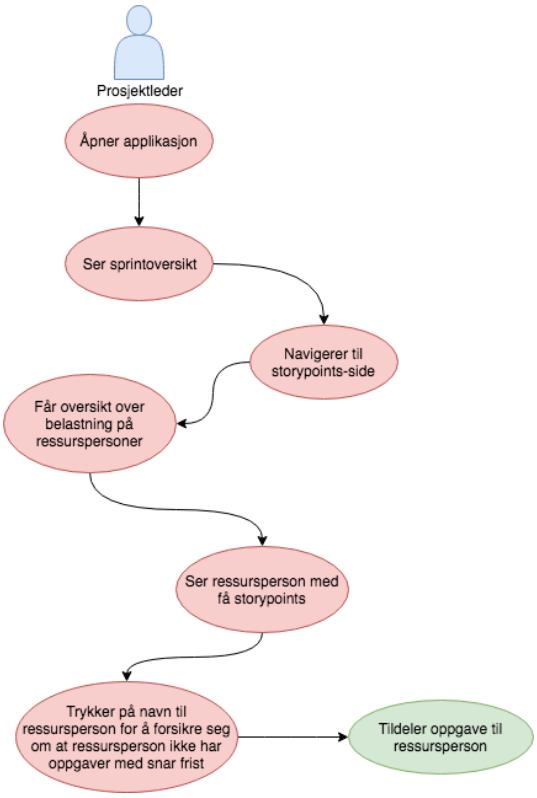
> Prosjektoversikt

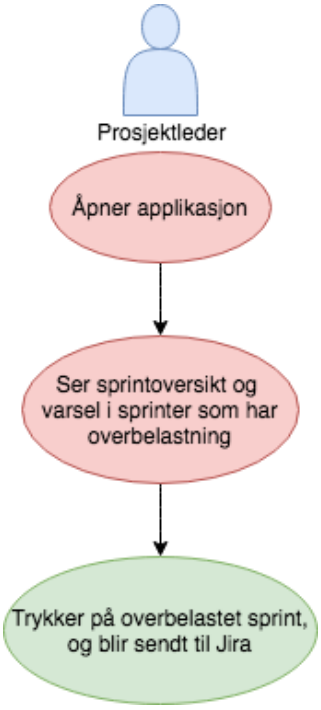
1. Prosjektleder opplever å ha et større prosjekt-team enn nødvendig, og ser verdi i å plassere noen av sine medarbeidere til andre prosjekter for bedre allokering.
2. Prosjektleder vil derfor se nærmere ønsker å hvordan framgang i sitt prosjekt målt mot andre prosjekter, slik at det kommer tydelig fram hvilke prosjekter det er aktuelt å flytte medarbeidere til
3. Prosjektleder navigerer seg til prosjektoversikt
4. Prosjektleder identifiserer de aktuelle prosjektene for ny allokering. Prosjektleder ser på informasjonsgrafikk som viser hvor mye arbeid som mangler vist med en storypoints total
5. Prosjektleder flytter aktuelle medarbeidere til ønskede prosjekter for å sikre ny og bedre allokering.

> Omrokkering av oppgaver

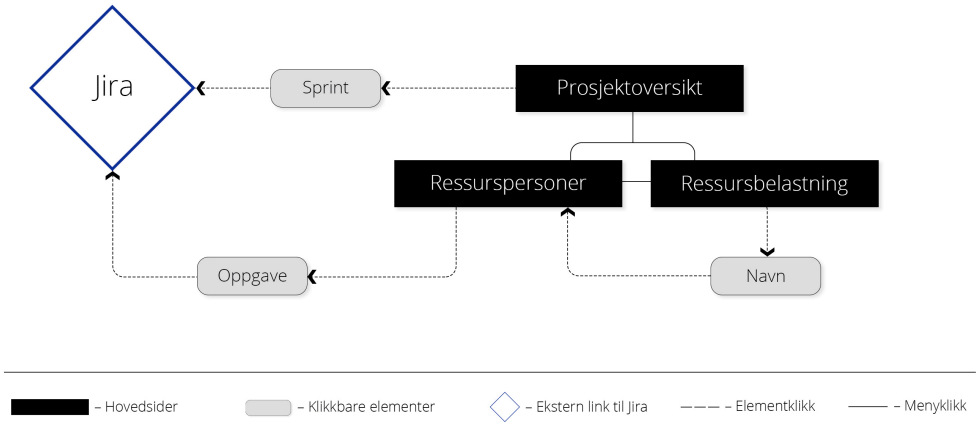
1. Prosjektleder identifisert en konflikt i allokeringen. To medarbeidere er kritisk belastet, og derfor må handling tas.
2. Prosjektleder navigerer seg til oppgaveoversikt og identifiserer oppgaver med status to-do. Prosjektleder trykker på ønsket oppgave, og blir presentert en liste arbeidere med kompetanse som egner seg for å løse oppgaven.
3. Prosjektleder tilegner oppgave til en egnet medarbeider.
4. Oppgaven blir utført av valgt ansatt, og prosjektleder har vært suksessfull i med den nye allokeringen.

Vedlegg 16: Bruksmønster

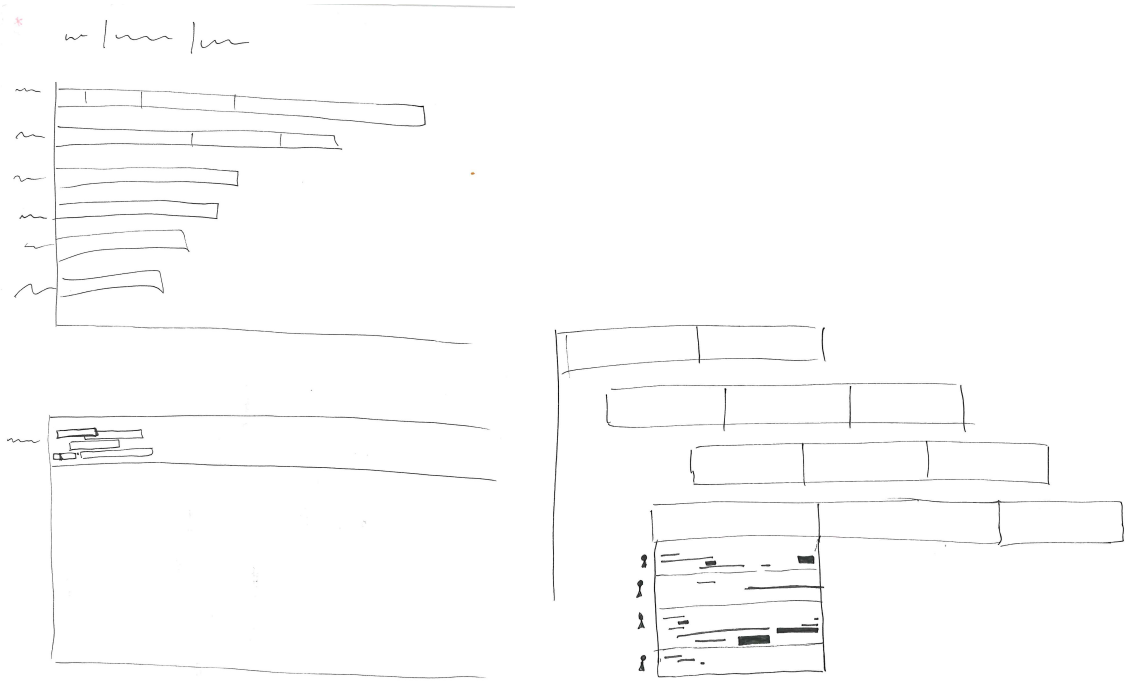




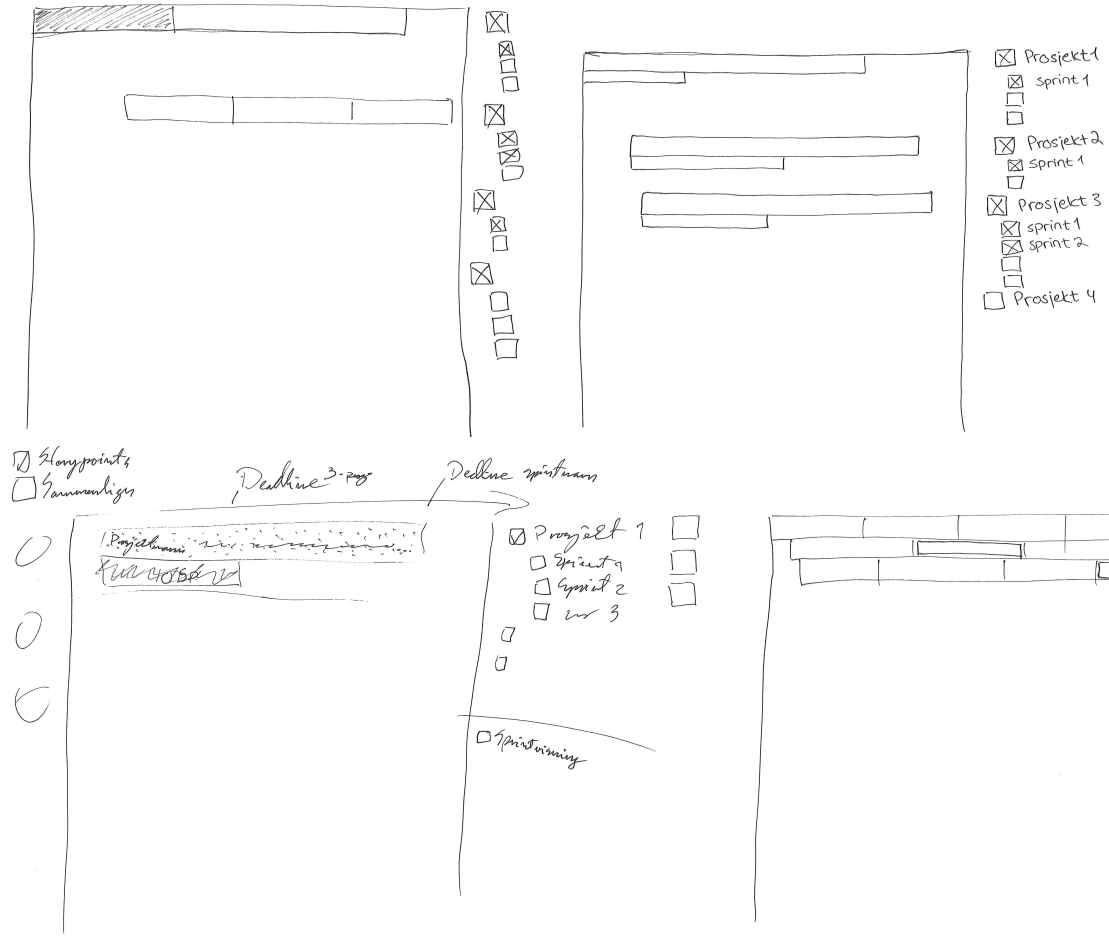
Vedlegg 17: Nettsidekart

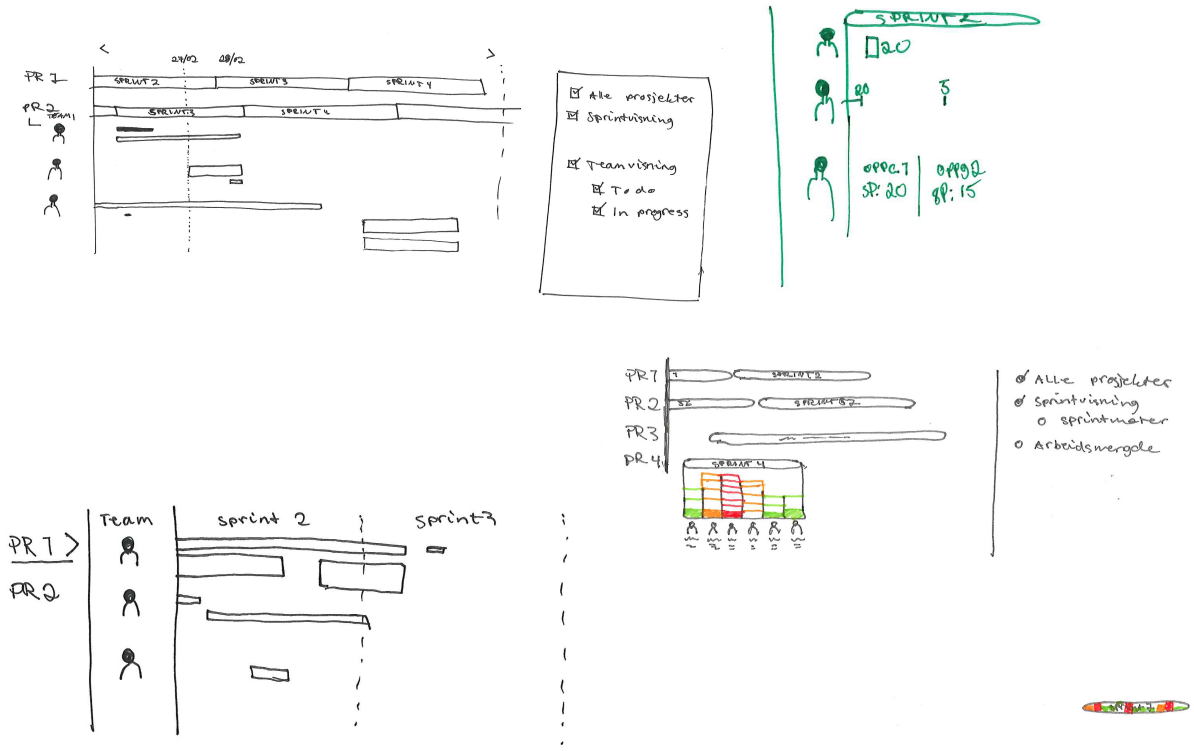


Vedlegg 18: Skisser

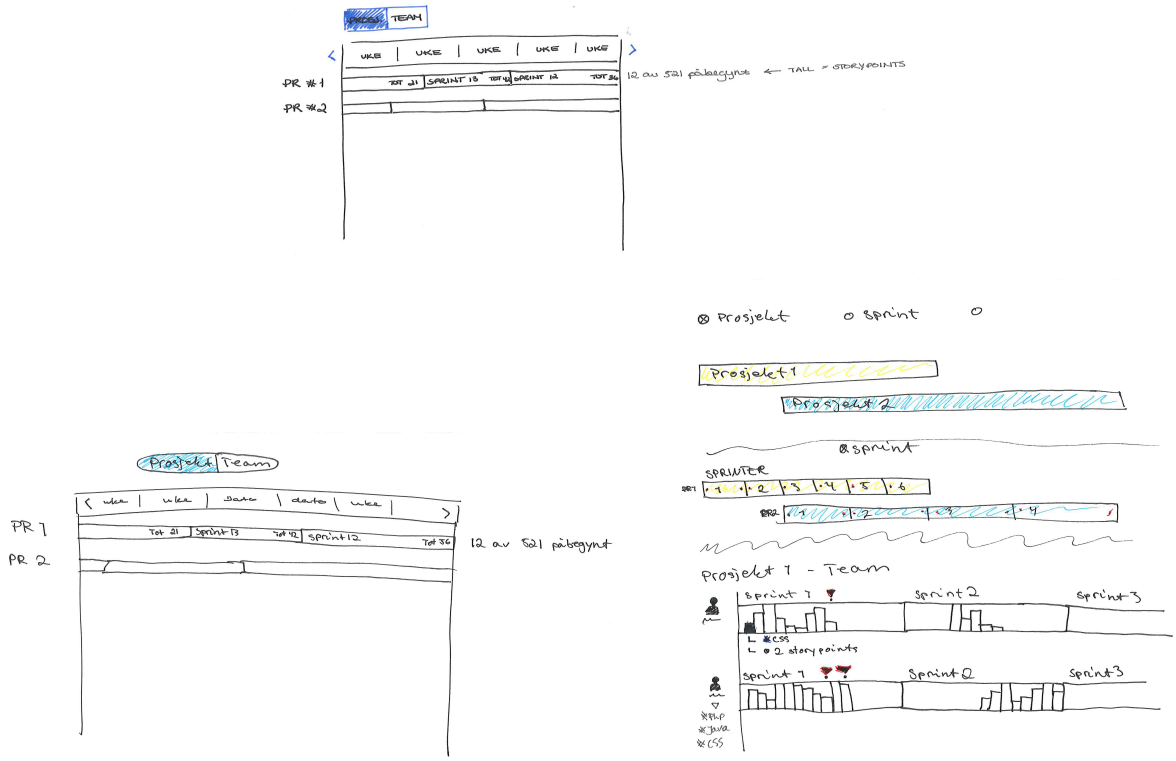


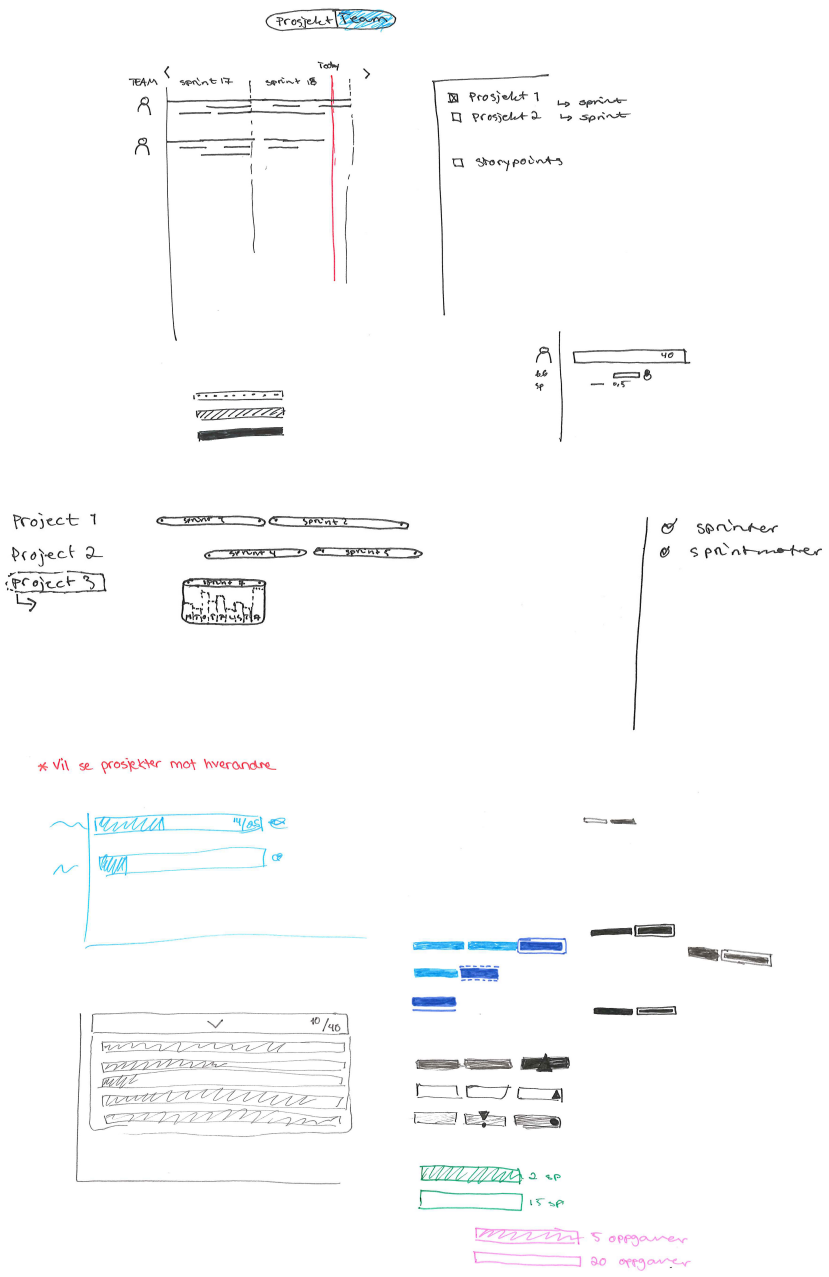
Over er tidlige skisser til sprintvisning. Under er tidlige skisser for filtrering av prosjekter.



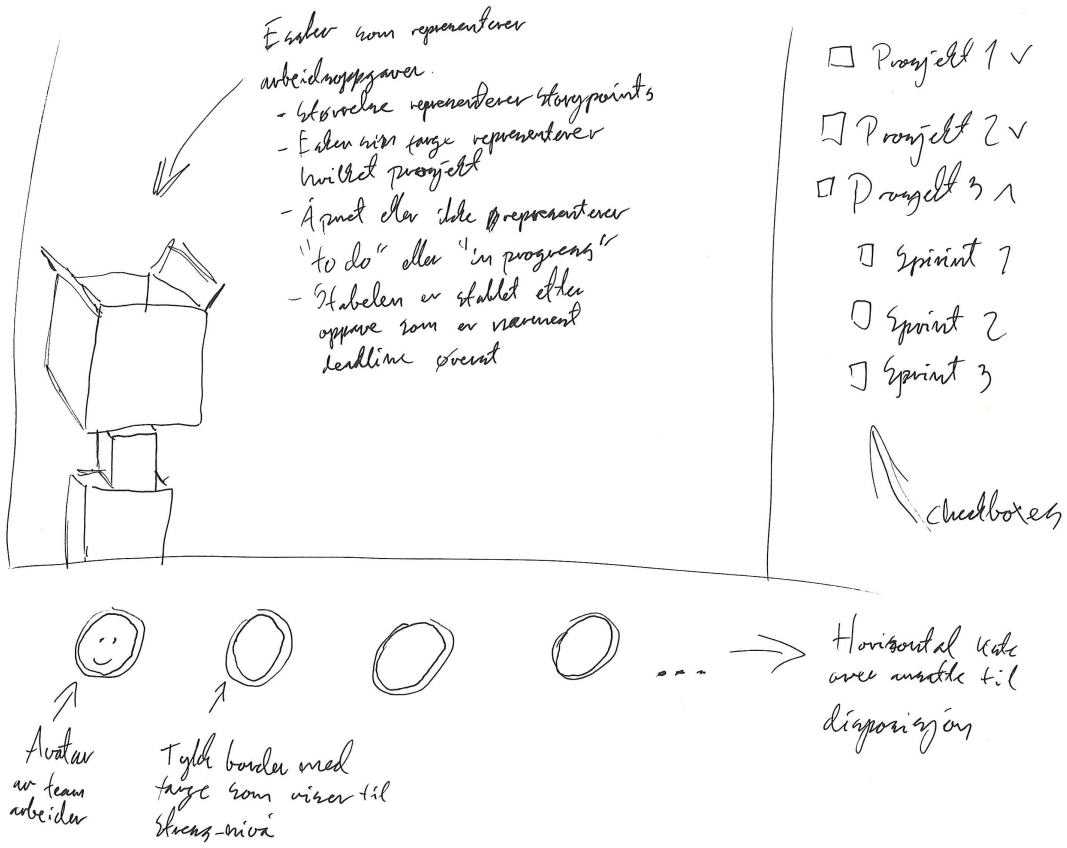


Over er forslag til ressursperson-visning. Under er flere ulike skisser til prosjekt.

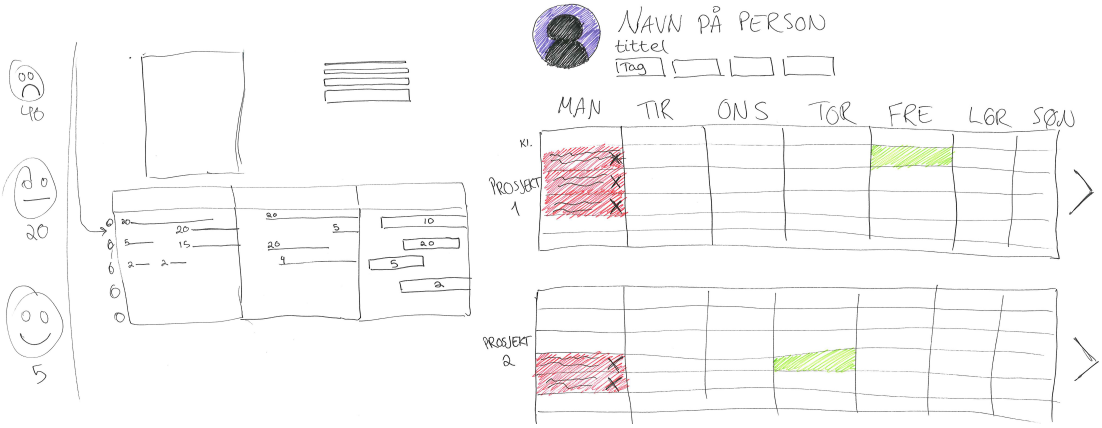


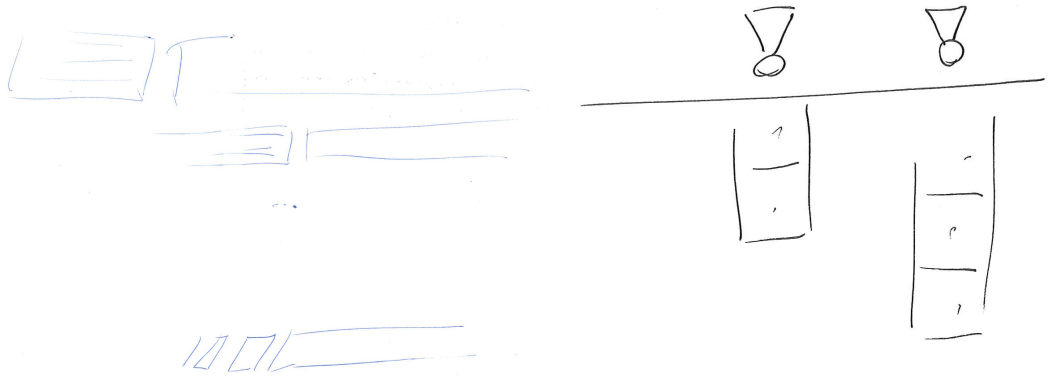


Over er ulik markering for overbelastning på sprinter

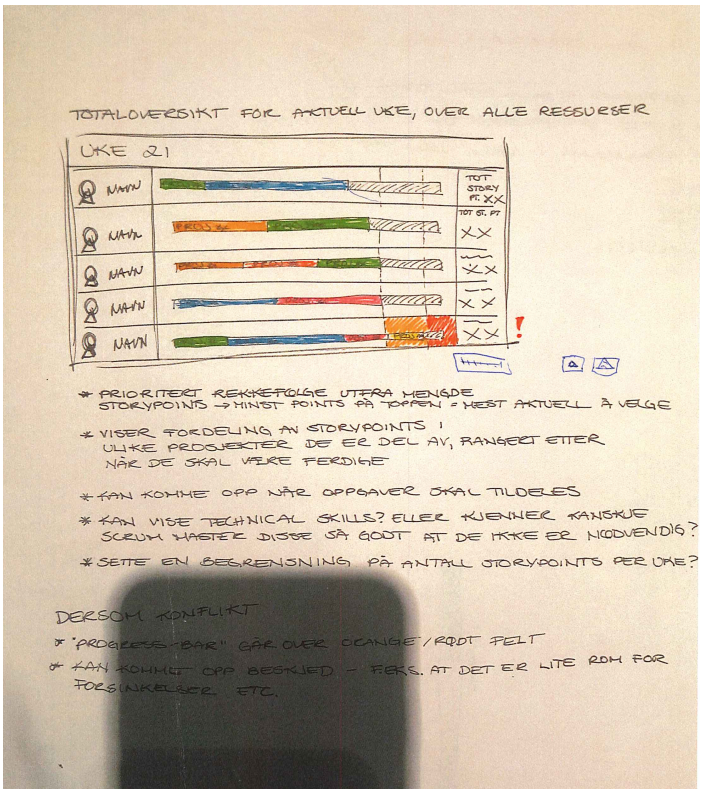


Over: Helt ny idé om å bruke bokser til oppgaver. Tok for mye plass og vanskelig å implementere i praksis. Under er tidlig og senere skisser til ressurspersoner.



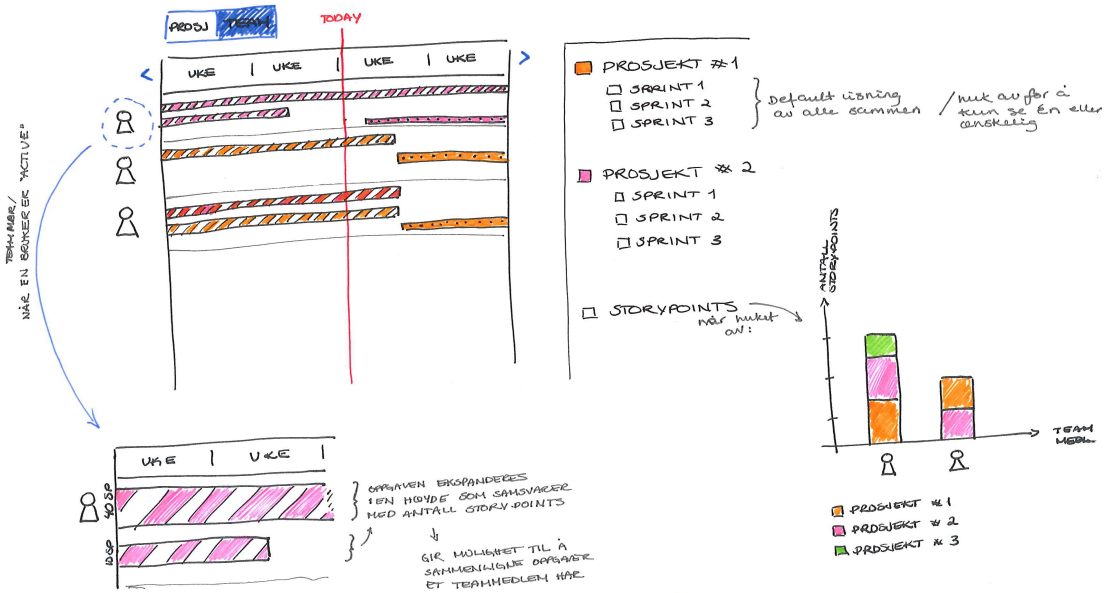


Over: Til venstre er forslag til avslutning på sprinter som ikke har sluttdato. Høyre viser forslag til symbol for overbelastning. Under, til høyre, vises et forslag til pop-up-vindu når man vil slette en oppgave.

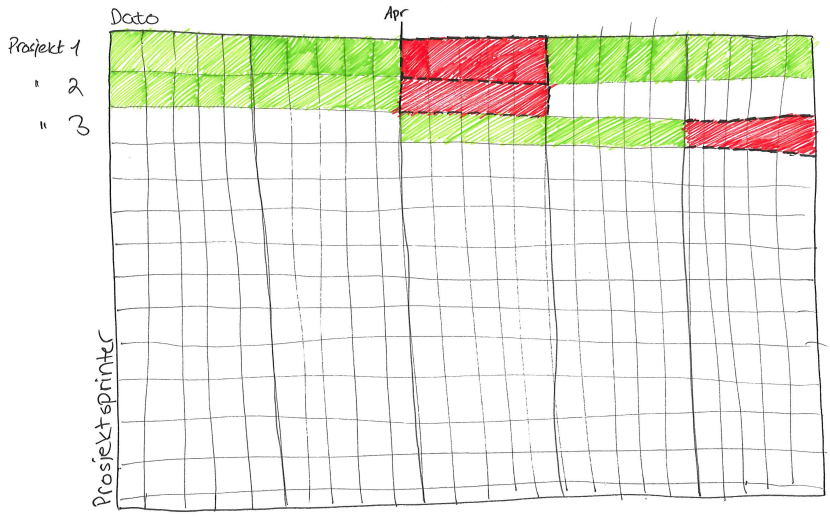


Over vises en mer forklarende skisse til totaloversikt.

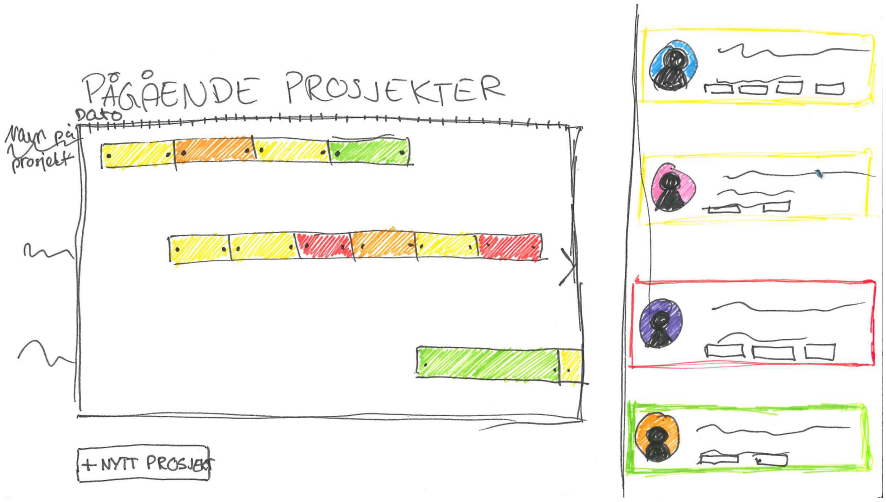
Vedlegg 19: Lo-fi prototype



Lo-fi prototype av prosjektsiden, som blir første side man kommer til.



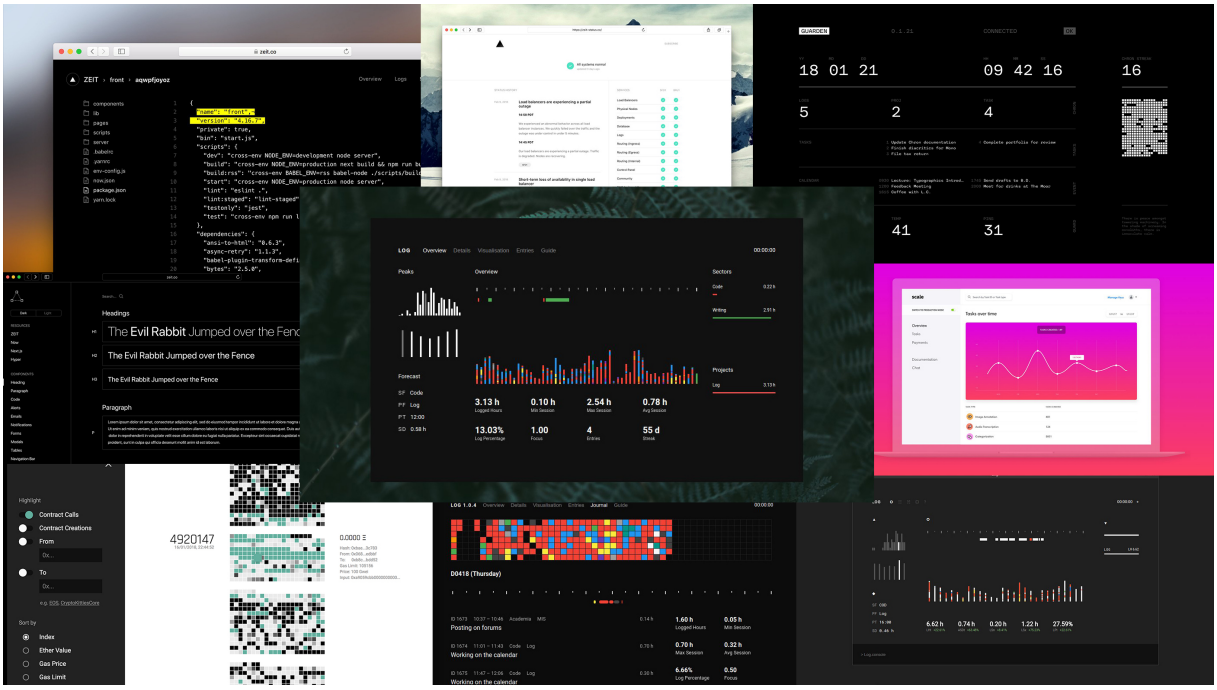
Lo-fi prototype der kun prosjekter vises, rødt for kritisk belastning og grønt for gode planlagte sprinter.



Her vises pågående prosjekter med prosjekter i parallell, der sprintbelastningen er markert etter fargegrad. Til høyre finnes en sidebar med oversikt over ressurspersoner, der rød kantlinje betyr kritisk overbelastet, grønt er ok og gul bør undersøkes.

Lo-fi prototype av ressurspersoner. Her vises navn, ansattes tittel og kompetanse-emneknagger. Er ressurspersonen tildelt flere prosjekter, vises disse sammen med overbelastning i rødt i ukesvisning.

Vedlegg 20: Moodboard



Vedlegg 21: Designmanual

Designmanual

Typografi

Brødtekst – 16 px (1 rem)

- mye info ellers på siden, typografi og font skal ikke være et forstyrrende element
- sans-serif er mye brukt på web / skjerm
- lesbar på de aller fleste skjermer
- 5 ulike stiler med tilhørende kursiv gir mange variasjoner og muligheter

Farger

- svart er "ingenting" alt som ikke er svart er viktig / av betydning
- svart / hvit - kontrast = universell utforming
- valg av temafarger/bakgrunn gir høyde for arbeidsmiljø, ulike situasjoner (lyst rom / mørkt rom)
- sterke og klare farger skaper blikkfang
- fargene skiller seg fra hverandre

Mulig å endre temafargene:
- Sort bakgrunn (#000)
- Hvit bakgrunn (#FFF)

Prosjektfarger

project-pink: #fa2ccc
project-lightpink: #ff7aa7
project-purple: #882cfa
project-lightpurple: #c863ff
project-blue: #5c6bff
project-lightblue: #5cb1ff
project-mintgreen: #5effc3
project-green: #66ff5e

Elementfarger

current-date: #fa2c2c

Tre ulike seriøsitetsgrader for risiko
Gul – Orange – Rød
project-yellow: #ffd34c
project-orange: #fa8d2c
current-date: #fa2c2c

Grid

CSS-grid: 25 + 50 + 75 + 100

Luft er brukt som skille-element (Gestalt prinsippet om nærhet)

Vedlegg 22: Samtykkeskjema

NTNU
Det skapende universitet

Gjøvik, 08.02.2018

SAMTYKKE: BRUKERUNDERSØKELSE

BACHELORPROSJEKT – SCRUM

Vi er en bachelorgruppe med fire studenter fra Bachelor i Webutvikling ved Institutt for Design, NTNU i Gjøvik. Bachelorgruppen består av Silje Jeanette Fruseth Lien, Susanne Waaler Stenshagen, Audun Meek Olsen og Martine Jacobsen. Veileder på prosjektet er Eivind A. Johansen og oppdragsgivers kontaktperson er Terje Krogstad.

Bachelorprosjektet går ut på at Escio, som utnytter seg av den smidige metodikken Scrum, ønsker en tilleggspakke til Jira de kan bruke til å visualisere allokeringen av teammedlemmer med relevant kompetanse, som de har til disposisjon til de ulike oppgavene som inngår i et prosjekt.

Formålet med vår undersøkelse er å kartlegge brukernes behov. Selv om løsningen i hovedsak vil ha størst verdi for prosjektledere som har ansvaret for å fordele arbeidsoppgaver til teammedlemmer, ønsker bachelorgruppen å kartlegge hvordan disse opplever denne fordelingen. Blant det bachelorgruppen ønsker å undersøke er hva teammedlemmene ønsker at prosjektledere skal ta hensyn til, hvilke konflikter oppleves at oppstår oftest og hvilke er mest kritiske.

Frivillig deltakelse

All deltakelse i denne undersøkelsen er frivillig, og du kan når som helst avslutte undersøkelsen, trekke samtykket eller trekke tilbake informasjon som er gitt under intervju og samtale. Du er på ingen måte pålagt å oppgi årsak for hvorfor du ønsker å trekke deg. Dersom du ønsker å trekke samtykket eller trekke tilbake informasjon, vil dette umiddelbart bli slettet.

Bachelorgruppen vil under undersøkelsen ta **notater og lydopptak**. Dette for å sikre at bachelorgruppen får med all nødvendig informasjon og på denne måten ta hensyn til alle brukere og involverte ressurser for å utvikle en løsning som gir verdi for alle parter på et Scrum-prosjekt.

Ved spørsmål knyttet til bachelorprosjektet og denne undersøkelsen, ta kontakt med:

Martine Jacobsen
Student / leder av bachelorgruppen
Mail: martijac@stud.ntnu.no
Tlf: 958 74 556

Eivind A. Johansen
Bachelorveileder / Universitetslektor
Mail: eivind.johansen@ntnu.no
Tlf: 916 37 882

Terje Krogstad (ESCIO)
Teknisk prosjektleder / seniorutvikler
Mail: terje@escio.no
Tlf: 959 72 382

NTNU
Det skapende universitet

Gjøvik, 08.02.2018

Studien er *ikke* meldt til personvernombudet for forskning, Norsk senter for forskningsdata.

Anonymitet

Notater, lydopptak og informasjonsskriv knyttet til undersøkelsen vil bli anonymisert. Lydopptakene vil kun være tilgjengelig for bachelorgruppens fire medlemmer i en begrenset periode. Denne perioden er satt **fra tidspunkt undersøkelsen finner sted og frem til og med 1. mars, 2018**. Lydopptakene vil bli transkribert og deretter slettet. Transkriptet vil bli anonymisert.

Det vil kun være bachelorgruppen som vet hvem som intervjues, og all informasjon som er gitt vil ikke kunne tilbakeføres til deg. Informasjonen som innhentes vil bli benyttet for å kartlegge og analysere behov, samt bistå bachelorgruppen i å ta avgjørelser i ulike situasjoner knyttet til utvikling av endelig løsning. All innhentet informasjon vil ikke bli brukt til annet formål enn for dette bachelorprosjektet.

Før undersøkelsen starter ber vi deg om å samtykke i deltakelse ved å krysse av boksene som representerer aktuelt samtykke, samt undertegne på at du har lest og forstått innholdet i undersøkelsen og dette informasjonsskrivet.

Samtykke

- ▢ Jeg har lest og forstått innholdet i dette informasjonsskrivet og er inneforstått med undersøkelsens formål
- ▢ Jeg samtykker til at lydopptak blir gjort og arkivert t.o.m. 01.03.2018
- ▢ Jeg gir mitt samtykke til å delta i undersøkelsen

Sted og dato	Signatur

2

Vedlegg 23: Intervjuspørsmål

1. Hva er de største utfordringene med smidige metoder?
    - Noen av disse som spesifikt gjelder SCRUM?
    - Noen av disse som spesifikt gjelder JIRA?
2. Hva er utfordringene når det kommer til å drive med prosjekter i parallell?
    - Er det noen av disse som JIRA løser godt, i så fall hvilke?
3. Hvis du blir møtt en uforsvarlig stor arbeidsmengde, hvordan rapporterer du dette videre?
    - Har dere noen verktøy for dette? Ev. hvilke?
    - Blir slike situasjoner oppdaget for sent? Ev. hvorfor?
    - Hvor lang tid tar det å løse en slik konflikt og anser dere dette som mye tapt tid?
4. Kan du beskrive prosessen fra nytt oppdrag, til ferdigsatte sprinter i JIRA?
    - Hvordan organiserer prosjektleder oppdraget?
        - I samarbeid med teamet? Alene?
    - I hvor stor grad har utviklerne innflytelse på oppgaver de blir tildelt?

Vedlegg 24: Scenarier til brukstest

Scenarier

1. Du ønsker et **generelt** overblikk over dine arbeidere og deres belastning. Naviger til visningen best utrustet til dette.
   Fasit: "storypoints"

2. Med utgangspunkt i storypoints visningen ser du at "Audun Olsen" ser ut til å ha størst total av storypoints. Du ønsker ytterligere informasjon om Audun sine oppgaver.
   Merk: det er ingen kobling i XD mellom en person under storypoints og en person under "ressurspersoner". Informer tester om at dette egentlig skal være tilstede.
   Fasit: Klikk "Audun Olsen" under "storypoints" (gå til "ressurspersoner")
   a. Se på oppgavene og forklar om du ser forskjellen på oppgavene sin status
   b. Med informasjonen du nå ser, ville du gått tilbake til Jira sprint-oversikt/backlog for å endre oppgaver? Hva ville du isåfall gjort? (flytte oppgaver, endre datoer, etc)
   c. Du er nysgjerrig på en oppgave og ønsker å se mer info om den.
      Fasit: klikk på en oppgave for tooltip
      Merk: kun en oppgave er klikkbar, informer om dette. Informer/diskuter fortløpende om prosjektnavn/oppgavebeskrivelse i selve oppgave "boksen" så langt boksen strekker for forbedret visual "grepping" mellom Jira og Simasu.
   d. Du er prosjektleder for "Gausdal Landhandleri" og ønsker kun å se oppgaver for dette prosjektet.
      Fasit: klikk "Gausdal Landhandleri" i høyre sidemeny. (informer om at den ikke er klikkbar)

3. Du er som sagt prosjektleder for "Gausdal Landhandleri" og ønsker å se arbeidere sin storypoints-total for kun Gausdal Landhandleri.
   Fasit: endre til "storypoints visningen" og filtrer i høyre sidemeny

4. Du ønsker å se storypoints-totalen for enhver arbeider og alle pågående prosjekter, men du ønsker å få en oversikt over "velocity"/fart aspektet på prosjektene
   Fasit: filtrer tilbake alle prosjekter og huk av på "3 ukers periode"
   Merk: diskuter om 3 uker er for arbitrært og om bruker bør kunne selv velge tidsintervallet. Diskuter også hvordan tidsintervallet bør forholde seg til dags dato.

5. Du vil få en oversikt over alle prosjekter og deres respektive sprinter.
   Fasit: Trykk på "prosjekt" i navigasjonsmenyen
   Merk: Her vil det kanskje være lurt å diskutere navnene på de forskjellige visningene

6. Se på sprintene å forsøk å få en formening om hvor bra de ulike sprintene har gått
   Merk: diskuter nyanse av farge som måte for å gjengi hvordan en sprint har gått

7. Gjør deg en formening om hvilke prosjekter som går godt og hvilke som går mindre godt.
   Fasit: se venstre sidemeny

Generelle spørsmål

- Bruker dere "due date" for å sette deadlines på oppgaver eller bruker dere bare sluttdatoen på en sprint som sluttdato for oppgave?
  Merk: her kan det være relevant å vise forrige iterasjon av prototypen sin "ressurspersoner" visning.

- Har du noen flere kommentarer?
  - forbedringer?
  - negative/positive kommentarer?

Vedlegg 25: Felles møtelogg

MØTELOGG

UKE 2

Onsdag 10. januar

TIDSBRUK	TILSTEDE	TEMA: Diskusjon innad i gruppen
ca. 1 time	Martine Jacobsen Silje Lien Audun Olsen Susanne Stenshagen	Gruppens fire medlemmer diskuterte de ulike parter som skulle kontaktes og møtes. Ble enig om å kontakte veileder for kort møte, deretter ble det avtalt møte med bibliotekar for hjelp i databasesøking.

TIDSBRUK	TILSTEDE	TEMA: Forelesning
ca. 1 time	Martine Jacobsen Silje Lien Audun Olsen Susanne Stenshagen	Første lynkurs i bachelor (forelesning).

Torsdag 11. januar

TIDSBRUK	TILSTEDE	TEMA: Møteforberedelse
ca. 1 time	Silje Lien Susanne Stenshagen	Forberedte dokument med spørsmål til møte med bibliotekar. Ble enige om å lese hver vår tidligere bachelor-oppgave før møte med bibliotekar.

UKE 3

Mandag 15. januar

TIDSBRUK	TILSTEDE	TEMA: Møte
30 minutter	Silje Lien Audun Olsen Susanne Stenshagen	Møte med Karen Marie Øvern (bibliotekar) om artikkelsøk i databaser.

Tirsdag 16. januar

TIDSBRUK	TILSTEDE	TEMA: Oppstartsmøte med veileder
40 minutter	Silje Lien Audun Olsen Susanne Stenshagen Eivind Johansen (veileder)	Detaljert informasjon og tips fra veileder (se referat)
ca. 1 time	Silje Lien Audun Olsen Susanne Stenshagen	Diskuterte og lagde førsteutkast til gruppeavtalen og kjøpte bok

Onsdag 17. januar

TIDSBRUK	TILSTEDE	TEMA: Forelesning
1 time	Silje Lien Audun Olsen Susanne Stenshagen	Lynkurs i bachelor (til medielinjene).
2 timer	Silje Lien Audun Olsen Susanne Stenshagen	Lagde utkast til prosjektbeskrivelse, rollefordeling, prosjektskisse/plan

UKE 4

Onsdag 24. januar

TIDSBRUK	TILSTEDE	TEMA: Spesifikasjon av oppgave
4 timer	Silje Lien Audun Olsen Susanne Stenshagen Martine Jacobsen	Diskuterte ulike forslag til oppgavespesifikasjon

Torsdag 25. januar

TIDSBRUK	TILSTEDE	TEMA: Oppstartsmøte med oppdragsgiver
45 min	Silje Lien Audun Olsen Martine Jacobsen Håvard Narvesen (Escio)	Diskuterte ulike forslag til oppgavespesifikasjon Høre oppdragsgivers formening om oppgaven og spesifisering/retning

Fredag 26. januar

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte
40 min	Silje Lien Audun Olsen Martine Jacobsen Eivind Johansen (veileder)	Diskuterte prosjektplanen og fant kilder/artikler

UKE 5

Mandag 29. januar

TIDSBRUK	TILSTEDE	TEMA: Prosjektplanen
5 timer	Silje Lien Audun Olsen Susanne Stenshagen	Jobbet ferdig med prosjektplanen og gruppeavtalen

Tirsdag 30. januar

TIDSBRUK	TILSTEDE	TEMA: Artikler og disposisjon
4 timer	Silje Lien Susanne Stenshagen	Jobbet med rapportdisposisjonen og fant artikler. Leverte prosjektavtalen

Torsdag 1. februar

TIDSBRUK	TILSTEDE	TEMA: Datavisualisering og konkurrenter
5 timer	Silje Lien Susanne Stenshagen Audun Olsen	Så etter konkurrenter Informasjonsgrafikk Datavisualisering

Fredag 2. februar

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte
30 min	Silje Lien Susanne Stenshagen Audun Olsen Martine Jacobsen Eivind Johansen (veileder)	Diskuterte prosjektplanen, rapport og artikler
5 timer	Silje Lien Susanne Stenshagen Audun Olsen Martine Jacobsen	Satte opp Jira prosjekt Lagde utkast til backlog

UKE 6

Tirsdag 6. februar

TIDSBRUK	TILSTEDE	TEMA: Forberedelse brukerundersøkelse
3 timer 30 min	Silje Lien Audun Olsen Susanne Stenshagen	Lagde utkast til brukerundersøkelse (intervju), samt så mer på backloggen

Onsdag 7. februar

TIDSBRUK	TILSTEDE	TEMA: Referanser
4 timer	Silje Lien Audun Olsen Susanne Stenshagen	Leste og skrev notater fra relevante bøker og artikler

Torsdag 8. februar

TIDSBRUK	TILSTEDE	TEMA: Møte med oppdragsgiver
2 timer	Silje Lien Martine Jacobsen Susanne Stenshagen Terje Krogstad (Escio)	Diskuterte backlog, samt utførte en brukerundersøkelse
1 time	Silje Lien Martine Jacobsen Susanne Stenshagen	Gjennomgikk temaene fra møtet og sammenfattet hva vi skulle gjøre hver for oss

Fredag 9. februar

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte
30 min	Silje Lien Martine Jacobsen Audun Olsen Susanne Stenshagen Eivind Johansen (veileder)	Godkjenning av prosjektplan og samtykkeskjema. Diskuterte oppdragsgiver-møtet og planen fremover

Onsdag 14. februar

TIDSBRUK	TILSTEDE	TEMA: Skissering
1 time 30 min	Silje Lien Audun Olsen Susanne Stenshagen	Idémyldring og skissering av wireframes skriftlig

Fredag 16. februar

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte
40 min	Silje Lien Susanne Stenshagen Eivind Johansen (veileder)	Diskuterte rapportdisposisjon og brukeranalyser
3 timer	Silje Lien Susanne Stenshagen	Redigerte rapportdisposisjon etter kommentarer fra veileder

UKE 8

Tirsdag 20. februar

TIDSBRUK	TILSTEDE	TEMA: Kodedag
6 timer	Martine Jacobsen Silje Lien Audun Olsen Susanne Stenshagen	Intensivkurs om Git, Sass, Pug, Coffee. Hadde retrospektiv møte om sprint 1.

Onsdag 21. februar

TIDSBRUK	TILSTEDE	TEMA: Wireframes og teknisk prototype
6 timer	Silje Lien Audun Olsen Susanne Stenshagen	Idémyldret oss frem til wireframes og brukerreiser (& scenarier). Jobbet med teknisk prototype. Deltok på seminar.

Fredag 23. februar

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte og oppdragsgiver
45 min	Silje Lien Audun Olsen Susanne Stenshagen Eivind Johansen (veileder)	Diskuterte blabla
2 timer	Audun Olsen Susanne Stenshagen Terje Krogstad (Escio)	Sprint review for sprint 1. Planla sprint 2.
3 timer	Silje Lien Audun Olsen Susanne Stenshagen	Fullførte backloggen for Sprint 2 og

UKE 9

Tirsdag 27. februar

TIDSBRUK	TILSTEDE	TEMA: Wireframes
6 timer	Silje Lien Susanne Stenshagen Audun Olsen	Videreutviklet ideer og tegnet wireframes

Onsdag 28. februar

TIDSBRUK	TILSTEDE	TEMA: Rapport og EndNote
4 timer	Silje Lien Susanne Stenshagen	Noterte fra ulike artikler og arbeidet med referanser (EndNote). Lagde rapportstruktur i Drive for å skrive rett inn

Torsdag 1. mars

TIDSBRUK	TILSTEDE	TEMA: Kodedag (Vue), rapport, wireframes
7 timer	Silje Lien Susanne Stenshagen Martine Jacobsen Audun Olsen	Kodedag i Vue, ferdigstilte wireframes og gjennomgikk rapportstruktur, samt EndNote

UKE 10

Tirsdag 6. mars

TIDSBRUK	TILSTEDE	TEMA: Hi-Fi Prototype
4 timer	Silje Lien Audun Olsen Susanne Stenshagen	Jobbet videre med teknisk prototype og begynte arbeidet med hi-fi prototype i Adobe XD.

Onsdag 7. mars

TIDSBRUK	TILSTEDE	TEMA: Rapport + Personas + Seminar
7 timer	Silje Lien Audun Olsen Susanne Stenshagen	Jobbet videre med teknisk prototype og skrev litt notater i rapporten. Lagde to personas til. Var på seminar.

Torsdag 8. mars

TIDSBRUK	TILSTEDE	TEMA: Hi-Fi Prototype
4 timer	Silje Lien Audun Olsen Susanne Stenshagen	Jobbet videre med teknisk prototype og ferdigstilte Hi-Fi prototype, samt klargjorde scenarier til brukertest.

Fredag 9. mars

TIDSBRUK	TILSTEDE	TEMA: Brukertest
6 timer	Silje Lien Audun Olsen Susanne Stenshagen	Brukertest 1 av Hi-Fi Prototype.

UKE 11

Tirsdag 13. mars

TIDSBRUK	TILSTEDE	TEMA: Analysering av brukertest 1
3 timer	Silje Lien Susanne Stenshagen	Skrev stikkord til rapport: analysering og evaluering av brukertest 1

Torsdag 15. mars

TIDSBRUK	TILSTEDE	TEMA: Diskusjon hi-fi og Sprint retrospektiv 2
7,5 timer	Silje Lien Susanne Stenshagen Audun Olsen Martine Jacobsen	Diskuterte funnene fra brukertest 1 av hi-fi prototypen, samt så på data. Hadde sprint retrospektiv 2.

Fredag 16. mars

TIDSBRUK	TILSTEDE	TEMA: Rapport og veiledningsmøte
5 timer	Silje Lien Audun Olsen Susanne Stenshagen Martine Jacobsen	Ferdigstilte sprint 2 ved å endre litt på hi-fi prototypen. Arbeidet med rapporten.
1 time	Martine Jacobsen Silje Lien Susanne Stenshagen Eivind Johansen (veileder)	Veiledningsmøte om hi-fi prototypen.

UKE 12

Tirsdag 20. mars

TIDSBRUK	TILSTEDE	TEMA: Oppdragsgivermøte
2 timer	Silje Lien Susanne Stenshagen Audun Olsen	Forberedelse til møte
2 timer?	Silje Lien Susanne Stenshagen Audun Olsen Martine Jacobsen Terje Krogstad (Escio)	Sprint review for sprint 2. Planla sprint 3. Diskuterte hi-fi prototypen.
2 timer?	Silje Lien Susanne Stenshagen Audun Olsen Martine Jacobsen	Organisering av den nyoppstartede sprinten

Onsdag 21. mars

TIDSBRUK	TILSTEDE	TEMA: Designmal og implementasjon
6 timer	Martine Jacobsen Silje Lien Audun Olsen Susanne Stenshagen	Laget en designmanual og deretter implementerte dette i kode.

Torsdag 22. mars

TIDSBRUK	TILSTEDE	TEMA: Rapport og skrivekveld
4,5 timer	Martine Jacobsen Silje Lien Audun Olsen Susanne Stenshagen	Rapportskriving
7,5 timer	Martine Jacobsen Silje Lien Audun Olsen Susanne Stenshagen	Skrivekveld på biblioteket: Foredrag og noterte fra referanser

Fredag 23. mars

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte og rapport
3 timer	Martine Jacobsen Audun Olsen Susanne Stenshagen	Notering fra referanser og rapportskriving
1 time	Martine Jacobsen Audun Olsen Susanne Stenshagen Eivind Johansen (veileder)	Generell oppdatering: Rapport, hi-fi prototype og koding
2 timer	Martine Jacobsen Audun Olsen Susanne Stenshagen Silje Lien	Notering fra referanser og rapportskriving

UKE 13

Tirsdag 27. mars

TIDSBRUK	TILSTEDE	TEMA: Klargjøring brukertest med Escio
5 timer	Silje Lien Susanne Stenshagen	Iterasjon på hi-fi

UKE 14

Tirsdag 3. april

TIDSBRUK	TILSTEDE	TEMA: Brukertest Escio og iterasjon
1 time 30 min	Silje Lien Susanne Stenshagen Martine Jacobsen	Brukertest på Håvard hos Escio
5 timer 30 min	Silje Lien Susanne Stenshagen Martine Jacobsen	Iterasjon og klikkbar hi-fi

Onsdag 4. april

TIDSBRUK	TILSTEDE	TEMA: Rapportskriving
3 timer	Susanne Stenshagen Martine Jacobsen Audun Olsen	Skriving i rapport og notater fra bøker
4 timer	Susanne Stenshagen Martine Jacobsen Audun Olsen Silje Lien	Sprint retrospektiv og planlegging av neste sprint + mer rapport/bøker/artikler

Torsdag 5. april

TIDSBRUK	TILSTEDE	TEMA: Rapport
6 timer	Silje Lien Susanne Stenshagen	Rapport (notater fra bøker og stikkord)

Fredag 6. april

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte og oppdragsgiver
1 time	Silje Lien Susanne Stenshagen Eivind Johansen (veileder)	Generell oppdatering: Rapport, hi-fi prototype og koding
2 timer	Audun Olsen Susanne Stenshagen Silje Lien Terje Krogstad (Escio)	Sprint review 3 og planlegging av sprint 4. Brukertest av hi-fi.
4 timer	Audun Olsen Susanne Stenshagen Silje Lien	Sammenfattet notater etter møtene tidligere på dagen og fikk oversikt ++

UKE 15

Tirsdag 10. april

TIDSBRUK	TILSTEDE	TEMA: Poker planning til sprint 4
6 timer	Silje Lien Audun Olsen Susanne Stenshagen	Poker planning for sprint 4 – laget backloggen ferdig. Rapportskriving + fra bøker.

Onsdag 11. april

TIDSBRUK	TILSTEDE	TEMA: Rapportskriving og bøker
7 timer	Silje Lien Audun Olsen Susanne Stenshagen	Rapportskriving + fra bøker.

Torsdag 12. april

TIDSBRUK	TILSTEDE	TEMA: Rapportskriving
5 timer	Silje Lien Susanne Stenshagen	Rapportskriving og oppdatering av kilder i EndNote

Fredag 13. april

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte og rapport
1 time	Silje Lien Susanne Stenshagen Audun Olsen Eivind Johansen (veileder)	Generell oppdatering: Rapport
6 timer	Audun Olsen Susanne Stenshagen Silje Lien	Rapportskriving

UKE 16

Tirsdag 17. april

TIDSBRUK	TILSTEDE	TEMA: Koding og rapport
5 timer	Silje Lien Audun Olsen Susanne Stenshagen	Kodet og skrev i rapporten
3 timer	Audun Olsen Susanne Stenshagen	Kodet og skrev i rapporten

Onsdag 18. april

TIDSBRUK	TILSTEDE	TEMA: Koding
5 timer	Audun Olsen Susanne Stenshagen	Kodet (se Jira-backlog)

Torsdag 19. april

TIDSBRUK	TILSTEDE	TEMA: Koding og rapport
3 timer	Susanne Stenshagen Silje Lien	Kodet (se Jira-backlog) og rapport
4 timer	Audun Olsen Susanne Stenshagen Silje Lien Martine Jacobsen	Kodet (se Jira-backlog) og rapport. Susanne la inn EndNote i OneDrive med oppdaterte kilder.

Fredag 20. april

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte og oppdragsgiver
3 timer (møte ca. 1 time)	Susanne Stenshagen Martine Jacobsen Audun Olsen Eivind Johansen (veileder)	Møte med veileder – snakk om rapport. Jobbet litt før/etterpå.
1 time	Audun Olsen Susanne Stenshagen Silje Lien Martine Jacobsen Terje Krogstad (Escio)	Statusmøte midt i sprint 4 – hva skal vi fokusere på den resterende uken?
2 timer	Audun Olsen Susanne Stenshagen Silje Lien	Koding og rapport + renskriving av notater fra statusmøte ang. Jira/Tempo o.l.

UKE 17

Tirsdag 24. april

TIDSBRUK	TILSTEDE	TEMA: Koding
5 timer	Silje Lien Audun Olsen Susanne Stenshagen	Kodet (se Jira-backlog)

Onsdag 25. april

TIDSBRUK	TILSTEDE	TEMA: Koding
11 timer	Silje Lien Audun Olsen Susanne Stenshagen	Kodet (se Jira-backlog)
2 timer	Audun Olsen Susanne Stenshagen	Filtrering på ressursbelastning og dropdown/accordion i ressurspersoner

Torsdag 26. april

TIDSBRUK	TILSTEDE	TEMA: Koding
6 timer	Silje Lien Audun Olsen Susanne Stenshagen	Kodet (se Jira-backlog)

Fredag 27. april

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte og oppdragsgiver
1 time	Silje Lien Audun Olsen Susanne Stenshagen Martine Jacobsen Eivind Johansen (veileder)	Diskuterte produktet (i kodet versjon)
2 timer	Silje Lien Audun Olsen Susanne Stenshagen Martine Jacobsen Terje Krogstad (Escio)	Sprint review 4 og totaloppsummering av produktet
3 timer	Silje Lien Audun Olsen Susanne Stenshagen Martine Jacobsen	Laget plan for de resterende ukene med tanke på rapportskriving. Rapport og kodet (2:2)
1 time	Susanne Stenshagen Audun Olsen Martine Jacobsen	Rapport og koding (1:2)
1 time	Audun Olsen Martine Jacobsen	Koding

UKE 18

Onsdag 2. mai

TIDSBRUK	TILSTEDE	TEMA: Rapport
4,5 timer	Silje Lien Susanne Stenshagen Audun Olsen	Silje skrev om mental modell (kap. 2). Susanne skrev om roller + kompetanse (kap. 1) + Scrum-teori (kap. 2) + sendte mail til Eivind. Audun skrev om webteknologier (kap.2)
4 timer	Silje Lien Susanne Stenshagen	Silje skrev om skisser/wireframes og evaluering brukertest. Susanne skrev om designprinsipper (kap. 2) og konkurrentanalyser i eget dokum. (kap. 3)

Torsdag 3. mai

TIDSBRUK	TILSTEDE	TEMA: Rapport
5 timer	Silje Lien Susanne Stenshagen Audun Olsen	Silje og Susanne skrev nytt punkt om litteratur-innledning. Audun skrev om samhandlingsverktøy og begynte på site-map. Silje skrev om prototype. Susanne skrev om konkurrent- og målgruppeanalyse(eget dokument).

Fredag 27. april

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte og rapport
2 timer	Silje Lien Audun Olsen Susanne Stenshagen Eivind Johansen (veileder)	Diskuterte rapport-elementer (se veiledningsdokument)
4 timer + 6 timer	Silje Lien Audun Olsen Susanne Stenshagen	Silje og Susanne ferdigstilte konkurrent- og målgruppeanalyse, skrev deretter om prototype (kap. 4) Audun skrev om informasjonsarkitektur (kap. 2)

UKE 19

Mandag 7. mai

TIDSBRUK	TILSTEDE	TEMA: Rapport
2 timer	Silje Lien Susanne Stenshagen	Prototype – kap. 4
4 timer	Silje Lien Susanne Stenshagen Audun Olsen Martine Jacobsen	Martine og Susanne skrev kap. 6 – Endelig løsning. Audun og Silje skrev kap. 7 – Videreutvikling.
3 timer	Susanne Stenshagen Audun Olsen Martine Jacobsen	Fikset litt på innledning og konklusjon. Skrev kap. 6 og kap. 7

Tirsdag 8. mai

TIDSBRUK	TILSTEDE	TEMA: Rapport
6,5 timer	Silje Lien Susanne Stenshagen Martine Jacobsen Audun Olsen	Brukerintervju i kap.2 – Teori. Sammendrag kap.0. Konklusjon kap.8. Innledning kap.1.

Onsdag 9. mai

TIDSBRUK	TILSTEDE	TEMA: Rapport
30 min	Silje Lien Susanne Stenshagen Martine Jacobsen	Konklusjon kap.8. Prototype kap.4. Martine kodet.
5,5 timer	Silje Lien Susanne Stenshagen Martine Jacobsen Audun Olsen	Sammendrag kap.0. Konklusjon kap.8. Prototype kap.4. Implementering kap.5. Martine kodet. Susanne fikset vedlegg i Word-fil.
30 min	Susanne Stenshagen Martine Jacobsen Audun Olsen	Implementering kap.5. Word og vedlegg.

Torsdag 10. mai

TIDSBRUK	TILSTEDE	TEMA: Rapport
10 timer (satt lenger, men ikke alt var effektiv arbeidstid)	Silje Lien Susanne Stenshagen Audun Olsen	Prototype kap.4. Implementering kap.5. Videreutvikling kap.7. Word-fil.

Fredag 11. mai

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte og rapport
16 timer	Silje Lien Audun Olsen Susanne Stenshagen	Prototype kap.4 Implementering kap.5 Vedlegg i Word ++ Midlertidig ferdigstillelse av hele rapporten

UKE 20

Mandag 14. mai

TIDSBRUK	TILSTEDE	TEMA: Veiledningsmøte og rapport
1 time	Martine Jacobsen Silje Lien Audun Olsen Susanne Stenshagen	Korrekturlesing og vedlegg i Word
2 timer	Martine Jacobsen Silje Lien Audun Olsen Susanne Stenshagen Eivind Johansen (veileder)	Diskuterte småpirk i rapport
14 timer	Martine Jacobsen Silje Lien Audun Olsen Susanne Stenshagen	Korrekturlesing, vedlegg i Word + innsetting i Word
1 time	Martine Jacobsen	Ferdigstille korrektur kap. 3

Tirsdag 15. mai

TIDSBRUK	TILSTEDE	TEMA: Rapport
1 time	Silje Lien Susanne Stenshagen	Word og korrektur
1 time	Silje Lien Susanne Stenshagen Martine Jacobsen	Word, korrektur ++
12 timer	Silje Lien Susanne Stenshagen Martine Jacobsen Audun Olsen	Ferdigstillelse rapport, korrektur ++

Vedlegg 26: Arbeidslogg for Martine

Arbeidslogg for Martine

DATO	TIDSBRUK	HVEM	HVA BLE GJORT?	HVA BLE IKKE GJORT?	TIL NESTE GANG
25/01	4 t 45 min	Martine Jacobsen	Prosjektplan - prosjektfaser		
26/01	3 t 30 min	Martine Jacobsen	GANTT		
29/01	3 t 30 min	Martine Jacobsen	GANTT		
30/01	1 t	Martine Jacobsen	ferdigstille GANTT og sette inn i plan		
06/02	1 t	Martine Jacobsen	finne informasjon for å skrive samtykkeskjema + finne relevante artiker (prototype)		
07/02	2t	Martine Jacobsen	Utforme samtykkeskjema og sende Eivind for godkjenning		
07/02	1	Martine Jacobsen	Sette opp oversikt over timebruk		
08/02	4 t	Martine Jacobsen	Endret sprintoppsett i GANTT-skjemaet og sende mail til Terje ang møtedatoer		
09/02	30 min (på skolen) + 1 t hjemme	Martine Jacobsen	lese i bachelorbok		
09/03	30 min	Martine Jacobsen	gikk gjennom prosess med å sette opp git og prosjekt via terminal til github (med eget prosjekt - ikke bachelor)		
13/02	1 t	Martine Jacobsen	Ukesplan for uke 7 og 8		
13/02	30 min	Martine Jacobsen	intervju med Håvard		

13/02	3 timer	Martine Jacobsen	transkribere intervju med Håvard	frem til 11:30 (i filen)	
14/02	1 t 30 min	Martine Jacobsen	transkribere intervju med Håvard		
14/02	1 t	Martine Jacobsen	rapportdisposisjon		
14/02	5 t	Martine Jacobsen	behovsanalyse: lete etter teori + utføring av analyse		
21/02	4 t	Martine Jacobsen	behovsanalyse teori, samle info fra bok (om bl.a. PACT) wireframes		
23/02	4 t	Martine Jacobsen	teori, samle info fra bok: PACT, Personas, Scenario		
24/02	1 t	Martine Jacobsen	Skrive om info hentet fra bok – Designing Interactive Systems		Skrive om info hentet fra bok – Designing Interactive Systems
25/02	1 t	Martine Jacobsen	Skrive om info hentet fra bok – Designing Interactive Systems		Skrive om info hentet fra bok – Designing Interactive Systems
25/02	3 t	Martine Jacobsen	Skrive om info hentet fra bok Gjennomføre PACT-analyse		Gjennomføre PACT-analyse
26/02	1,5 t	Martine Jacobsen	Ferdigstille PACT-analyse		
26/02	1,5 t	Martine Jacobsen	Høre på opptak av intervju med Terje + endre i PACT i henhold til hva som kom frem		
28/02	4,5 t	Martine Jacobsen	Notere fra bok – Designing Interactive Systems		
02/03	4 t	Martine Jacobsen	Notere fra bok – Designing Interactive Systems		

07/03	7 t	Martine Jacobsen	Notere fra bok + stikkord til rapport		
09/03	1 t	Martine Jacobsen	Lage samtykke-skjema til test		
15/03	1 t	Martine Jacobsen	Legge til fargevariabler i miljø		
16/03	2,5 t	Martine Jacobsen	Endringer prototype		
27/03	3 t	Martine Jacobsen	Skrive fra bok		
28/03	2 t	Martine Jacobsen	Skrive fra bok		
03/04	2 t	Martine Jacobsen	Endringer + klikkbar prototype		
05/04	2 t	Martine Jacobsen	Legge inn progressbarer	KOMMENTAR: møtte på litt error, warnings og slikt så tok litt tid	
05/04	1 t	Martine Jacobsen	Legge inn vertikal kalender-grid	KOMMENTAR: enkel løsning, må muligens endre for å passe fungerende løsning	
11/04	4 t	Martine Jacobsen	Lese og skrive om prototype	Fikk ikke veldig mange setninger, men fikk lest en del, prøver å skrive litt på hvert tema nå og heller fylle på senere	
13/04	3 t	Martine Jacobsen	Skrive om PACT-analyse	gjenstår litt mer	
13/04	3 t	Martine Jacobsen	litt mer om PACT	gjenstår litt mer	
16/04	4 t	Martine Jacobsen	Avslutte del om PACT i teori	Skrive inn PACT i analyse-delen Skrive om Use Cases	
17/04	4 t	Martine Jacobsen	Skrive inn PACT i analyse-delen Skrive om Use Cases		
18/04	6 t	Martine Jacobsen	skrive teori om designprinsipper lese design of	Skrevet i eget dokument	

			everyday things - (Don Norman)		
20/04	3 t	Martine Jacobsen	Kode tidsbegrensnings-filter		
22/04	4,5 t	Martine Jacobsen	Kode ressursbelastning		
23/04	4 t	Martine Jacobsen	Kode ressursbelastning	Møtte på litt utfordringer med å få kode til å fungere i nytt oppsett	
23/04	1 t	Martine Jacobsen	Finne ut av Vue-problem	Fant ingen ting	
24/04	6 t	Martine Jacobsen	Koding av prosjekt-sidebar og sette inn grid i nytt system		
25/04	5 t	Martine Jacobsen	Se på Vue-løsninger og grid til prosjekter		
25/04	3 t	Martine Jacobsen	Se på gridløsninger til prosjektoversikt og ressurspersoner		
26/04	1 t	Martine Jacobsen	Kode prosjektoversikt		
26/04	1,5 t	Martine Jacobsen	Kode prosjektoversikt		
26/04	5 t	Martine Jacobsen	Småpirk, backlog og ny accordion + prøve å forstå Vue		
28/04	2 t	Martine Jacobsen	Ferdigstilling av resources.pug		
29/04	9 t	Martine Jacobsen	Ferdigstilling av resources.pug		
30/04	5 t	Martine Jacobsen	Prosjektbeskrivelse + finne alle leveringskrav og frister + sende mail til Briskeby Media for print av oppg	Gjenstår noen detaljer under noen av punktene	
01/05	5 t	Martine Jacobsen	Prosjektorganiserin g		

03/05	5 t	Martine Jacobsen	Prosjektorganiserin g + risikoanalyse + universell utforming		
04/05	6 t	Martine Jacobsen	Universell utforming + prototype		
05/05	5 t	Martine Jacobsen	Analyse av eksisterende løsning		
08/05	4,5 t	Martine Jacobsen	ferdigstille konklusjon + kode		
09/05	4 t	Martine Jacobsen	skrive Implementering (Front-end) + starte på responsivt forslag		
10/05	4 t	Martine Jacobsen	Ferdigstille responsivt forslag + litt skriving (brukerreiser)		
11/05	5,5 t	Martine Jacobsen	Gjennomgang og omskriving på Prototype		
11/05	5,5 t	Martine Jacobsen	Omskriving og legge til på diverse kapitler, spes. 5, 7 og 8		
12/05	6,5 t	Martine Jacobsen	Korrekturlesing		
13/05	6 t	Martine Jacobsen	Korrekturlesing		

Vedlegg 27: Arbeidslogg for Silje

Arbeidslogg for Silje

DATO	TIDSBRUK	HVEM	HVA BLE GJORT?	HVA BLE IKKE GJORT?	TIL NESTE GANG
23/1	2 timer	Silje Lien	Utkast til disposisjon		
25/1	4t 45 min	Silje Lien	Prosjektplan - Gruppeavtale		
08/02	3 timer	Silje Lien	Transkriberte intervju	Ikke ferdig med transkribering	
09/02	4 timer	Silje Lien	Transkriberte intervju	Ikke ferdig med transkribering	
13/02	6 timer	Silje Lien	Transkriberte intervju. Planla og fordelte arbeidsoppgaver i sprint 1	Ikke ferdig med transkribering	
14/02	3 timer	Silje Lien	Transkriberte intervju	Ikke ferdig med transkribering	
15/02	4 timer	Silje Lien	Transkriberte intervju	Ble ferdig med transkrbering	
20/03	2 timer	Silje Lien	Planla ny sprint og skrev rapport		
29/3	3 timer	Silje Lien	Skrev av fra pensumbok	Ikke ferdig med skriving	
30/3	3 timer	Silje Lien	Skrev av fra pensumbok	Ble ferdig med skriving	
31/3	5 timer	Silje Lien	Skrev av fra pensumbok	Ble ikke ferdig med skriving	
15/4	1,5 timer	Silje Lien	Skrev i rapport (om brukstest)		
12/5	2 timer	Silje Lien	Korrekturlesing av rapport		
13/5	6 timer	Silje Lien	Korrekturlesing av rapport		

Vedlegg 28: Arbeidslogg for Audun

Arbeidslogg for Audun Olsen

DATO	TIDSBRUK	HVA BLE GJORT?	HVA BLE IKKE GJORT?	TIL NESTE GANG
15.01	3 timer	NPM utviklermiljø		Fortsett utviklingen
16.01	3 timer	NPM utviklermiljø		Fortsett utviklingen
17.01	3 timer	NPM utviklermiljø		Fortsett utviklingen
24.01	3 timer	NPM utviklermiljø		Fortsett utviklingen
31.01	3 timer	NPM utviklermiljø		Fortsett utviklingen
02.02	3 timer	NPM utviklermiljø		Fortsett utviklingen
05.02	3 timer	NPM utviklermiljø		Fortsett utviklingen
10.02	2 timer	NPM utviklermiljø		Fortsett utviklingen
28.02	2 timer	Teknisk prototype		
16/03	2,5 t	Teknisk prototype		
23/03	3t	Rapport		Skrive om teknologi/verktøy
22/04	6 timer	Typografisk hierarki og one-pager		
13/05	12 timer	Fylle ut kap. 5, 6, 7		

Vedlegg 29: Arbeidslogg for Susanne

Arbeidslogg for Susanne Stenshagen

DATO	TIDSBRUK	HVEM	HVA BLE GJORT?	HVA BLE IKKE GJORT?	TIL NESTE GANG
10/01	30 min	Susanne Stenshagen	Lagde Drive-mappe og møtelogg		
19/01	3 timer	Susanne Stenshagen	Leste bachelorbok		
29/01	30 min	Susanne Stenshagen	Lagde prikkesystem		
31/01	3 timer	Susanne Stenshagen	Leverte prosjektplan til veileder. Fant artikler. Leste og noterte stikkord fra artikler.		
08/02	3 timer	Susanne Stenshagen	Omrokkerte backlog. La inn Sprint 1. Sendte mail til Håvard om brukerundersøkelse		
09/02	4 timer	Susanne Stenshagen	Lastet ned EndNote, og lærte programmet via tutorials. Satte opp et utkast til bibliotek og Word-fil.		
13/02	6 timer	Susanne Stenshagen	Skrev notater fra bok. Planla og fordelte arbeidsoppgaver i sprint 1		
14/02	4 timer	Susanne Stenshagen	Skrev notater fra bok. Så på rapportdisposisjon og delte med Eivind		
15/02	1 time 30 min	Susanne Stenshagen	Skrev notater fra bok. Ferdig med bok.		

15/02	2 timer	Susanne Stenshagen	Så tutorials om Vue + noen JavaScript videoer		
27/02	45 min	Susanne Stenshagen	Lagde sprintplan for sprint 2		
16/03	1 time	Susanne Stenshagen	Skrev stikkord til rapporten		
20/03	2 timer	Susanne Stenshagen	Fullførte sprint 2 og planla sprint 3		
30/03	4 timer	Susanne Stenshagen	Noterte fra Scrum-bok		
15/04	1 time	Susanne Stenshagen	Skrev rapport (om Scrum)		
16/04	1 time	Susanne Stenshagen	Skrev rapport (om Scrum)		
17/04	1 time	Susanne Stenshagen	Skrev rapport (om Scrum) og la inn nytt EndNote-bibliotek i OneDrive		
24/04	2 timer	Susanne Stenshagen	Oppdaterte prototype i XD		
01/05	1 time	Susanne Stenshagen	Noterte i eget dokument om gruppens bakgrunn + info om oppdragsgiver		
06/05	6 timer	Susanne Stenshagen	Skrev om brukstest i kap. 4 – Prototype	Ikke helt ferdig utfylt, tar det neste dag	
07/05	1 time	Susanne Stenshagen	Skrev om brukstest i kap. 4 – Prototype		
12/05	2 timer	Susanne Stenshagen	Referanseliste i Word – la inn alt fra EndNote i løpende tekst		
13/05	7,5 timer	Susanne Stenshagen	Korrekturlesing + la inn kommentarer i Rapport/Drive		
13/05	1 time	Susanne Stenshagen	Vedlegg i Word	Ikke ferdig, mangler å legge inn alle skisser, tar det i morgen	

Vedlegg 30: Timeliste over arbeidstimer

UKE #	DATO	DAG	AUDUN	MARTINE	SILJE	SUSANNE	TOTAL TIMEBRUK
2	08.01.2018	mandag					
	09.01.2018	tirsdag					
	10.01.2018	onsdag	2	2	2	2,5	
	11.01.2018	torsdag			1	1	
	12.01.2018	fredag					
	13.01.2018	lørdag					
	14.01.2018	søndag					
	TOTALT PER PERS		2	2	3	3,5	**10,5**
3	15.01.2018	mandag	0,5		0,5	0,5	
	16.01.2018	tirsdag	1,75		1,75	1,75	
	17.01.2018	onsdag	3		3	3	
	18.01.2018	torsdag					
	19.01.2018	fredag				3	
	20.01.2018	lørdag					
	21.01.2018	søndag					
	TOTALT PER PERS		5,25	0	5,25	8,25	**18,75**
4	22.01.2018	mandag					
	23.01.2018	tirsdag			2		
	24.01.2018	onsdag	7	4	4	4	
	25.01.2018	torsdag	0,75	5,5	5,5		
	26.01.2018	fredag	0,75	4,25	0,75		
	27.01.2018	lørdag					
	28.01.2018	søndag					
	TOTALT PER PERS		8,5	13,75	12,25	4	**38,5**
5	29.01.2018	mandag	5	3,5	5	5,5	
	30.01.2018	tirsdag		1	4	4	
	31.01.2018	onsdag	3			3	
	01.02.2018	torsdag	5		5	5	
	02.02.2018	fredag	5,5	5,5	5,5	5,5	
	03.02.2018	lørdag					
	04.02.2018	søndag					
	TOTALT PER PERS		18,5	10	19,5	23	**71**

UKE #	DATO	DAG	AUDUN	MARTINE	SILJE	SUSANNE	TOTAL TIMEBRUK
6	05.02.2018	mandag	3				
	06.02.2018	tirsdag	3,5	1	3,5	3,5	
	07.02.2018	onsdag	4	3	4	4	
	08.02.2018	torsdag		7	6	6	
	09.02.2018	fredag	0,5	2	0,5	4,5	
	10.02.2018	lørdag	2				
	11.02.2018	søndag					
	TOTALT PER PERS		13	13	14	18	**58**
7	12.02.2018	mandag					
	13.02.2018	tirsdag		4,5	6	6	
	14.02.2018	onsdag	1,5	7,5	4,5	4,5	
	15.02.2018	torsdag			4	3,5	
	16.02.2018	fredag			4	4	
	17.02.2018	lørdag					
	18.02.2018	søndag					
	TOTALT PER PERS		1,5	12	18,5	18	**50**
8	19.02.2018	mandag					
	20.02.2018	tirsdag	6	6	6	6	
	21.02.2018	onsdag	6	4	6	6	
	22.02.2018	torsdag					
	23.02.2018	fredag	6	4	4	6	
	24.02.2018	lørdag		1			
	25.02.2018	søndag		4			
	TOTALT PER PERS		18	19	16	18	**71**
9	26.02.2018	mandag		3			
	27.02.2018	tirsdag	6		6	7	
	28.02.2018	onsdag	2	4,5	4	4	
	01.03.2018	torsdag	7	7	7	7	
	02.03.2018	fredag		4			
	03.03.2018	lørdag					
	04.03.2018	søndag					
	TOTALT PER PERS		15	18,5	17	18	**68,5**

UKE #	DATO	DAG	AUDUN	MARTINE	SILJE	SUSANNE	TOTAL TIMEBRUK
10	05.03.2018	mandag					
	06.03.2018	tirsdag	4		4	4	
	07.03.2018	onsdag	7	7	7	7	
	08.03.2018	torsdag	4	1	4	4	
	09.03.2018	fredag	6		6	6	
	10.03.2018	lørdag					
	11.03.2018	søndag					
	TOTALT PER PERS		21	8	21	21	**71**
11	12.03.2018	mandag					
	13.03.2018	tirsdag			3	3	
	14.03.2018	onsdag					
	15.03.2018	torsdag	7,5	8,5	7,5	7,5	
	16.03.2018	fredag					
	17.03.2018	lørdag					
	18.03.2018	søndag					
	TOTALT PER PERS		7,5	8,5	10,5	10,5	**37**
12	19.03.2018	mandag					
	20.03.2018	tirsdag	6	4	8	8	
	21.03.2018	onsdag	6	6	6	6	
	22.03.2018	torsdag	12	12	12	12	
	23.03.2018	fredag	5	5	5	5	
	24.03.2018	lørdag					
	25.03.2018	søndag					
	TOTALT PER PERS		29	27	31	31	**118**
13	26.03.2018	mandag					
	27.03.2018	tirsdag		3	5	5	
	28.03.2018	onsdag		2			
	29.03.2018	torsdag			3		
	30.03.2018	fredag			3	4	
	31.03.2018	lørdag			5		
	01.04.2018	søndag					
	TOTALT PER PERS		0	5	16	9	**30**

UKE #	DATO	DAG	AUDUN	MARTINE	SILJE	SUSANNE	TOTAL TIMEBRUK
14	02.04.2018	mandag					
	03.04.2018	tirsdag		9	7	7	
	04.04.2018	onsdag	7	7	4	7	
	05.04.2018	torsdag		3	6	6	
	06.04.2018	fredag	6		7	7	
	07.04.2018	lørdag					
	08.04.2018	søndag					
	TOTALT PER PERS		13	19	24	27	**83**
15	09.04.2018	mandag					
	10.04.2018	tirsdag	6		6	6	
	11.04.2018	onsdag	7	4	7	7	
	12.04.2018	torsdag			5	5	
	13.04.2018	fredag	7	6	7	7	
	14.04.2018	lørdag					
	15.04.2018	søndag			1,5	1	
	TOTALT PER PERS		20	10	26,5	26	**82,5**
16	16.04.2018	mandag		4			
	17.04.2018	tirsdag	8	4	5	8	
	18.04.2018	onsdag	5	6		5	
	19.04.2018	torsdag	4	5	7	7	
	20.04.2018	fredag	6	7	6	6	
	21.04.2018	lørdag					
	22.04.2018	søndag	6	4,5			
	TOTALT PER PERS		29	30,5	18	26	**103,5**
17	23.04.2018	mandag		5			
	24.04.2018	tirsdag	5	6	5	7	
	25.04.2018	onsdag	13	8	11	13	
	26.04.2018	torsdag	6	7,5	6	6	
	27.04.2018	fredag	8	8	6	7	
	28.04.2018	lørdag		2			
	29.04.2018	søndag		9			
	TOTALT PER PERS		32	45,5	28	33	**138,5**

UKE #	DATO	DAG	AUDUN	MARTINE	SILJE	SUSANNE	TOTAL TIMEBRUK
18	30.04.2018	mandag		5			
	01.05.2018	tirsdag		5		1	
	02.05.2018	onsdag	4,5		8,5	8,5	
	03.05.2018	torsdag	5	5	5	5	
	04.05.2018	fredag	12	6	12	12	
	05.05.2018	lørdag		5			
	06.05.2018	søndag				6	
	TOTALT PER PERS		21,5	26	25,5	32,5	**105,5**
19	07.05.2018	mandag	7	7	6	10	
	08.05.2018	tirsdag	6,5	11	6,5	6,5	
	09.05.2018	onsdag	6	10,5	6	6,5	
	10.05.2018	torsdag	10	4	10	10	
	11.05.2018	fredag	16	11	16	16	
	12.05.2018	lørdag		6,5	2	2	
	13.05.2018	søndag	12	6	6	8,5	
	TOTALT PER PERS		57,5	56	52,5	59,5	**225,5**
20	14.05.2018	mandag	17	18	17	17	
	15.05.2018	tirsdag	12	13	14	14	
	16.05.2018	onsdag					
	17.05.2018	torsdag					
	18.05.2018	fredag					
	19.05.2018	lørdag					
	20.05.2018	søndag					**122**
			312,25	**323,75**	**358,5**	**386,25**	**1502,75**