

HOW-WE-ROLL Rapport

Emnekode:	UIA IS-114
Emnenavn:	<i>Samskapning, kommunikasjon og Prosjektarbeid</i>
Emneansvarlig (normalt faglærer):	<i>Janis Gaillis og Niels F. Garmann-Johnsen</i>
Eventuell veileder:	
Innleveringsfrist/ tidspunkt:	<i>20.Nov.2022</i>
Antall ark inkl. denne forside:	<i>22</i>
Merknader/Tittel på bacheloroppgave:	<i>HOW-WE-ROLL rapport</i>

Jeg/vi bekrefter at jeg/vi ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	Ja <input checked="" type="checkbox"/>	Nei <input type="checkbox"/>
---	---	---------------------------------

kildehenvisning,

*Kopiering av andres tekster eller annen bruk av andres arbeider
uten kan bli betraktet som fusk.*

Kan besvarelsen gjenbrukes til forskning og undervisningsformål? (og i evt. publiseringer?)	Ja x	Nei
---	---------	-----

Gjelder kun gruppeeksamen:

Vi bekrefter at alle i gruppa har bidratt til besvarelsen	Ja x	Nei
---	---------	-----

Lenke til repository: <https://github.com/jonaslefdal/HOW-WE-ROLL-8B>
Github-projcts: <https://github.com/users/jonaslefdal/projects/3>

Introduksjon	3
(1)GR-KUNN-FERD	3
(2) FS-ANALYSE	4
Brukerhistorie nr .1	5
Brukerhistorie nr. 2	5
Brukerhistorie nr. 3 (HOW-WE-ROLL)	5
(3) Innsamling og representasjon av data	5
(4)FS-SYSTEM og FS-SYSTEM-STEG	6
(5) Velg Funksjoner	8
(6) FS-TESTER	10
Henting av verdier	10
Sending av verdier	11
Innhenting av data fra server	12
Fremstilling av data	13
(7) GITHUB-SAMARBEID	17
Issues	18
Fork-metoden	20
Konklusjon	21
Litteraturliste	22
Figurliste	

Figur 1	7
Figur 2	7
Figur 3	8
Figur 3.1	8
Figur 4.1	10
Figur 4.2	10
Figur 4.2.1	11
Figur 4.3	11
Figur 4.4	12
Figur 4.5	12
Figur 4.6	12
Figur 4.7	13
Figur 4.8	14
Figur 4.9	14
Figur 4.10	15
Figur 4.11	15
Figur 4.12	16
Figur 4.14	17
Figur 4.15	17
Figur 4.16	17
Figur 5.1	18
Figur 5.2	18
Figur 5.3	19
Figur 5.4	19
Figur 5.5	20
Figur 5.6	20
Figur 5.7	20
Figur 5.8	20
Figur 5.9	21
Figur 5.10	21

Introduksjon

I denne rapporten skal vi gå gjennom hele prosessen vår i utviklingen av en applikasjon som kan gjøre organisering av samarbeid i grupper bedre (HOW-WE-ROLL).

(1)GR-KUNN-FERD

Vi begynte med å kartlegge vår kunnskap om prosjektarbeid så langt og beskrive hva som er relevant for denne oppgaven.

Dzemil: Min kunnskap om prosjektarbeid kommer for det meste fra tidligere år på skolen. Der har jeg opparbeidet meg grunnleggende kunnskap om akademisk tenkning og skriving. Samt en god del IT-kunnskap , det vil si HTML5, CSS og CANVAS. Generelt har jeg en del erfaring når det kommer til samarbeid i grupper, hvordan man skal forholde seg til andre mennesker og jobbe sammen for å oppnå et gitt mål. Jeg kan se for meg at mye av denne kunnskapen kommer til å være relevant for denne gruppeoppgaven, ettersom mye av arbeidet foregår i gruppen der vi sammen skal lage en applikasjon og skrive en rapport underveis.

Jonas: Min tidligere kunnskap om prosjektarbeid er minimal. Jeg kommer direkte fra militæret hvor samarbeid var et av hovedfokusene opp gjennom opplæringen og utplasseringen min. Jeg har mesteparten av mine relevante erfaringer fra videregående hvor vi drev med koding i grupper. Fra det har jeg god kunnskap rundt html og css. Jeg har i tillegg en god kunnskap om php og mysql. Jeg har tidligere kodet sider med muligheter for profil opprettning, betaling av annonser gjennom paypal, søknader opp imot annonsene, kontaktskjemaer og andre detaljer tilhørende en slik side. Jeg nevner hovedelementene jeg skapte gjennom stor bruk av php og js.

Mathias: Erfaring når det kommer til prosjektarbeid begrenser seg egentlig til dette faget her. Vært en del av team og grupper i arbeidslivet og tidligere skolegang, men ikke noe så spisset som dette. På kode delen begynner jeg få øke kontroll på HTML og CSS, mens JS kunnskapene bør jobbes mer med. Når det kommer til akademisk skriving føler jeg selv det er noe jeg har grei kontroll på, men som selvfølgelig kan jobbes mer med. Dette vil være en styrke i gruppearbeidet, da jeg kan ta mer ansvar på denne delen.

Daniel: Mine tidligere erfaringer om prosjektarbeid har jeg drevet mye med i videregående skole i Jessheim, men dette var kun grupper fra minimum 2 til 3 per gruppe. Men jeg har videreutviklet meg her på universitetet i Agder og har drevet med prosjektarbeid i større gruppe. Gjennom mine tidligere skolegang har jeg også gått veldig dyktig gjennom akademisk skriving og kildebruk. Jeg har også hatt faget teknologi og forskning som har gitt meg en stor fordel på koding, men har også lært litt av min far som er dataingeniør. Fram til nå innen mine IT erfaringer er jeg ganske komfortabel med å kode HTML, github, Javascript, CANVAS og pyret; men ønsker å lære mer.

Mohammed: Tidligere erfaring om prosjektarbeid har jeg erfart noe i min tidligere bachelorutdanning i bioingeniør, men dette er ikke helt å sammenligne i faget her. Har jobbet som bioingeniør i nesten ti år og dette er ikke min felt for å si det slik. Jeg har ikke så veldig mye erfaring om tekniske innhold når det kommer til koding, men har mine interesser om å lære mer om HTML, github, javascript og mer til å kunne bidra ytterligere i et prosjektarbeid.

(2) FS-ANALYSE

Vi fikk i oppgave å utvikle en applikasjon som kunne vært nyttig for organisering av samarbeid i grupper bedre. Etter at vi fikk en god forståelse for kravspesifikasjonen kom vi fram til at samarbeid i grupper kan forbedres på flere forskjellige måter. Derfor valgte vi å se på hvilke funksjoner en applikasjon som "Felles-Skapet" kan ha i tillegg til HOW-WE-ROLL (HWR). Den ene funksjonen tenker vi kan gjøre det lettere for en prosjektgruppe å holde kontroll over fremmøteprosenten til gruppemedlemmene. Den andre kan hjelpe med å organisere gruppemøter utenfor skole/arbeid. Ved å representere ideene våre gjennom

brukerhistorier greide vi å få en bedre forståelse for oppgaven, og hvilke muligheter som ligger til grunn hos en applikasjon som har som mål å forbedre samarbeid i grupper.

Brukerhistorie nr .1

Som bruker ønsker jeg å registrere når enkeltpersoner fra gruppen møter opp på planlagte møter. Slik at gruppen kan holde kontroll over fremmøteprosenten til gruppemedlemmene, og holde folk ansvarlig dersom de ikke respekterer kravet som er satt for oppmøte i gruppe kontrakten.

Løsningsforslag 1:

Lage en web applikasjon som gjør det mulig å registrere oppmøte til gruppemedlemmer på planlagte møter. Gruppelederen får en gruppekode, som gir muligheten til å registrere oppmøte. Applikasjonen holder kontroll over fremmøteprosenten etter et eller flere møter og kan representere fremmøteprosenten til de med gruppekode etter ønske.

Brukerhistorie nr .2

Som bruker ønsker jeg å bruke en kalender for å sette opp tilgjengeligheten min for en periode, slik at prosjektgruppen min lettere kan planlegge møter utenfor skolen/arbeidsplassen.

Løsningsforslag 2:

En slik funksjon hadde tatt i bruk en kalender, der gruppemedlemmene hadde valgt ut spesifikke dager i løpet av en måned der de er tilgjengelig. Gruppemedlemmene hadde fått tildelt en "gruppekode" som hadde blitt brukt for å sette tilgjengeligheten. De hadde brukt den samme koden for å tilgang til siden der de kan se dager alle eller et mindretall av gruppen kan møtes.

Brukerhistorie nr.3 (HOW-WE-ROLL)

Som bruker ønsker jeg at at prosjektgruppen skal få bedre oversikt over medlemmenes vaner og eventuelt uvaner, for å potensielt kunne planlegge prosjektarbeidet bedre.

Løsningsforslag 3:

HOW-WE-ROLL samler inn informasjon ved at brukeren bruker slidere for å avgjøre svar på spørsmål om deres vaner (og eventuelt uvaner). Til slutt blir alle svar på hvert av spørsmålene presentert som sirkler på en linje.

Begrunnelse for valg av funksjon til applikasjonen

Vi tenker at en funksjon som registrerer oppmøte til gruppemedlemmene hadde vært nyttig for enhver gruppe som har en fastsatt oppmøteprosent. Ettersom det ikke alltid er like lett å holde kontroll over fremmøteprosenten til medlemmene over en lengre periode. En slik applikasjon hadde gjort medlemmene mer bevisst på om de møter

opp eller ikke ved at gruppen har bedre kontroll over det. I tillegg vi tenker at en funksjon som bruker en kalender for å planlegge møter, hadde vært nyttig for grupper som har behov for å møtes utenfor skole/jobb, og som sliter med å finne dager å møtes på som passer for alle. Den hadde gitt gruppen et bedre overblikk over mulige dager å planlegge et møte på. I vår variant av "Fellesskapet" går vi videre med HOW-WE-ROLL funksjonen. Med tanke på at det er en obligatorisk del av oppgaven. En prosjektgruppe vanligvis jobber bedre sammen når gruppemedlemmene er godt kjent med hverandre. HOW-WE-ROLL gir prosjektgruppen en mulighet til å bli bedre kjent, samtidig som det kan bidra med å effektivisere arbeidsprosessen ut fra gruppemedlemmenes vaner og uvaner.

(3) Innsamling og representasjon av data

Eksempler på hvordan en gruppe kan samle inn informasjon (data) om gruppemedlemmene:

- En måte å samle inn data på er gjennom spørreundersøkelser med fastsatte spørsmål og svar for å lettere å kunne analysere data fra flere folk. En spørreundersøkelse kan gjennomføres på papir eller digitalt.
 - **Intervjuer** kan foregå ansikt til ansikt eller digitalt der intervjueren stiller spørsmål og noterer svarene. (Hellevik, 2015).
 - **Feltstudier** er en metode der er "undersøkeren" er fysisk tilstede under situasjoner som er relevante for undersøkelsen, og observerer hvordan personer handler uten at de vet noe om det. (Wikipedia, 2022).
-
- **Workshop** er en måte å utforske et tema eller en prototype i en gruppe. Målet er at deltakeren skal bli godt kjent med problemstillingen, for å kunne diskutere og stille spørsmål. (Frøyen, 2017).
 - **Web-form** bestående av forskjellige input-elementer gir et godt grunnlag for å hente inn informasjon. Her kan man fastsette "spørsmålene" på forhånd, og distribuere form-en enkelt digitalt.

Det er viktig å representer dataen på en fornuftig måte, for å kunne få en bedre forståelse for resultatet. Det finnes flere måter å representer data på:

- **Søylediagram** er en måte sammenligne kategoriske verdier på
- **Linjediagram** brukes for å vise utvikling og sammenheng i data
- **Sektor-diagram** er nyttig til å sammenligne flere kategorier (Ramsøy, 2021).
- **Tabell** representer data i rader og kolonner
- Representere data som **tekst** ved å skrive ned resultat i avsnitt og punkter (Nguyen, 2022).

Vår løsning av en input-side som skal samle inn data i web-applikasjonen vår (HOW-WE-ROLL):

Hva er ditt navn?

Når står du opp?

7:08:00 10:00 12:00<

Hvordan løser du konflikter?

Liker å jobbe m andre Liker å jobbe alene

Når ønsker du å bli kontaktet?

Oppsatt moto Ring når du vil

Hvordan ønsker du å diskutere?

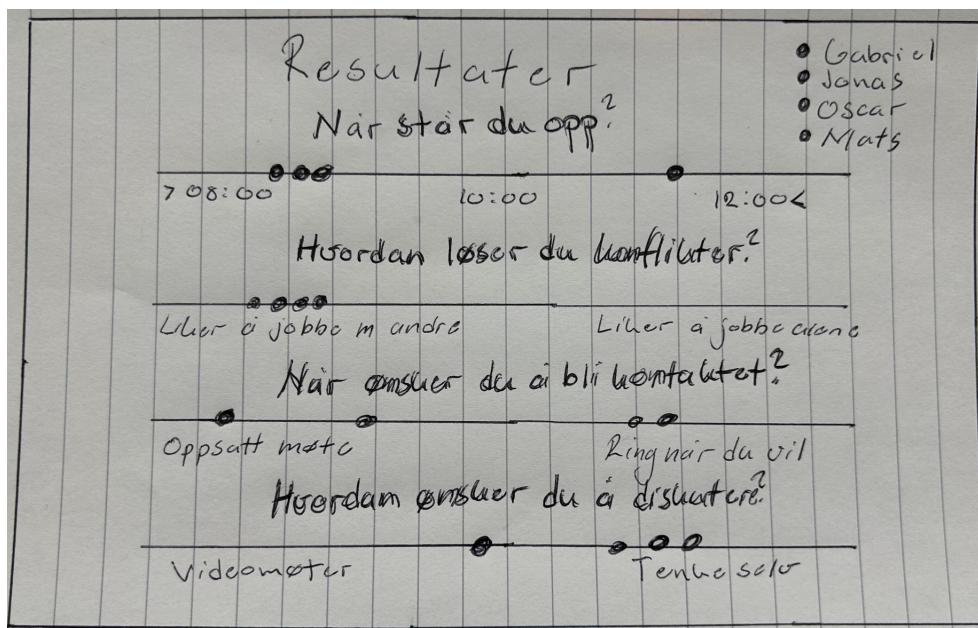
Videomøter Tenke selv

Send

Figur 1

Brukeren skal skrive inn navnet sitt og bruke slidere for å tilpasse svaret på spørsmålene om vaner og uvaner, som vi tenker er aktuelle for gruppearbeidet. Deretter sende inn skjemaet.

Output-siden som skal representeres resultatene:

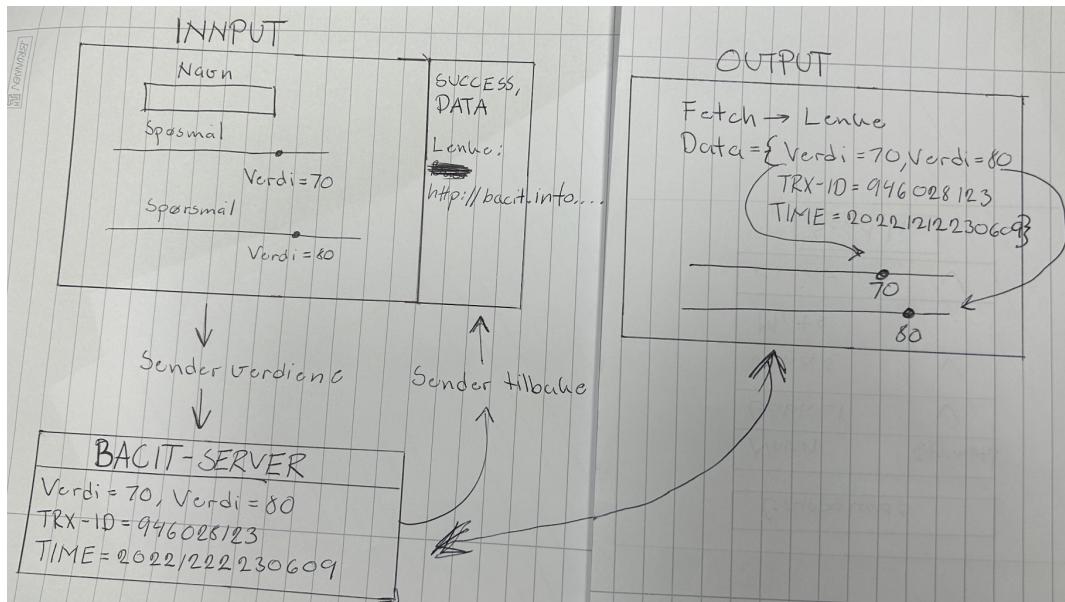


Figur 2

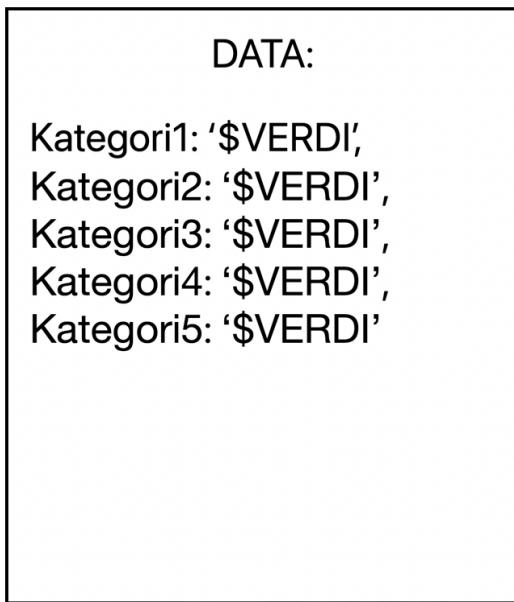
Resultatene representeres ved at alle svarene på hvert av spørsmålene blir vist. Hver sirkel har sin egen farge, og tilhører et gruppemedlem. Slik at gruppen enklest mulig kan analysere resultatet og bruke det til å forbedre gruppearbeidet.

(4)FS-SYSTEM og FS-SYSTEM-STEG

For å kunne lage applikasjonen baser på kravspesifikasjonene til HWR trenger vi først å se for oss hvordan system skal fungere i sin helhet. Figur 3 viser et FS-SYSTEM og prosessen som transformerer input om til output. Vi forklarer denne prosessen gjennom FS-SYSTEM-STEG.



Figur 3



Figur 3.1

Innput siden i figur 3 bruker slidere slik at brukeren skal kunne svare på spørsmålene. Hver slider representerer en verdi ut fra hvor sirkelen ligger på

slideren. Her vil hver slider korresponder med hver sin variabel på output-siden vår. Når brukeren har fylt inn data og trykker på “send-knappen”, aktiveres en funksjon som gjør om disse verdiene og setter dem inn i en datamodell og produserer en URL.

Figur 3.1 er en illustrasjon av datamodellen vår, den registrerer hver verdi som hører til en kategori. “KategoriX” referer til hver nye slider på siden og “\$VERDI” er verdien brukerens slider har ved innsendelse. Å bruke “\$VERDI” vil fungere dårlig ettersom det ikke vil gi muligheten til å registrere ulike verdier til flere sliders. Derfor må sliderne få en egen ID som: “slider1”, “slider2” osv. Slik at vi skal kunne hente verdiene de ulike sliderne gir.

Videre i figur 3 ser vi hvordan data kan bli sendt til og fra “BackIT-serveren”. For å sende data til serveren bruker vi en “fetch-post” metode, samtidig som datamodellens data blir gjort om til en “json-string”. Serveren vil ved en vellykket innsendelse sende tilbake dataen som originalt ble sendt med to ekstra verdier: “txid” og “date”. Disse to verdiene vil gjøre det mulig for systemet å automatisk lage en URL til serveren, som kan kopieres og brukes på output-siden for å hente ut informasjonen.

Videre i figur 3 kan vi se output-siden. Prosessen med å hente data ut data starter med en “fetch-GT” forespørsel til “BackIT-serveren” med de/den korrekte URL-en/e. Dersom denne prosessen er vellykket, vil serveren svare med all data den har lagret i “json-filen”. For å kunne transformere input til output trenger vi noen funksjoner som kan tegne ulike elementer på et canvas. Det er viktig med en funksjon som kan tegne sirkler ut fra verdien som er lagret innenfor hver datakategori for hver innsendelse. Funksjonen bruker verdien som sin egen x-akse verdi. Det gjøres ved bruk av “for loop”, som gjør det mulig å gjenta funksjonen flere ganger ut fra hvor mange elementer og objekter som er representert. En repetisjon av “loopen” vil tegne alle sirklene for en bruker med en spesifikk avstand på y-aksen for hver nye datakategori.

Output-siden må også kunne tegne streker under sirkelen for bedre visualisering av hvor sirkelen ligger. Her vil en funksjon for tegning av streker kunne fungere ved å følge y-verdien til sirklene og bli repeteret like mange ganger som det er objekter i ett enkelt element, minus irrelevante objekter. For at dette skal fungere ut fra HOW-WE-ROLL kravspesifikasjonen, må også sirklene kunne markeres med en farge eller en annen måte som skiller brukernes ulike verdier.

Selektorer i stilark blir navngitt etter deres oppgave. Vi behøver nok noe CSS for sliderne. Dette kan enkelt navngis “input” ettersom knappen som brukes for å sende ikke vil være en “input”, men bare en “button”. Ellers er det ingenting som unødig vil endres ved bruk av denne selektoren. Dersom vi ønsker at innholdet på siden skal være sentrert trenger vi bare å angi “body” som en selektor og gi den nødvendige

“properties”. Javascript funksjoner blir navngitt etter deres funksjon. Funksjoner for å tegne sirkler hadde blitt navngitt “sirkelTegning” eller “drawCircle”. Det samme gjelder for andre funksjoner som farger eller linjer. Funksjonen på input-siden som aktiveres når knappen for innsending blir trykket på “(onclick())”, vil bli navngitt “send()”.

(5) Valg av Funksjoner

Valg av funksjoner er dokumentert i koden. (Disse kommentarene er ikke representert i figurene i dette dokumentet).

(6) FS-TESTER

Vi implementerte funksjoner for hvert steg i FS-SYSTEM-STEG og gjennomførte tester for hver funksjon for å sjekke at de tilfredsstilte kravspesifikasjonen.

Henting av verdier

Det første vi måtte teste var funksjonen som henter verdien fra slider-elemente. Dette testet vi med å gi sliderne en satt verdi på 250 og logget i konsollen hva variablene innenfor funksjonen i figur 4.1 ble når de ble sendt. Figur 4.2 viser at dette fungerte slik det skulle. Figur 4.2.1 viser hvordan sliderne ser ut.

```
<input type="range" min="50" max="450" value="250" class="slider" id="verdi2">

function send() {
    var verdi1 = document.getElementById("verdi1").value;
    var verdi2 = document.getElementById("verdi2").value;

    const json = {
        verdi1: verdi1,
        verdi2: verdi2
    };console.log(json);console.log("Verdi1: "+verdi1);console.log("Verdi2: "+verdi2);
```

Figur 4.1



Figur 4.2

INPUT

Verdi 1 TEST



Verdi 2 TEST



Figur 4.2.1

Sending av verdier

Videre testet vi en funksjon som kunne sende disse verdiene til “BackIT- serveren”, for å se at det ble gjennomført riktig og at informasjonen serveren sendte tilbake kunne gjøres om til en lenke brukeren kan kopiere.

Figur 4.3 viser “fetch-post” metoden som brukes for å sende verdiene fra slider elementene til “BackIT-serveren” som “json”. Verdiene blir sendt innenfor “body:” og omgjort til “json”. Serveren gir en respons for hver innsendelse med “success: data”, der data inneholder all informasjonen for den gjeldende innsendelsen. Figur 4.4 viser at serveren korrekt har respondert med begge verdiene som ble sendt: “trxic” og “time”. Dette er at hver innsendelse skal få en gjeldende URL. Vi “logger” også URL-adressen vi lager for å sjekke at den blir opprettet riktig og gjør den om til en variabel som senere kan bli printet ut på selve siden.

```
fetch("https://bacit.info/", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify(json),
}).then((response) => response.json())
.then(data) => {
  console.log('Success:', data);
  console.log('Lenke:', 'https://bacit.info/hwr/' + data.time + '-' + data.trxic + '.json');
  const lenke = ('Lenke:', 'https://bacit.info/hwr/' + data.time + '-' + data.trxic + '.json');
```

Figur 4.3

```

▶ {verdi1: '250', verdi2: '250'}
Verdi1: 250
Verdi2: 250
Success:
▶ {verdi1: '250', verdi2: '250', trxid: '3f32f6c9bd9d0ed5ea8e53a0f504d6a2', time:
  '20221214161806'}
Lenke: https://bacit.info/hwr/20221214161806-3f32f6c...json
> Lenke: https://bacit.info/hwr/20221214161806-3f32f6c9bd9d0ed5ea8e53a0f504d6a2.json

```

Figur 4.4

Innhenting av data fra server

Videre testet vi funksjonen for innhenting av data fra serveren. Under i figur 4.5 ser vi den originale funksjonen vi valgte for å hente data fra serveren. Vi har lagt til en ekstra URL med data her for å bevise at funksjonen tilfredsstilte testene bedre.

```

Promise.all([
  fetch('https://bacit.info/hwr/20221214161806-3f32f6c9bd9d0ed5ea8e53a0f504d6a2.json'),
  fetch('https://bacit.info/hwr/20221214160501-8077ed15e3154c5197c5245c3ba235f3.json')
])
  .then((data) => {
    console.log('Success:', data);
  })

```

Figur 4.5

Vi tester “Promise.all - fetch” funksjonen for å hente data fra de to URL-ene som er oppgitt under. Som vi ser på den første “indexen” i “arrayet” i figur 4.6, henter serveren korrekt de samme verdiene som ble sendt og “logget” i figur 4.4. Dette beviser at funksjonen tilfredsstiller testen. Da vi kom lengre i prosjektet endret vi funksjonen for innhenting av data fra serveren til det som er kodet i figur 4.7. Dette er grunnet bedre optimalisering ettersom den nå ikke trenger å gå igjennom “fetch” for hver enkelt lenke, men henter alt på en “fetch” gjennom “arrayet”.

```

Success: ▶ (2) [{}], [{}]
  ▼ 0:
    time: "20221214161806"
    trxid: "3f32f6c9bd9d0ed5ea8e53a0f504d6a2"
    verdi1: "250"
    verdi2: "250"
    ▶ [[Prototype]]: Object
  ▼ 1:
    time: "20221214160501"
    trxid: "8077ed15e3154c5197c5245c3ba235f3"
    verdi1: "438"
    verdi2: "437"
    ▶ [[Prototype]]: Object
  length: 2
  ▶ [[Prototype]]: Array(0)

```

Figur 4.6

```
const urls = ["https://bacit.info/hwr/20221214161806-3f32f6c9bd9d0ed5ea8e53a0f504d6a2.json",
  "https://bacit.info/hwr/20221214160501-8077ed15e3154c5197c5245c3ba235f3.json",
];
Promise.all(urls.map(url =>
  fetch(url).then(resp => resp.json())
)).then(data => {
  console.log('Success:', data);
```

Figur 4.7

Fremstilling av data

Det siste som ble testet, er metoden for fremstilling av dataen som er blitt registrert på input-siden og lagret på “BackIT-serveren”. Slik vi ønsket å fremstille det kan man se i Figur 2.

Figur 4.8 og 4.9 viser den første versjonen av dette. Denne versjonen ga oss ønsket resultat ved bruk av et nytt canvas for hvert objekt i arrayet. I figur 4.9 ligger beviset for at denne metoden fungerer på en “ok” måte. Alle verdiene innenfor elementene blir tegnet på riktig sted på strekene og blir tegnet på riktig linje ettersom vi bruker to “canvas-er” som i denne testen er tilhørende de to datakategoriene på input-siden. Vi bruker “Filltext” innenfor hver funksjon for å bevise synlig at hver sirkel blir tegnet riktig sted i denne testen. Som du ser er den oransje og grønne sirkelen tegnet på 250 som er verdien innsendt i Figur 4.3 og “date” stemmer også med denne innsendelsen. Den blå og rosa sirkelen er en annen innsendelse som dobbeltsjekker at dette fungerer som det skal.

```
let fulldata = data.length;
for (let i = 0; i < fulldata; i++) {
    console.log(fulldata);

var canvas1 = document.getElementById('canvas1');
const canvas1_ctx = canvas1.getContext("2d");
canvas1_ctx.beginPath();
canvas1_ctx.moveTo(50, 100);
canvas1_ctx.lineTo(450, 100);
canvas1_ctx.stroke();
canvas1_ctx.beginPath();
canvas1_ctx.arc(data[i].verdi1,100, 8, 0, 2 * Math.PI, false);
canvas1_ctx.fillStyle = 'hsl(' + 360 * Math.random() + ', 50%, 50%)';
canvas1_ctx.font = "16px Arial";
// Write the text to the right of the circle
canvas1_ctx.fillText("Verdi: "+data[i].verdi1, data[i].verdi1, 90);
canvas1_ctx.fillText("Date: "+data[i].time, data[i].verdi1 - 50, 130);
canvas1_ctx.fill();
canvas1_ctx.lineWidth = 2;
canvas1_ctx.strokeStyle = 'black';
canvas1_ctx.stroke();

var canvas2 = document.getElementById('canvas2');
const canvas2_ctx = canvas2.getContext("2d");
canvas2_ctx.moveTo(50, 100);
canvas2_ctx.lineTo(450, 100);
canvas2_ctx.stroke();
canvas2_ctx.beginPath();
canvas2_ctx.arc(data[i].verdi2, 100, 8, 0, 2 * Math.PI, false);
canvas2_ctx.fillStyle = 'hsl(' + 360 * Math.random() + ', 50%, 50%)';
canvas2_ctx.font = "16px Arial";
// Write the text to the right of the circle
canvas2_ctx.fillText("Verdi: "+data[i].verdi2, data[i].verdi2, 90);
canvas2_ctx.fillText("Date: "+data[i].time, data[i].verdi2 - 50, 130);
canvas2_ctx.fill();
canvas2_ctx.lineWidth = 2;
canvas2_ctx.strokeStyle = 'black';
canvas2_ctx.stroke();

}
});
```

Figur 4.8

TEST av Verdi1



TEST av Verdi2



Figur 4.9

Denne versjonen av output-siden var ikke god nok for oss som følge av dårlig automatisering og forbedringspotensiale på nesten alle felt. Derfor gikk vi over til en ny versjon som bruker funksjoner og “for loops” på en bedre måte. I Figur 4.10 kan vi se de første funksjonene for å lage sirkler og streker til den nye output-siden. For å teste at disse fungerer som de skal må vi først gå igjennom “for loopen” som bruker disse.

```
function drawCircle(x, y, color, name) {
    var canvas = document.getElementById('canvas');
    var ctx = canvas.getContext("2d");
    ctx.beginPath();
    ctx.lineWidth = 2;
    ctx.fillStyle = color;
    ctx.strokeStyle = 'black';
    ctx.arc(x, y, 8, 0, Math.PI*2, true);
    ctx.font = "16px Arial";
    ctx.fillText(name, 620, y+5);
    ctx.stroke();
    ctx.fill();
    ctx.closePath();
}
function drawDashedLine(y) {
    // Get the canvas element and its 2D context
    var canvas = document.getElementById('canvas2');
    var ctx = canvas.getContext("2d");
    // Draw the lines
    ctx.beginPath();
    ctx.moveTo(40, y);
    ctx.lineTo(460, y);
    ctx.strokeStyle = 'black';
    ctx.stroke();
    ctx.closePath();
}
```

Figur 4.10

Her i Figur 4.11 kan vi se hvordan den nye “for loopen” ser ut og hvordan den kan tegne sirkler og streker på en bedre måte ser ut. I Figur 4.12 under ser på resultatet på denne koden.

```
let y2 = 150;
for (let i = 0; i < data.length; i++) {
    let color = colors[i]; //Definerer color som colors[i] som altså er farge arrayet og går igjennom denne med +1 for hver gjennomgang av loopen
    let y = 150; //Setter start y verdien
    let j = 0; //Setter start verdien på farge arrayet. (Den første fargen er alltid index 0)
    drawCircle(600, y2, color, data[i].navn); //Tegner sirkelene med navn ved siden av utenfor den andre for let løopen så det bare blir en sirkel per index. Her brukes navn parameteren
    //i sirkel funksjonen som er det brukeren selv har skrevet inn.
    y2 += 20; //Øker y verdien til hver nye sirkel med 20

    let element = data[i]; // Henter det gjeldene elemente

    // Går over propertiesene innenfor det gjeldene elemente
    for (let property in element) {
        if (property === "navn" || property === "trxid" || property === "time") continue; //Hopper over navn, trxid og time properties fordi vi ikke skal tegne de

        let x = element[property]; //Setter x verdien på sirkelene som fører til at den blir riktig plassert på strekene

        let trxid = data[i].trxid; //Gjør det enklere for testing å se at sirkelene tilhører riktige indexer

        drawDashedLine(y); //Tegner strekene med en y verdi som begynner på 150 definer over denne løkken

        drawCircle(x, y, color, ""); //Tegner sirkelene. Her er x verdien definert over.Y verdien vil være det samme som strekene og begynner på 150. Color er definert over.

        console.log(x,y,trxid) //Dette brukes til testing

        y += 100; // Øker y verdien til hver sirkel og strek med 100 så ikke alt blir på en linje
    }
}
```

Figur 4.11



Figur 4.12

Figur 4.13 viser hvordan konsollen logger x-verdien, y-verdien og “trx-iden” til hver av sirkelen som blir tegnet for å teste. Vi kan se at sirklene har riktig x-verdi i henhold til verdien som ble hentet av serveren i Figur 4.6. Y-verdien er også korrekt for det første objektet i hver “index” og har en verdi på 150 som gjør at de vil ligge på samme linjen og det neste objektet er økt med 100 som koden vår sier den skal.

Figur 4.11

```
250 150 3f32f6c9bd9d0ed5ea8e53a0f504d6a2
250 250 3f32f6c9bd9d0ed5ea8e53a0f504d6a2
438 150 8077ed15e3154c5197c5245c3ba235f3
437 250 8077ed15e3154c5197c5245c3ba235f3
```

Figur 4.13

Til slutt måtte vi også ha en funksjon som automatisk kunne lage et nødvendig antall farger. Koden for dette kan du se under i Figur 4.14. Ser vi her på Figur 4.14 burde dette oppfylle kravet vi satt. Funksjonen vil lage en “random” “RGB-farge” og vil lage “data.length” antall farger. “Data.length” er det antallet med forskjellige lenker/innsendelser vi henter. I Figur 4.15 kan man se hvordan vi velger å teste dette. Vi tegner sirkelen som plasseres på siden for å identifisere hvilke farger som tilhører hvem. Som vi ser under i Figur 4.16 viser vi her at testen var vellykket ettersom den riktige RBG koden ligger bak riktig objekt.

```

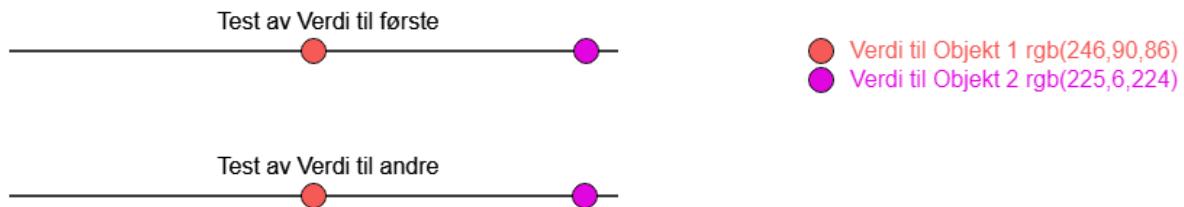
for (let j = 0; j < data.length; j++) {
    // Generate random red, green, and blue values
    let red = Math.floor(Math.random() * 256);
    let green = Math.floor(Math.random() * 256);
    let blue = Math.floor(Math.random() * 256);
    let color = "rgb(" + red + "," + green + "," + blue + ")";
    // Check if the color already exists in the array
    if (!colors.includes(color)) {
        // Add the color if it doesn't exist
        colors.push(color)
    }
}

```

Figur 4.14

```
drawCircle(600, y2, color, data[i].navn+ ' '+ color);
```

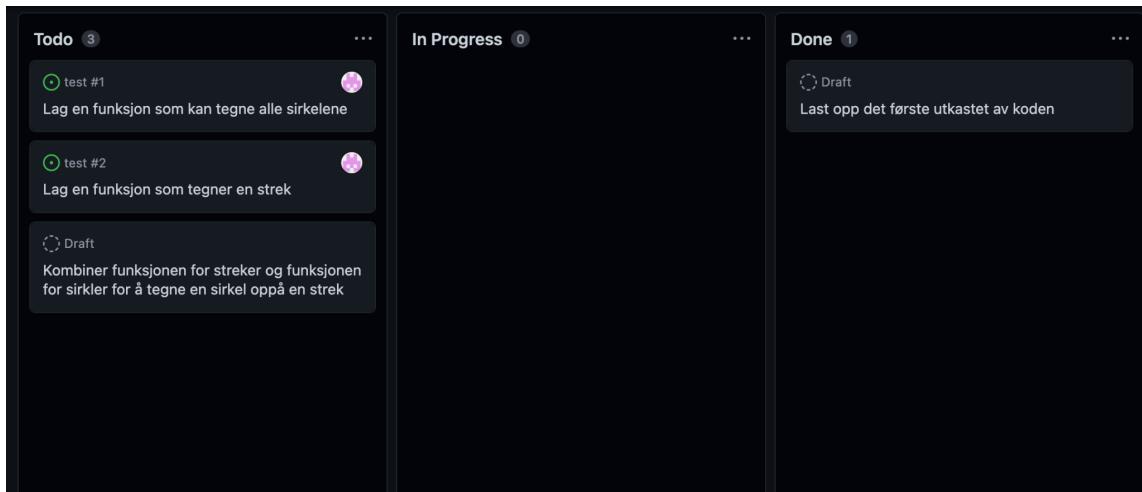
Figur 4.15



Figur 4.16

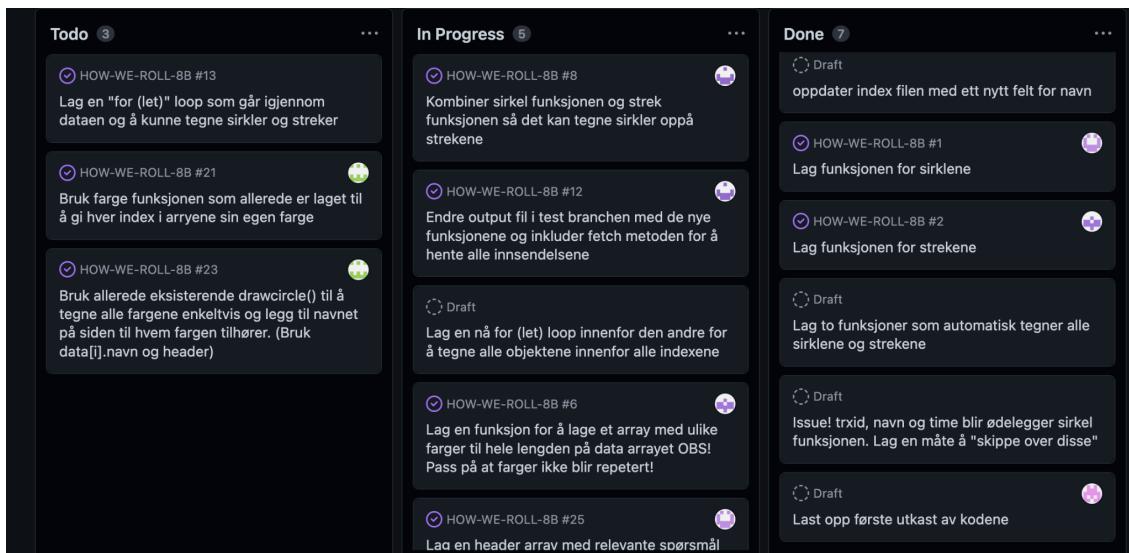
(7) GITHUB-SAMARBEID

Arbeid i dette prosjektet foregikk for det meste via GitHub. Vi startet med å å lage et GitHub Project som alle medlemmene ble invitert til. Innenfor prosjektet satte vi opp et Kanban-bord, for å holde oversikt over hva som må gjøres om hva som har blitt gjort. Et Kanban-bord er en tavle med flere kolonner. Disse kolonnene kan f.eks. hete “To-Do”, “In Progress” og “Done”, som tilsier hvordan du ligger ann med oppgaven. Figur 1 viser hvordan “kanban boardet” vår så ut til å begynne med. Et utkast av koden var lastet opp, men vi trengte funksjoner som kunne tegne sirkler og streker.



Figur 5.1

I figur 5.2 ser vi hvordan “kanban boardet” så ut i en senere fase av prosjektet, når ting begynte å bli mer ferdigstilt. Dersom vi støtte på problemer under kodingen og testingen brukte vi “Issues” funksjonen i GitHub. Det var for å vise at vi hadde møtt på et problem, slik at vi kunne få hjelp eller delegere oppgaven videre til noen andre som kunne løse den. En slik funksjon var nyttig for samarbeidet i gruppen. Den gjorde det mulig å løse problemene vi hadde effektivt og lære av de, ettersom vi fikk en klar ide om hvor utfordringene lå.



Figur 5.2

Issues

Figur 5.3 og 5.4 viser prosessen på hvordan vi løste en “issue” som vi møtte på underveis. Vi ikke greide å få frem alle sirklene fra arrayet med en “for let loop”. @dzme87 prøvde først å få dette til å fungere, men greide ikke å få frem all dataen til alle de ulike objektene innenfor hver av indeks. @jonaslefdal kommenterte på pull requesten om @mathhaug kunne se videre på “issuen” og eventuelt løse problemet. @mathhaug fikk alle elementene til å til å tegnes, men hadde problemer med at det

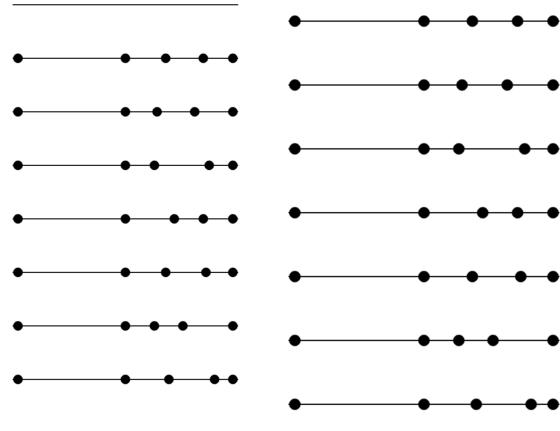
ble for mange linjer. @Kroxxa555 valgte også å se på problemet og løste det med at "for let loopen" hopper over attributene "txid", time og navn, som gjorde at for mange streker ble tegnet siden "loopen" også gikk over disse. I figur 5.5, 5.6 og 5.7 ser man alle versjonene av output-siden.

Figur 5.3

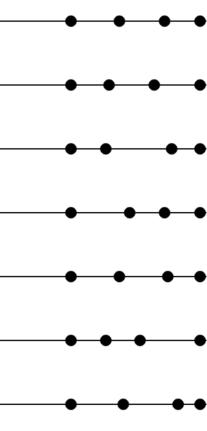
Figur 5.4



Figur 5.5



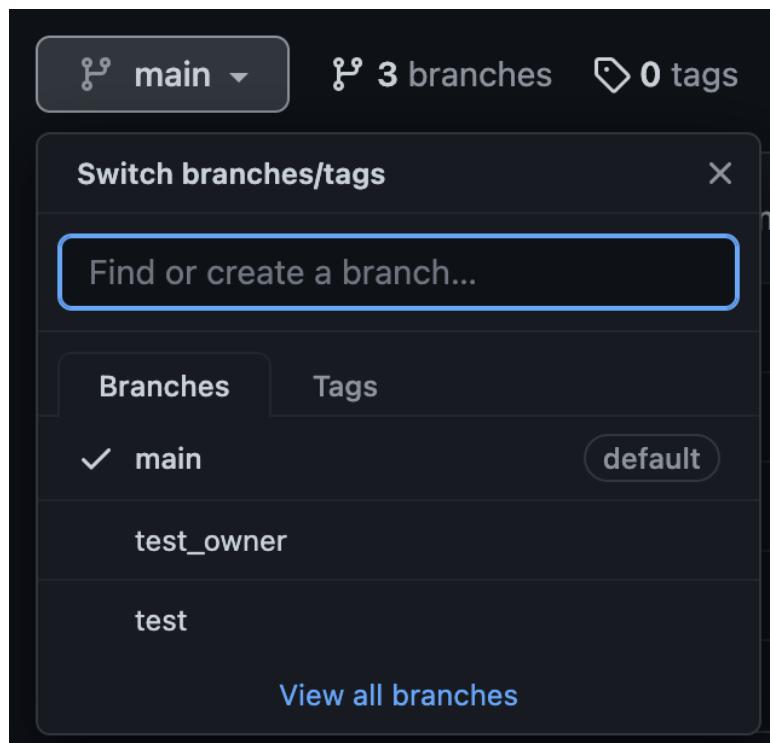
Figur 5.6



Figur 5.7

Fork-metoden

Vi valgte å bruke "fork-metoden" for å jobbe samme med å lage applikasjonen. Figur 5.8 viser hvilke "branches" vi hadde i "hoved-repositoryen". Gruppemedlemmene "forket" "hoved-repositoryen". Vi valgte i tillegg å lage en ny branch i hoved repositoriet navngitt 'test', for å holde "main-branchen" ryddig. Det var her alle sendte inn oppdateringer på koden. Det ble også laget en branch "test owner" siden eieren av "hoved-repositoryen" ikke kunne "forke" den, så han lagde "pull requests" fra denne "branchen" til "test".



Figur 5.8

Figur 5.10 viser hvordan “test-branchen” ble brukt i forhold til “main-branchen” som vi ser på Figur 5.9. Test branchen ble brukt til endring og opplastning av ulike små filer som til slutt skulle implementeres i “main-branchen”

Figur 5.9

Figur 5.10

Konklusjon

I denne rapporten har vi gått gjennom de forskjellige aspektene i prosjektarbeid, direkte knyttet til HWR prosjektet vi ble tildelt. Med kunnskapen vi har tilegnet oss via forelesninger og selvstudium har vi fulgt og dokumentert alle steg fra før vi begynte til refleksjon når arbeidet var gjennomført. Vi startet med å kartlegge hvilke ferdigheter og kompetanse de forskjellige medlemmene tok med seg fra tidligere, før vi gikk sammen i gruppen og fikk en forståelse for hvilke verktøy ol. som trengtes for å oppnå alle kravspesifikasjoner.

Neste steg var så å gjøre analyser om hvilke problemstillinger en slik applikasjon kan være med på å løse. Her lagde vi forskjellige brukerhistorier for å illustrere hvordan dette kunne fungert i praksis. Her mener vi at vi kom opp med tre varierte og gode

forslag, som vi legger frem og diskuterer eventuelle gevinstene av disse. Etter diskusjon i gruppen endte vi opp med en løsning vi valgte å gå videre med.

Videre går vi inn på forskjellige metoder man kan bruke for å innhente data. Her presenteres forskjellige metoder dette kan gjøres på, som alle er effektive på forskjellige måter. Her gikk vi for den metoden som samsvarer mest med kravspesifikasjonene.

Så går vi grundig gjennom stegene og funksjonene vi bruker fra input til output, begrunner valgene våre her og dokumenterer tester vi har gjennomført underveis. Her var det viktig for oss å lage en kode som er "futureproof". Her er koden bak og output siden laget på en slik måte at vi kan supplere med 100 ekstra spørsmål og medlemmer, uten at dette ville ødelagd resultatet. Til slutt har vi drøftet og reflektert rundt positive og negative sider ved bruk av GitHub

Litteraturliste

Frøyen, H. M. (2017, 10. august). Datainnsamling - kvifor og korleis? *UX-bloggen*.
<https://www.usit.uio.no/om/organisasjon/bnt/web/ux/blogg/2017/datainnsamling.html>

Hellevik, O. (2015, 18. mai). *Spørreundersøkelser*. De nasjonale forskningsetiske komiteene.

<https://www.forskningsetikk.no/ressurser/fbib/metoder/sporreundersokelser/>

Wikipedia. (2022, 25. oktober). Field research. I Wikipedia. Hentet 15. desember 2022 fra

https://en.wikipedia.org/w/index.php?title=Field_research&oldid=1118140495

Ramsøy, C. (2021, 21. april). Hva er viktig å huske på når du skal visualisere data? *Business Intelligence*.

<https://www.visma.no/blogg/hva-er-viktig-a-huske-pa-nar-du-skal-visualisere-data/>

Nguyen, L. (2022, 14. september). *10 metoder for datapresentasjon (+ eksempler og 5 gode tips) i 2022*. AhaSlides.

<https://ahaslides.com/no/blog/10-methods-of-data-presentation/>