Forside

IS-304: 2023

Tittel: Visualisering av sensordata fra vann og avløp

| | |
|---|---|
| Emnekode | IS-304 |
| Emnenavn | Bacheloroppgave i informasjonssystemer |
| Emneansvarlig: | Hallgeir Nilsen |
| Veileder | Janis Gailis |
| Oppdragsgiver: | Norkart |

Studenter:


| Fornavn | Etternavn |
|---|---|
| Johannes | Leite |
| Erlend | Lotsberg |
| Karen | Thorsen |
| Joachim | Western |
| Tai-Gwen | Woo |


| | |
|---|---|
| Vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeid uten at dette er oppgitt og at alle referanser er oppgitt i litteraturlisten. | **Ja** |
| Kan besvarelsen brukes til undervisningsformål? | **Ja** |
| Vi bekrefter at alle i gruppa har bidratt til besvarelsen | **Ja** |

# Forord

Etter tre lærerike år ved Universitetet i Agder, nærmer vi oss slutten på vår utdanning. Vi vil derfor takke Universitetet i Agder for mulighetene og erfaringene vi har fått.

Bachelorprogrammet "IT og informasjonssystemer" har bratt læringskurve, og gitt oss mange gode stunder. Kjernegruppen som kaller seg internt for "B+ students", har alltid vært en ambisiøs gjeng som ønsker best mulig læringsutbytte. Derfor har prosjektene vært både motiverende og givende.

I løpet av studietiden har vi også hatt gleden og muligheten av å knytte bekjentskap med mange kunnskapsrike og inspirerende fakultetsansatte og medstudenter. Derfor vil vi takke vår kjære veileder Janis Gailis for at han så potensialet i oppgaven, viste interesse, møtte opp på våre presentasjoner, samt veiledet oss gjennom prosjektet vårt. Vi vil også takke Norkart som tok oss godt imot og pekte oss i riktig retning, og sørget for å gi oss både utstyr og kontorplass. Her fikk vi et innblikk i bedriften og ble kjent med de ansatte. De bistod også med veiledning når vi satt fast med prosjektet, noe som har gitt oss verdifulle innspill til arbeidet vårt.

Vår største takknemlighet går til våre familier og nærmeste,
for håp og støtte de har gitt oss under hele prosessen.

# Innholdsfortegnelse

# Figurliste

# 1. Introduksjon

Dette kapittelet tar for seg en introduksjon av prosjektet, oppgavebeskrivelse, samt visjoner og mål gruppen har for prosjektet.

## 1.1 Bakgrunn

I emnet IS-304 skal gruppen gjennomføre et prosjekt i samarbeid med en ekstern aktør, enten fra offentlig eller privat sektor. Hensikten med prosjektet er å ta i bruk kunnskaper og ferdigheter som er tilegnet så langt i studieprogrammet, og anvende disse i praksis. Studentene skal definere, planlegge, estimere, utføre, teste, følge opp og dokumentere et informasjonssystem- eller informasjonsteknologiprosjekt, og sikre kvalitet i både prosess og produkt (Universitetet i Agder, u.d.).

Gruppen inngikk et samarbeid om et prosjekt med Norkart. Ifølge Norkart er de «…et norsk teknologiselskap som tilbyr markedsledende løsninger innen kommunalteknikk, kart- og eiendomsinformasjon til privat og offentlig sektor» (Om oss, u.d.). Norkart utvikler og vedlikeholder KOMTEK Drift og Vedlikehold (heretter forkortet KDV) som er et system for private og offentlige virksomheter som ønsker å få kontroll på driftsoppgaver, og sikrer virksomhetens troverdighet hos abonnenter og myndigheter (Komtek, u.d.). KDV inneholder kart over vannledninger, avløpsledninger og kummer som binder infrastrukturen sammen (Kurt Pedersen, personlig kommunikasjon, 29. november 2022).

Normatic AS er et selskap som tilbyr automatisert driftskontroll i form av systemer for administrering, regulering og overvåking av ulike typer tekniske anlegg, inkludert sensorer for vann- og avløpssystemer (Normatic AS, u.d.).

En av Norkarts kunder bruker også Normatic sine tjenester, og ønsker at sensordata fra Normatic sine sensorer skal kunne vises i KDV-applikasjonen. Dette vil føre til bedre situasjonsforståelse, da kunden på denne måten vil ha all informasjonen samlet på et kart, og slipper å bla frem og tilbake mellom applikasjoner for å holde oversikt over de forskjellige systemene som er i bruk.

## 1.2 Oppgavebeskrivelse

Gruppens oppgave i Norkart består i å nyttiggjøre sensordataene fra vann- og avløpssystemer og vise disse i et kart eller på et dashbord. Utgangspunktet var å konsumere data fra et test-API produsert av Normatic, og lage en dataflyt for behandling og visualisering av output. I følge Store Norske Leksikon er et API «et grensesnitt som gir direkte tilgang til data og funksjonalitet i et datasystem» (Vihovde, 2022). Løsningen vil være en proof of concept (heretter forkortet POC) for å kunne dra nytte av disse sensordataene i for eksempel Norkart sin KDV-løsning. Oppgaven har et fleksibelt scope, og gruppen har i stor grad stått fritt til å bestemme fremgangsmåte og arkitektur selv.

## 1.3 Visjoner og mål

*Målet for Norkart* med dette prosjektet er å la studentene gjøre forsknings- og utviklingsarbeid på vegne av selskapet, og lage en POC for hvordan sanntidsdata fra et eksternt API kan hentes inn og visualiseres i et kart og/eller dashbord, noe som kan gi verdi for selskapets etablerte KDV-løsning.

*Gruppens mål* for prosjektet er på tre forskjellige områder:

- *Mengdetrening*: for vår egen erfaring og kompetanse, ønsker gruppen å gjennomføre så mye realistisk mengdetrening som mulig hvor man jobber i relevante prosjekter. Gruppen har dermed ikke kun fokus på produktet som blir laget gjennom prosjektet, men også på egen læring og kompetanseutvikling, både innen systemutvikling og styring av prosjekt.
- *Nytteverdi for partnerbedriften*: gruppen ønsker naturligvis også at produktet som utvikles skal gi en nytteverdi for Norkart, og at deler av løsningen eller innsikt som oppnås underveis vil kunne brukes inn mot Norkarts KDV-løsning eller andre relevante prosjekter.
- *Akademisk*: gruppen ønsker å anvende kritisk tenkning til å forbedre prosesser og arbeid gjennom prosjektet. Vi ønsker selvsagt også å oppnå et godt resultat i emnet IS-304.

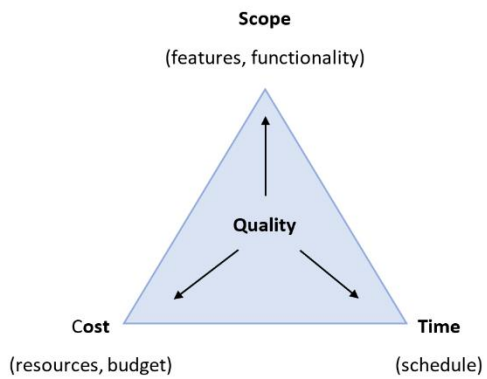
# 2. Prosjektstyring

Dette kapittelet gir innsikt i hvordan gruppen har hatt kontroll over prosjektet. Arbeidsmetodikken som ble brukt var Scrum, og for styring av prosjektet ble Azure DevOps benyttet som verktøy. Usikkerhet og risiko ble redusert gjennom risikovurdering.

## 2.1 Prosjekttrekanten

I et prosjekt er det flere faktorer som har stor betydning for kvaliteten som er mulig å oppnå. Ett verktøy for å få oversikt over og forstå disse faktorene er prosjekttrekanten. Prosjekttrekanten består av en trekant hvor sentrum av trekanten representerer kvalitet, og hver av sidene representerer tre av faktorene som kan påvirke kvaliteten i prosjektet: tid, kostnad og omfang (scope) (Microsoft, u.d.). Se Figur 1 for mer detaljer.

Hensikten med å representere dette som et triangel er for å vise at alle tingene henger sammen og vil påvirke hverandre – for eksempel, hvis man begynner å gå tom for tid i et prosjekt kan man enten øke budsjettet, redusere scopet, eller godta en lavere kvalitet.




*Figur 1. Prosjekttrekanten. (egenprodusert)*

I dette prosjektet har en av faktorene vært helt fastlåst, nemlig **tid**. Fristen for å bli ferdig med prosjektet er satt av universitetet, og må overholdes av gruppen. De andre delene av trekanten er i større eller mindre grad modifiserbare.

Gruppen har hele veien ønsket å levere et produkt og et prosjekt av høy kvalitet. For å realisere målene fra kapittel 1 må kvaliteten i både resultat og prosess være høy. Dermed blir også **kvalitet** en relativt låst faktor.

Betydningen av dette er at gruppen står igjen med to virkemidler for å kontrollere kvaliteten – *scope* og kostnad.

Siden dette er et universitetsprosjekt som ikke har noe nevneverdig monetært budsjett, er det mer korrekt i dette tilfellet å endre begrepet kostnad til for eksempel ressurs, hvor ressurs refererer til de arbeidstimene gruppemedlemmene klarer å produsere i løpet av prosjektet. På denne måten kan man regulere kvaliteten i prosjektet ved å legge inn flere arbeidstimer.

Til slutt kommer *scope*. Selv om gruppen i noen grad har kunnet regulere ressursbruken (bruke flere eller færre arbeidstimer), er det i hovedsak scopet som har vært den avgjørende faktoren for å oppnå ønsket kvalitet. I starten av prosjektet, når det skulle fastsettes et Minimum Viable Product (MVP), ble det tatt et bevisst valg om å legge opp til et relativt lite scope i første omgang, for at MVP skulle holde ønsket kvalitet. Deretter kunne man modulært bygge ut funksjonalitet, helt frem til man gikk tom for tid. Et MVP inneholder de viktigste funksjonene som trengs for at systemet skal kunne tas i bruk (Babych, 2021).

Det er likevel viktig å huske på at prosjekttrekanten ikke tar hensyn til alle mulige faktorer som kan påvirke kvalitet, men bare er en forenkling av virkeligheten. Den er etter gruppens mening likevel et nyttig verktøy, og blir blant annet brukt til å planlegge arbeidsmengde og tilgjengelige ressurser i prosjektet.

## 2.2 Arbeidsstruktur og samarbeid

For å sikre et godt samarbeid ble det holdt et gruppemøte for å avklare tanker om arbeidsstruktur og forventninger.

I løpet av studiet har gruppen arbeidet tett sammen på flere prosjekter og oppnådd positive resultater. Medlemmene i gruppen hadde derfor høye forventninger til hverandre. Under møtet ble det bekreftet at hvert enkelt medlems ambisjoner og motivasjon fremdeles var på et høyt nivå, og gruppen så dermed for seg et godt samarbeid i det kommende prosjektet.

Videre ble det enighet om å bruke forventet arbeidsomfang i emnet som utgangspunkt for hvor mye ressurser som var tilgjengelig for prosjektet. Gruppen tok utgangspunkt i at 30 studiepoeng utgjør en 37,5 timers arbeidsuke, og to tredjedeler av dette er lik 24,75 timer, som utgjør litt over tre arbeidsdager. Basert på dette ble det bestemt å arbeide på kontoret hos Norkart i åtte timer om dagen, tre dager i uken, med mulighet for mer dersom omstendighetene tilsa at det var nødvendig. Denne rutinen har gitt gruppen både forutsigbarhet og god struktur på arbeidsdagene. Gruppen har også lært gjennom erfaring at fysisk tilstedeværelse og tett samarbeid legger til rette for kommunikasjon, kunnskapsdeling, bedre gruppedynamikk og lettere tilgang på støtte.

Det ble valgt en gruppeleder, som har ansvar for kommunikasjon med Norkart og universitetet, og en Scrum Master som vil tilpasse og veilede gruppen i bruken av Scrum-rammeverket etter gruppens behov. Scrum-roller blir mer utdypet i neste kapittel om Scrum.

Formålet med bestemmelsene fra møtet var å legge til rette for effektivt samarbeid og arbeidsprosess. For å formalisere bestemmelsene ble det utarbeidet en gruppekontrakt, se Vedlegg A.

## 2.3 Scrum

Scrum er et rammeverk som brukes i mange organisasjoner som jobber med informasjonssystemer, og kjennetegnes av en agil tilnærming til arbeid hvor man arbeider i korte sprinter, og kontinuerlig forsøker å forbedre både produktet og selve arbeidsprosessen (Scrum.org, u.d.). Scrum-rammeverket består hovedsakelig av tre sentrale elementer: *Scrum Roles*, *Scrum Activities* og *Scrum Artifacts*. Videre vil vi beskrive hvordan gruppen har benyttet rammeverket og hvilken nytte det har hatt for oss.

### 2.3.1 Scrum Roles

I Scrum har man tre roller*: Scrum Master*, *Development Team* og *Product Owner* (McKenna, 2016, s.27). Hver rolle har sin funksjon, og samarbeidet mellom rollene er med på å sikre fremdrift og kvalitet i prosjektet.

Scrum Master: I begynnelsen av prosjektet ble en person utpekt til å være Scrum Master, da vedkommende hadde stor interesse for rollen. En viktig del av rollen er å sikre riktig bruk av det agile rammeverket, og å sørge for høy grad av produktivitet og effektivitet (McKenna, 2016, s.49-53). Scrum Master har i tillegg håndtert hindringer, konflikter, planlagt møter og dokumentert arbeid slik at teamet kan fokusere på utvikling av produktet.

Development Team (teamet): Alle medlemmene i gruppen deltok som en del av utviklingsteamet og hadde ansvar for å bidra til utviklingen av produktet. Gruppemedlemmene hadde ulike interesser og spesialiseringer, inkludert backend-utvikling, frontend-utvikling, UX Design og prosjektledelse. Den brede kompetansen var en stor fordel i de ulike fasene av prosjektet, fra kartlegging av brukerbehov til utvikling av design og implementering av kode, da vi kunne dra nytte av hverandres styrke og dekke for hverandres svakheter. Samtidig som hvert medlem hadde sitt område for spesialisering, arbeidet gruppen tett sammen for å nå et felles mål, og sikret at alle oppgaver ble utført på en effektiv og kvalitetsbevisst måte.

Product Owner: Produkteieren, som i dette tilfellet er Norkart, har ansvaret for å bestemme hva som skal produseres og hva som skal inkluderes i Product Backlog. I dette prosjektet har imidlertid ikke Norkart hatt ansvaret for å utvikle en Product Backlog. I stedet har de kommunisert behovene og forventningene til det endelige produktet gjennom jevnlige møter med gruppen. Gruppen har analysert systemet som skulle utvikles og fokusert på å oppfylle kriteriene fra både Norkart og aktuelle brukere. Deretter har gruppen utviklet en Product Backlog basert på disse kriteriene.

### 2.3.2 Scrum Artifacts

I Scrum har man tre "artefakter" som hjelper med å håndtere arbeidet. De tre artefaktene er *Product Backlog*, *Sprint Backlog*¸ og *Product Increment* (Scrum Alliance, u.d.).

Product Backlog: Product Backlog er en liste over arbeidet som skal utføres. Som tidligere nevnt er gruppens Product Backlog basert på kriterier fra både Norkart og relevante brukere. Disse kriteriene ble deretter utviklet videre til brukerhistorier, som ble tildelt en prioritering ved hjelp av MoSCoW-metoden. MoSCoW er et akronym bestående av bokstavene M (Must Have), S (Should Have), C (Could Have) og W (Won't Have), som hver beskriver en prioriteringsgrad (MoSCoW Method, 2022). Hver oppgave i Product Backlog ble knyttet til en spesifikk brukerhistorie, og dette ga en tydelig prioritering av oppgavene og en klar oversikt over hvilken funksjon som skulle arbeides med.

Sprint Backlog: I løpet av hver Sprint overfører man arbeid fra Product Backlog til Sprint Backlog. Sprint Backlog er da ment som listen over arbeidet man skal fullføre i sprinten. Som en del av denne prosessen, brytes arbeidet som overføres ned i mindre oppgaver (Rubin, 2012, s.22). Dette initierer en prosess med å reflektere over omfanget av oppgavene og gir mulighet for å estimere tidsbruken på de enkelte oppgavene (Upland Software, u.d.). Gruppen estimerte tidsbruken for hver oppgave, og dette tillot planlegging av mengden arbeid som skulle inkluderes i sprinten.

Product Increment: I Scrum kaller vi resultatene fra en Sprint for "Potentially Shippable Product Increment " (Rubin, 2012, s.25). Dette betyr rett og slett at det som Scrum-teamet har blitt enige om å gjøre faktisk er fullført i henhold til de kravene som har blitt satt. Med andre ord, det som ble planlagt for sprinten er nå klart og kan potensielt leveres som en del av det endelige produktet. I dette prosjektet ble det satt et mål for hver Sprint, og da sprintmålet ble oppnådd, ville det resultere i et produktinkrement som oppfyller et, eller flere, av brukerhistorienes akseptansekriterier. Som tidligere nevnt er brukerhistoriene basert på krav fra Norkart og aktuelle brukere. Gjennom hver Sprint har vi arbeidet mot og produsert et produktinkrement som oppfyller akseptansekriteriene for brukerhistoriene. Dette sikrer at vi jobber mot å oppfylle de fastsatte kravene i prosjektet.

### 2.3.3 Scrum Activities

Scrum Activites innebærer *Sprint, Sprint Planning, Sprint Review, Daily Scrum* og *Retrospective.*

I dette prosjektet begynte hver arbeidsdag med gjennomgang av Daily Scrum, der gruppemedlemmene ble oppdatert på hva som har blitt gjort, pågående oppgaver og eventuelle utfordringer som måtte løses. Rutinen med Daily Scrum sørget for en effektiv start på arbeidsdagen og holdt gruppemedlemmene oppdatert på prosjektet og hva de andre jobbet med. Det gjorde det også enklere å samarbeide og løse eventuelle problemer sammen.

Lengde på sprintene ble satt til to uker, da dette ble ansett som en lengde som var tilstrekkelig lang for at det skulle være mulig å gjøre en viss fremgang per Sprint, samtidig som de ikke ble så lange at det ble veldig få sprinter og lite kontinuitet. Gruppen hadde et ønske om å få øve på de forskjellige elementene av Scrum gjennom mengdetrening, så å få litt flere gjennomganger ble sett på som nødvendig. Det ble gjort evalueringer underveis om det var hensiktsmessig å endre lengden på

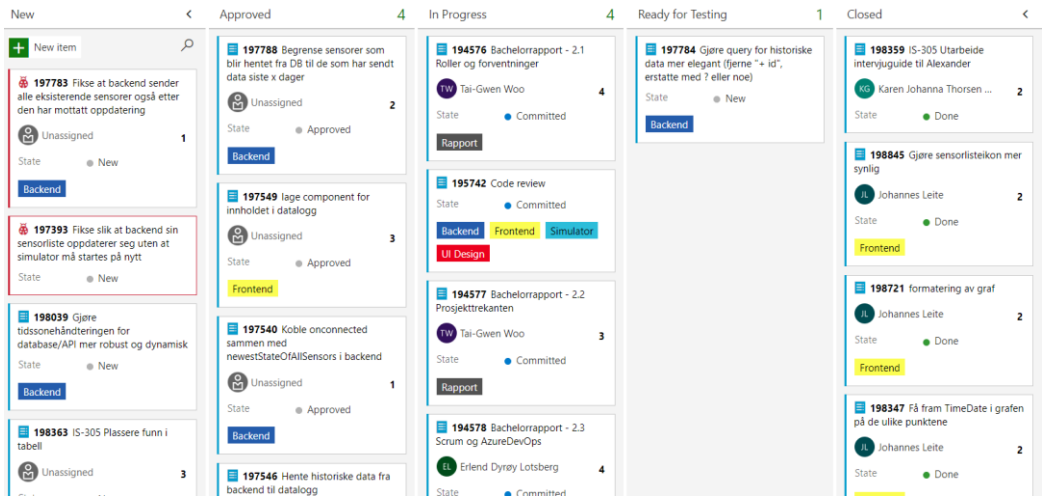
sprintene, men dette ble ikke funnet nødvendig. I starten av hver Sprint har gruppen hatt en Sprint Planning. Basert på brukerhistoriene som ble laget i analyse- og designprosessen laget gruppen en Sprint Backlog. Her ble det både trukket inn oppgaver fra Project Backlog, og det ble holdt relativt omfattende diskusjoner rundt nye oppgaver som kunne opprettes og tas inn i Sprint Backlog.

Med jevne mellomrom gjennomførte gruppen også Sprint Review sammen med «produkteier» (veiledere i Norkart). I Scrum-rammeverket skal disse gjennomføres etter hver Sprint, men på grunn av lengden på sprintene i dette prosjektet var det ikke nok nytt arbeid å presentere etter to uker til at det ble prioritert å ha Sprint Reviews med så høy frekvens – det ble i stedet arrangert Sprint Review med noenlunde jevne mellomrom, etter behov. Veileder fra universitetet deltok også ofte på disse møtene. På Sprint Reviews ble ny funksjonalitet og fremgang vist frem, og det ble gjort diskusjoner rundt veien videre og eventuelle justeringer av allerede gjort arbeid.

Etter hver Sprint hadde gruppen en Sprint Retrospective, hvor man diskuterte tre hovedpunkter: hva har gått bra, hva kunne vært bedre, og hvordan skal de punktene som kunne vært bedre utbedres i neste Sprint, se Vedlegg B.

## 2.4 Styringsverktøy: Azure DevOps

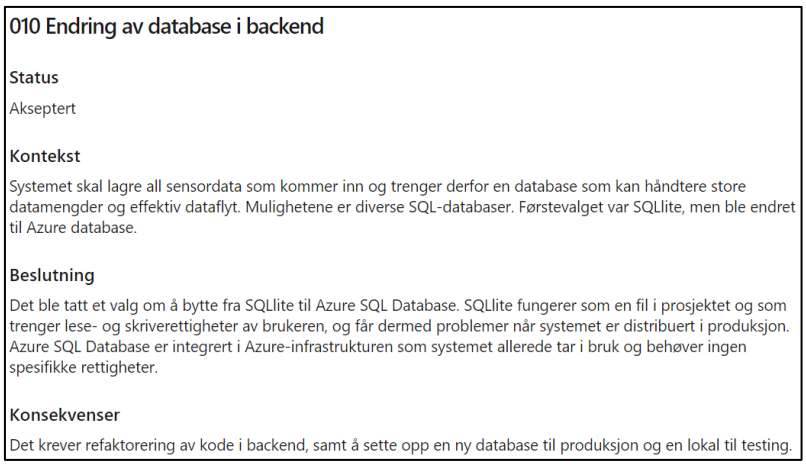
Azure DevOps er en programvare-som-tjeneste-plattform (SaaS) med en rekke innebygde verktøy for utvikling og distribusjon av programvare (Bigelow, 2023). Azure DevOps ble valgt som styringsverktøy hovedsakelig for å ha kontroll over elementene i Scrum. Videre hadde Azure DevOps fordelen av å fungere både som versjonskontrollsystem og dokumenteringsverktøy. I tillegg kunne applikasjonen settes i produksjon fra Azure, noe som gjorde det til en alt-i-ett-løsning. Azure DevOps ble et viktig verktøy for gruppen, og ble flittig brukt i forbindelse med de forskjellige artefaktene og hendelsene i Scrum-rammeverket. Azure DevOps ble brukt til å planlegge Product Backlog og Sprint Backlog, samt til å planlegge sprinter og å prioritere og organisere oppgaver. I tillegg ble verktøyet brukt til å markere oppgaver med tidsestimat, ansvarsperson, type og tilhørende brukerhistorie, se Figur 2.




*Figur 2. Board som inneholder organiserte oppgaver med område, estimat og ansvarsperson.*

For hver Sprint ble det også lagt inn et Sprint Goal i Azure DevOps, da plattformen har et eget sted hvor man kan sette opp dette. Dette målet var helt eller delvis basert på akseptansekriteriene i en eller flere av brukerhistoriene.

Azure sin Wiki ble også brukt til å føre en Architectural Decision Record, heretter kalt ADR. På denne måten var denne informasjonen oppbevart på samme sted som resten av prosjektet, for enklere tilgang og for å redusere antall plattformer man gjorde seg avhengig av. En ADR er en logg eller oversikt over valgene som har blitt truffet rundt oppbyggingen og løsningene man har valgt, og prosessene og argumentasjonen som har ledet til disse valgene. Det kan være nyttig å ha mulighet til å kunne se tilbake på disse beslutningene lenger fremme i prosessen, hvor man ikke alltid husker hvorfor noe ble gjort på en spesifikk måte (Yin, 2022). Figur 3 viser eksempel på én av flere oppføringer i ADR.



**010 Endring av database i backend**

**Status**

Akseptert

**Kontekst**

Systemet skal lagre all sensordata som kommer inn og trenger derfor en database som kan håndtere store datamengder og effektiv dataflyt. Mulighetene er diverse SQL-databaser. Førstevalget var SQLlite, men ble endret til Azure database.

**Beslutning**

Det ble tatt et valg om å bytte fra SQLlite til Azure SQL Database. SQLlite fungerer som en fil i prosjektet og som trenger lese- og skriverettigheter av brukeren, og får dermed problemer når systemet er distribuert i produksjon. Azure SQL Database er integrert i Azure-infrastrukturen som systemet allerede tar i bruk og behøver ingen spesifikke rettigheter.

**Konsekvenser**

Det krever refaktorering av kode i backend, samt å sette opp en ny database til produksjon og en lokal til testing.

*Figur 3. Én av flere oppføringer i ADR.*


*Figur 4. Skjermdump av pipeline-verktøy i Azure DevOps.*

I tillegg gav Azure DevOps enkel oversikt over blant annet Repoene i prosjektet, Branching, Pull Requests, samt Pipeline for å sette prosjektet ut i produksjon, se Figur 4.

## 2.5 Estimering

Hensikten med å estimere arbeid i et prosjekt er å ha kontroll over ressursbruk, og kunne si noe om hvor lang tid noe vil ta, og når det kan være ferdig (Amsjø, 2022).

Som nevnt i et tidligere kapittel har gruppen gjort estimater av forventet tidsbruk for de forskjellige oppgavene i Backlog. Gruppen har benyttet seg av et grovinndelt estimeringssystem basert på poeng, og estimert på Backlog Items. Det er tre hovedgrunner til dette:

- Vår samarbeidspartner, Norkart, er ikke et miljø som har et spesielt stort fokus på estimering.
- Flere kilder, blant andre gjesteforelesere i IS-304, har uttalt at estimering er vanskelig og tar tid, og i stor grad ikke har noen verdi for deres bedrift eller arbeidsprosess – det er bare i prosjekter hvor man har en helt fastlåst ramme og må rettferdiggjøre tids- og ressursbruk at de er nøye med estimering.
- Gruppen har begrenset erfaring innenfor mange av områdene i dette prosjektet, og det er vanskelig å komme med presise estimater.

Det ble gjort en kalkulert avveining om at man vil kunne miste noe kontroll, mot at man unngår å bruke uforholdsmessig mye tid på en aktivitet som tilsynelatende ville ha begrenset verdi for prosjektet og gruppen. I den sammenheng ble det valgt å gjøre estimering i en grovdelt poengskala, med poeng fra en til fem. Se Tabell 1 for flere detaljer.

| Poeng | Verdi |
|---|---|
| 1 | Enkel oppgave, tar ca. 1-2 timer |
| 2 | Middels oppgave, tar ca. en halv dag |
| 3 | Omfattende oppgave, tar ca. 1 dag |
| 5 | Vanskelig oppgave, tar mer enn 1 dag |

*Tabell 1. Poengsystem for estimering.*

Disse poengene kan også multipliseres, for eksempel om to utviklere ventes å jobbe sammen en hel dag om en trepoengsoppgave vil dette regnes som seks poeng, da det opptar hele kapasiteten for begge utviklerne den dagen. På denne måten mener gruppen at man har hatt tilfredsstillende kontroll over tidsbruk og omfang. Gruppen anser ikke en mer detaljert løsning som hensiktsmessig, da det ville være ekstremt vanskelig å gi mer presise estimater enn det som er nevnt over på grunn av manglende erfaring.

## 2.6 Risikovurdering

Risiko kan betegnes som forholdet mellom sannsynlighet for at en uønsket hendelse kan inntreffe, sammen med en konsekvens av hendelsen som følger (Datatilsynet, 2019). Selve risikoanalysen inneholder vurderinger av hvor trolig de ulike hendelsene og de påfølgende konsekvensene er. De tre stegene innen risikostyring er å identifisere, analysere og evaluere risiko. Disse stegene er en nødvendig del av prosjektet for å gi gruppen grunnlag for å ta beslutninger i henhold til risiko.

Gruppen startet først med risikoidentifisering. Formålet med dette er å forsøke og lage en liste over de tingene som kan gå galt i prosjektet (Digdir A, u.d.). Da gruppen hadde identifisert de ulike risikoene, så gruppen på antatt årsak til hendelsene, samt hvilke konsekvenser hendelsen ville få, og sannsynligheten for at det ville inntreffe. Figur 5 viser en typisk risikomatrise, som illustrerer forholdet mellom konsekvens og sannsynlighet, og representerer dette med et risikotall og en fargekode.


*Figur 5. Risikomatrise med sannsynlighetsfordeling mellom ulike konsekvensnivå. (Wood, 2019)*

Det siste og tredje steget er evaluering av risiko. Formålet med dette er å bestemme hvilke risikoer som aksepteres, hvilke risikoer som skal håndteres, og hvilke tiltak som eventuelt skal settes inn (Digdir A, u.d.). Feltene sannsynlighet og konsekvens multiplisert med hverandre. Fargekodene kan forklares som følger: grønn betyr lav risiko, gul er moderat, oransje er høy, og rød er kritisk. Både fargekode og tall gir indikasjon på alvorlighet.

| Hendelse | Sannsynlighet | Konsekvens | Risiko | Avbøtende tiltak |
|---|---|---|---|---|
| Sosial loffing | 1 | 4 | 4 | Tydelige krav til arbeidsinnsats<br>Fleksitid for å lette reiser o.l.<br>Behandle oppgaven som en jobb<br>Sitte på felles kontorplass |

*Figur 6. Definert risiko på Sosial loffing, med rangert sannsynlighet, konsekvens og handlingsplan.*

| Hendelse | Sannsynlighet | Konsekvens | Risiko | Avbøtende tiltak |
|---|---|---|---|---|
| Støter på problemer som er for vanskelige | 5 | 4 | 20 | Selvstudium/kompetanseheving<br>Lese dokumentasjon og tutorials<br>Samarbeide med andre på gruppen<br>Lav terskel for å spørre om hjelp fra veiledere |

*Figur 7. Definert risiko når det kommer til å støte på problemer som er for vanskelig.*

Ett eksempel på risiko er hendelsen sosial loffing, se Figur 6. Det er bestemt at laveste sannsynlighet og konsekvens skal markeres med tallet én, og det høyeste med tallet fem. I dette eksempelet ligger sannsynligheten på én og konsekvens på fire. Produktet av de to feltene blir da en risiko på fire, altså en moderat risiko. Et annet eksempel er hendelsen med å støte på vanskelige problemer, se Figur 7. Her er det høy sannsynlighet på fem og relativt høy konsekvens på fire, sammen blir dette en kritisk

risiko på 20. Dette forteller oss at det er nesten garantert at det vil oppstå vanskelige problemer i prosjektet, og man burde planlegge hvordan man skal håndtere problemene. Gjennom prosjektet ble denne risikoen en realitet og mange oppgaver endte med å være vanskelige. Da disse problemene oppstod ble det satt i gang tiltak ved å samarbeide mer om oppgavene og få råd fra Norkart om hjelp eller eventuelle løsninger på problemene. Risikovurderingen kan sees i sin helhet i Vedlegg C.

Etter gruppens mening var det viktig å gjennomføre en risikovurdering for å ha en bevissthet rundt risikoene i prosjektet, og kunne legge planer for hvordan de høyeste risikoene kunne reduseres i forkant, eller etter hvert som hendelser inntraff. Dette gir gruppen bedre forutsigbarhet i arbeidet.

# 3. Prosjektgjennomføring

Dette kapittelet vil redegjøre for stegene som har blitt tatt i forbindelse med selve gjennomføringen av prosjektet. Først vil kartlegging av systemkrav og brukerbehov bli avklart, videre hvordan denne innsikten har ført til et visuelt design og til slutt selve implementeringen av systemet.

## 3.1 Kartlegging av systemkrav

Dette kapittelet tar for seg hvordan gruppen har kommet frem til en systemdefinisjon og hvordan ulike systemkriterier har blitt prioritert.

### 3.1.1 Systemdefinisjon

En systemdefinisjon er en kortfattet beskrivelse av et system, og skal klargjøre grunnleggende elementer og egenskaper ved systemet som utvikles (Mathiassen et al., 2018, s. 24). For å komme frem til en systemdefinisjon anbefaler Mathiassen å bruke "FACTOR Criteria" som hjelpemiddel (2018, s. 40). FACTOR-kriteriene består av seks faktorer som kan brukes til å støtte utviklingen av en systemdefinisjon. Dette kan oppnås ved å vurdere de seks faktorene under utviklingen av definisjonen, eller ved å beskrive systemet og deretter identifisere hvordan faktorene passer inn i systemdefinisjonen (Mathiassen et al., 2018, s. 40). Tabell 2 viser gruppens utfylte FACTOR-kriterier.

| | |
|---|---|
| **Functionality** | Overvåkning av sensordata |
| **Application Domain** | Vise sensordata i en oversiktlig form |
| **Conditions** | Vise geografisk lokasjon og data i et enkelt og oversiktlig grensesnitt |
| **Technology** | Azure, .NET, API, React, Leaflet, Typescript, Webklient |
| **Objects** | Normatic, Norkart, ansatte i kundebedrifter, sensorer |
| **Responsibility** | Visualisere sensordata |

*Tabell 2. FACTOR criteria.*

Den endelige systemdefinisjonen for prosjektet ble som følger:

> *«Et informasjonssystem for å hente og representere sensordata fra vann og avløp på en nettside. Nettsiden inneholder et kart som automatisk oppdaterer en grafisk fremstilling av sensordataene med geografisk lokasjon og måleverdier. Kartet skal gi et oversiktsbilde over status til de forskjellige sensorene, og bidra til økt situasjonsforståelse.»*

Systemdefinisjonen har hjulpet oss med å trekke ut essensen av produktet og har gjort det klarere hvilken funksjonalitet som er mest nødvendig å ha. Dette har vært til hjelp ved utvikling og prioritering av brukerhistorier, da det har gitt et grunnlag for å prioritere vekk det som ikke er ansett som like essensielt som andre deler. Etter at systemdefinisjonen hadde hjulpet med å skape et generelt overblikk av systemet, ble neste steg å gå dypere inn i detaljene og sette ulike systemkriterier opp mot hverandre.

### 3.1.2 Prioritering av systemkriterier

Systemkriterier er en rekke retningslinjer, eller en guide, for å definere hvor viktig hvert krav i systemet er (Mathiassen et al., 2018, s. 180). Prioriteringen av de ulike kriteriene vil kunne være til hjelp når man skal designe en systemarkitektur som passer til konteksten av produktet man lager. Tabell 3 under viser hvilke prioriteringer som har blitt gjort for vårt system, og under tabellen følger en utfyllende forklaring som går dypere inn i hvorfor kriteriene har blitt plassert som tabellen viser.

| Criterion | Very Important | Important | Less Important | Irrelevant |
|---|---|---|---|---|
| Usable | | x | | |
| Secure | | | | x |
| Efficient | | | x | |
| Correct | x | | | |
| Reliable | x | | | |
| Maintainable | x | | | |
| Testable | x | | | |
| Flexible | x | | | |
| Comprehensible | x | | | |
| Reusable | x | | | |
| Portable | | x | | |
| Interoperable | | x | | |

*Tabell 3. Systemkriterier.*

***Usable*** – Konseptet/produktet trenger ikke å tilpasses ut fra hvem eller hvor det skal brukes, da utformingen er relativt universal.

***Secure*** – Årsaken til at dette har blitt satt som irrelevant er todelt. For det første har Norkart ferdige sikkerhetsløsninger som kan benyttes, noe som gjør at gruppen ikke trenger å utvikle sin egen sikkerhetsløsning for denne applikasjonen. For det andre er dette en POC som kun håndterer syntetiske data uten noen som helst viktighet eller konfidensialitetsbehov, og sikkerhet er dermed ikke det viktigste behovet i denne sammenheng.

***Efficient*** – Dette punktet handler om effektiv utnyttelse av den tekniske plattformens potensiale. Dette er et prosjekt i ganske liten skala, og vi har fått tildelt en teknisk plattform som skal brukes uansett. Vi forsøker etter beste evne å utnytte de mulighetene som kommer med den tekniske plattformen der det er relevant for sluttproduktet, men det er ikke en hovedprioritet.

***Correct*** – I og med at dette er et POC-prosjekt med en klart definert hensikt, er det viktig at produktet oppfyller de relativt få kravene som stilles til punkt og prikke. Derfor vurderer vi dette til «very important».

***Reliable*** – Hvis sensordataen skal gi reell verdi til dem som skal bruke systemet er det viktig at det ikke er avvik i kvaliteten på dataene. For at det skal bli et pålitelig produkt bør det være en gjennomgående pålitelig kvalitet i hele prosjektet.

***Maintainable*** – Siden det er en POC-applikasjon så vil den trolig ikke kjøre i årevis og behøve mye vedlikehold, så for applikasjonens egen del trenger den ikke å være enkel å vedlikeholde. På den annen side ønsker gruppen å bruke utviklingen av applikasjonen som en øvelse i å skrive kode som er lett å vedlikeholde, så i den sammenheng er det likevel en svært viktig del av prosjektet.

***Testable*** – Ved hjelp av enhets- og integrasjonstester kommer vi til å sikre at systemet alltid gir rett resultat for den aktuelle funksjonen.

***Flexible*** – Systemet vil bli bygget med løs kobling som hovedregel, hvor hver enkelt lille del skal kunne endres uten at dette vil påvirke de andre delene av prosjektet. Hvis noe må endres, skal resten av systemet fortsatt fungere.

***Comprehensible*** – For å kunne bruke det vi har lagd inn i et annet system, eller å videreutvikle vårt konsept, så må det være enkelt både å forstå og kunne bruke de forskjellige delene av systemet. Abstraksjon i form av klassifisering, generalisering og modularisering er måter man kan gjøre systemet forståelig og oversiktlig på, og det er noe som er høyt prioritert.

***Reusable*** – Konsekvent bruk av de overnevnte måtene å abstrahere deler av systemet på, vil kunne bidra til at de ulike delene blir mer forståelig og i tillegg at de kan brukes på nytt i andre kontekster.

***Portable*** – Vi lager et system som skal være løst koblet og enkelt skal kunne brukes i ulike situasjoner. Derfor er det viktig at det ikke er et isolert system, men at det brukes teknologi som støtter opp om prinsippet om portabilitet.

***Interoperable*** – Her gjelder mye av det samme som for punktet over. Systemet vil i prinsippet være enkelt å integrere i andre applikasjoner, så det vil automatisk bli relativt enkelt å koble til andre systemer.

Prioriteringen av systemkriterier har hjulpet med å konkretisere de ulike delene som er viktigst for utviklingen av dette systemet. Det har vært til stor hjelp for å vite hvilke programmeringsoppgaver som skal prioriteres, og hvordan man kan begynne planleggingen av stegene som må til for å implementere disse. Systemkriteriene har også blitt justert underveis ettersom den øvrige kartleggingen har fått fram nye og viktige aspekter.

## 3.2 Kartlegging av brukerbehov

I dette kapittelet vil vi redegjøre for hvilke steg vi har tatt og hvilke verktøy som har blitt tatt i bruk for å lage en løsning tilpasset aktuelle brukere av systemet.

### 3.2.1 Brukere

Brukerne av KDV er hovedsakelig kommuner, i tillegg til enkelte private aktører som drifter tjenester med lignende behov som kommunene. Mer spesifikt er det typisk en driftsavdeling innenfor de enkelte kommunene eller bedriftene som er de faktiske brukerne av tjenestene. Dette prosjektet er en gren av KOMTEK relatert til vann- og avløpssektoren, hvor systemet kan tenkes å bli tatt i bruk av de overnevnte brukerne.

Det har ikke vært noen reelle brukere å kunne kontakte i forbindelse med brukertesting, på grunn av avgjørelser tatt fra arbeidsgivers side. Denne avgjørelsen ble tatt i bakgrunn av at vann og avløp er vurdert som kritisk infrastruktur og inneholder dermed en mengde sikkerhetsrisikoer (Justis- og beredskapsdepartementet, 2021). Løsningen til dette var at prosjektet skulle ta i bruk syntetiske testdata og ikke reell data fra en av deres kunder.

Representantene fra Norkart har derfor valgt å ta på seg rollen som brukere for vårt system, og har kommet med innspill og tilbakemelding underveis gjennom hele prosjektet. Selv om representantene fra Norkart ikke er faktiske brukere, har de fortsatt innsikt i brukerbehov fra kartlegging de har gjort i forbindelse med de andre tjenestene i KDV.
Et annet aspekt ved mangelen på brukere er at dette er et undersøkende prosjekt som har mer til hensikt å utforske teknologi og konsepter, og at brukerne dermed kan bli involvert ved et senere tidspunkt om Norkart ønsker å gå videre med løsningen.

### 3.2.2 Personas

Personas fremstiller fiktive karakterer som representerer de ulike brukertypene ved et produkt eller system (Additive, u.d.). Det ble utviklet Personas for å gi gruppen innblikk i brukerens behov, erfaringer, atferder og mål, se Figur 8. Personaene er laget med et innebygget verktøy i Azure DevOps og er basert på samtalene med Norkart.


*Figur 8. Gruppens fem personas.*

Gruppen kan dermed lettere identifisere seg med brukeren og deretter utvikle brukerhistorier som skal dekke både behov og forventninger for systemet.

### 3.2.3 Brukerhistorier

Brukerhistorier gir et grunnlag for prioritering av oppgaver opp mot akseptansekriteriene (Entur, u.d.). En brukerhistorie er fra brukerens perspektiv og beskriver diverse funksjonalitet som må være til stede for å oppfylle systemkravene. Formatet til alle brukerhistoriene er fast og sier noe om *hvem* som ønsker noe, hvilken *funksjon* det gjelder og hvilken *verdi* funksjonen tilfører. De aktuelle funksjonene har blitt bestemt etter samtaler med arbeidsgiver og den øvrige kartleggingen av systemet. Figur 9 viser en av de utviklede Persona-ene med tilhørende brukerhistorie. Se Vedlegg D for resten av Personas og Vedlegg E for alle brukerhistoriene.

Tilsynsmannen Pål må foreta kontroll av V/A, samt se på risikoen for forurensning av miljøet og spredning av sykdom ved å sikre godt drikkevann og god vannkvalitet.

Han trenger faktiske sanntidsdata som kan fortelle han noe om vannkvaliteten, men også dersom forurensning skulle oppstå. Som tilsynnsmann, er han ansvarlig at kontrollene blir foretatt; slik at innbyggerne er sikret god vannkvalitet og miljø.

| | | | | |
|---|---|---|---|---|
| **Must have** | 3# Som bruker ønsker jeg å se temperaturmåling (i sanntid) fra hver sensor slik at jeg har oversikt over temperaturen til hver pumpestasjon. | Temperaturmåling er blir vist i et popup-vindu i form av tall/tekst. | Gitt at jeg bruker web-appen, skal jeg ha:<br>• Mulighet til å trykke inn på hver sensor og lese av tilhørende sanntidsmålinger (popup-vindu). | Vannets temperatur sier noe om dens kvalitet og forhold. Oversikt over vannets temperatur er derfor et must have. |

*Figur 9. Persona til venstre viser tydelig sammenheng med brukerhistorien til høyre.*

Prioriteringen av brukerhistoriene har blitt gjort i form av MoSCoW. De brukerhistoriene som blir definert som Must Have er et absolutt krav å ha med i systemet, mens Should Have klassifiseres som funksjoner som ikke er helt essensielle, men likevel vil tilføre betydelig verdi. Ved å bruke MoSCoW-metoden har det dermed blitt enklere å kunne definere hva som skal være systemets Minimum Viable Product (MVP), da det har blitt tatt utgangspunkt i Must Have fra MoSCoW-prioriteringen. MVP inneholder de viktigste funksjonene for at systemet skal kunne tas i bruk (Babych, 2021). Innsiktene fra kartleggingen blir brukt videre i utformingen av det grafiske brukergrensesnittet.

## 3.3 Design av grafisk brukergrensesnitt

Etter å ha undersøkt systemkrav og brukerbehov, brukte gruppen denne innsikten til å fremstille utseendet på det grafiske brukergrensesnittet (GUI). Dette kapittelet beskriver designprosessen, fra inspirasjonsfasen til skissering, samt videreutvikling med Navigation Map, Wireframes og Mockup. Brukskvaliteten til brukergrensesnittet blir ivaretatt ved å anvende grafiske designprinsipper.

### 3.3.1 Designinspirasjon

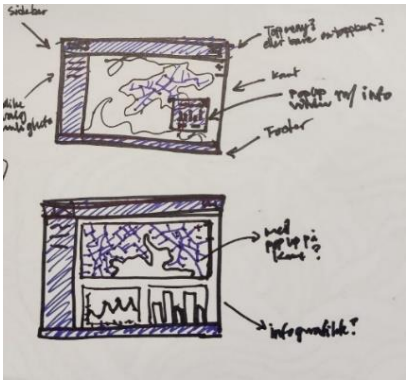
Basert på systemdefinisjonen og brukerbehovene har gruppen konkludert med at systemet vil være best egnet som en nettbasert kartløsning. Kartet skal gi en oversikt over sensordata og være enkel å ta i bruk. DOGA påpeker at et designprosjekt starter med en utforskningsfase for å samle inspirasjon og innsikt (u.d.), så gruppen begynte designprosessen ved å søke etter inspirasjon.
Pinterest, Google Maps og andre lignende tjenester av samme karakter ble brukt for å skape en oversikt over designtrender. Her ble det sett på alt fra form, fargekombinasjoner, funksjoner, komposisjoner og balanse. Det ble også stilt interne spørsmål på hvordan systemet burde se ut:

- Hva slags funksjoner og elementer vil kunden ha?
- Hvordan kan disse visualiseres på en minimalistisk og oversiktlig måte?
- Hva slags inntrykk skal kunden få når de ser på systemet?
- Er det viktig å ha med alle elementene?
- Hva bør prioriteres i forbindelse med sprintene som skal utføres?
- Hvordan skal vi ta hensyn til designprinsipper?

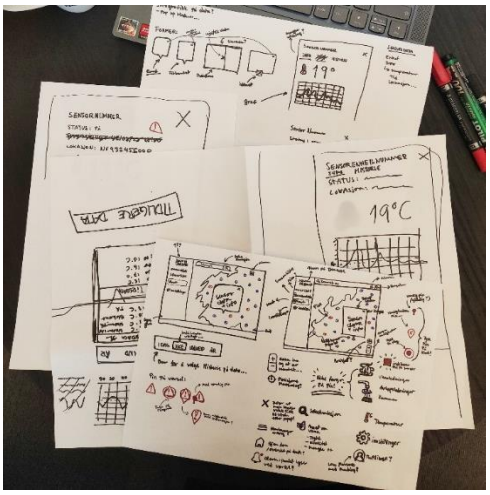
Spørsmålene over er med på å sette i gang en tankeprosess rundt hva gruppen skulle ta hensyn til. Dette la et godt grunnlag for det videre arbeidet med å tegne skisser.

### 3.3.2 Skissering

Gruppen valgte å fremstille idéene sine i form av skisser i første omgang, da det var flere fordeler ved denne tilnærmingen. Skisser har ikke mange detaljer, men nok til at det grunnleggende blir fremhevet (Mørstad, 2023). På denne måten var det enkelt for alle å delta og presentere sine idéer. Samtidig er skisser lette å endre og krever få ressurser. I skissene fokuserte gruppen først og fremst på å inkludere de høyest prioriterte funksjonene fra brukerhistoriene. Figur 10 viser noen idéer på hvordan systemet kunne se ut.



*Figur 10. Fra første skisse har vi to alternativer med ulik fremvisning.*

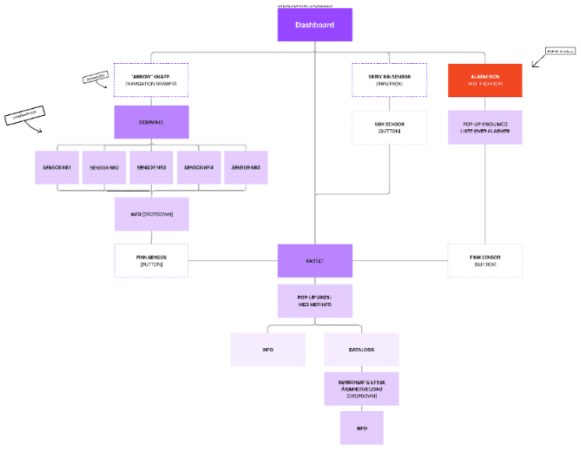


*Figur 11. Flere skisser med både ikoner, popup-alternativer og andre vesentlige funksjoner.*

Det ble tegnet flere skisser som kunne blitt tatt videre, men valget endte opp på de mer minimalistiske fremstillingene. Etter en skisse ble valgt ut diskuterte gruppen videre designvalg på forskjellige funksjoner som ikoner, farger, plassering, størrelse og kontrast, se Figur 11. Skissene var et eksempel på hvordan designet kunne se ut og la til rette for neste designfase, Navigation Map.

### 3.3.3 Navigation Map

Et Navigation Map, også kalt for navigasjonskart, er nyttig å ha med i designprosessen, da det representerer hvordan brukeren navigerer seg i systemet. I boken Conseptual Design for Interactive Systems, skriver professor Avi Parush at hensikten med å utvikle et Navigation Map er å skape oversikt over hvordan brukeren utfører nødvendige trinn for å oppnå et mål (Parush, 2015, s. 25). Det handler med andre ord om å legge frem ulike måter brukeren kan navigere seg gjennom systemet. Det hjelper også gruppen med å få en forståelse av navigasjonsmulighetene og hvordan de ulike elementene er knyttet sammen. Figur 12 viser en enkel presentasjon av hvordan brukeren navigerer seg gjennom systemet. Se Vedlegg F for et større format av Navigation Map.



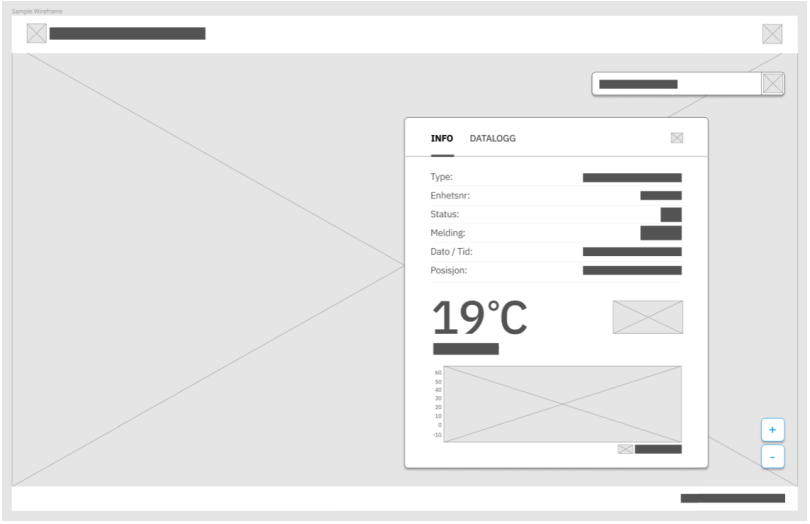

*Figur 12. Navigation Map.*

Videre i designprosessen fra skisse til Navigation Map, var neste steg å lage Wireframe av sidene som allerede er definert (Slickplan, 2022).

### 3.3.4 Wireframe

En Wireframe er et skjelett av hvordan designet skal se ut og kan være essensielt i designprosessen, da det tillater både utviklere og designere å gå gjennom produktet før det blir bygget (Website wireframe, 2023). En Wireframe kan brukes til å få flere perspektiver på funksjonalitet og designidéer. Her har gruppen lagd et oppsett til hvordan de ulike delene av nettsiden skulle se ut. Figur 13 viser et sidefelt med en oversikt over ulike sensorer, og Figur 14 viser utseende til sensorinformasjonen.


*Figur 13. Wireframe av systemet. Her ser vi Sidebar på venstre side.*

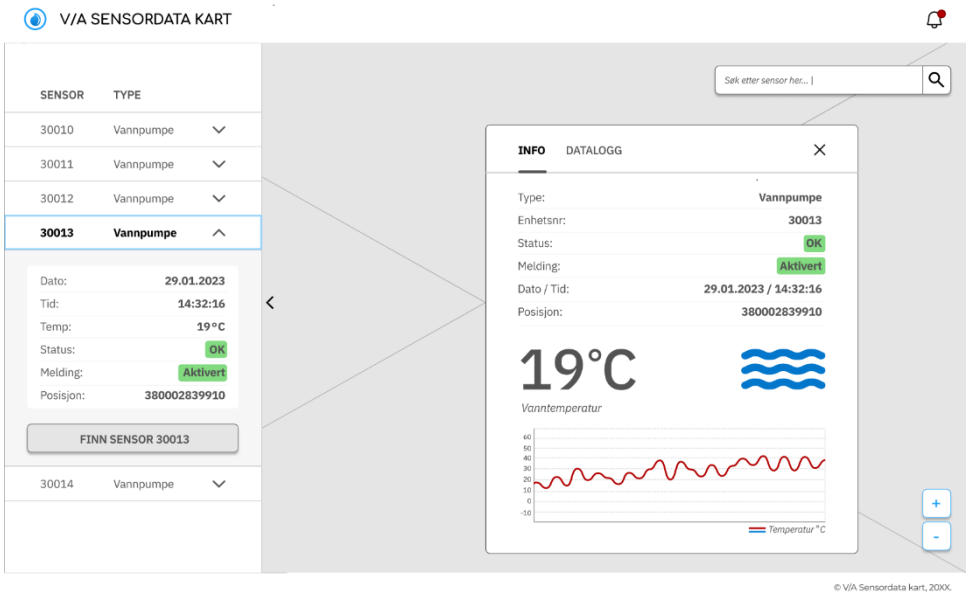



*Figur 14. Wireframe av systemet. Pop-up vindu over det som skal være kartet.*

Eksemplene inneholder designelementer som støtter de høyest prioriterte funksjonene for systemet som skal lages. De inkluderer dermed ikke en del av de mindre prioriterte funksjonene som kunne blitt implementert om prosjektomfanget var større. Neste steg i designfasen handler om Mockups som vil legge til rette for detaljer i designet.

### 3.3.5 Mockup

En Mockup er en høyoppløselig gjengivelse av produktets design som viser hvordan det ferdige produktet vil se ut (Sketch, 2022). Ved å utvikle Mockups, visualiseres designet med et mer realistisk brukergrensesnitt. De skaper et helhetsinntrykk og blir ofte brukt til å lage prototyper og for å få tilbakemeldinger fra brukeren. For å lage Mockups benyttet gruppens seg av verktøyet Figma og la inn nye detaljer som farge, bilder og typografi, se Figur 15.




*Figur 15. Mockup med designforslag. Sidepanel til venstre, og popup vindu til høyre.*

Brukskvaliteten av det grafiske brukergrensesnittet ble tatt hensyn til gjennom hele designprosessen, da brukervennlighet ble identifisert som et krav i brukerhistoriene. Gruppen har dermed anvendt grafiske designprinsipper for å opprettholde brukskvaliteten i designet vårt.

### 3.3.6 Grafiske designprinsipper

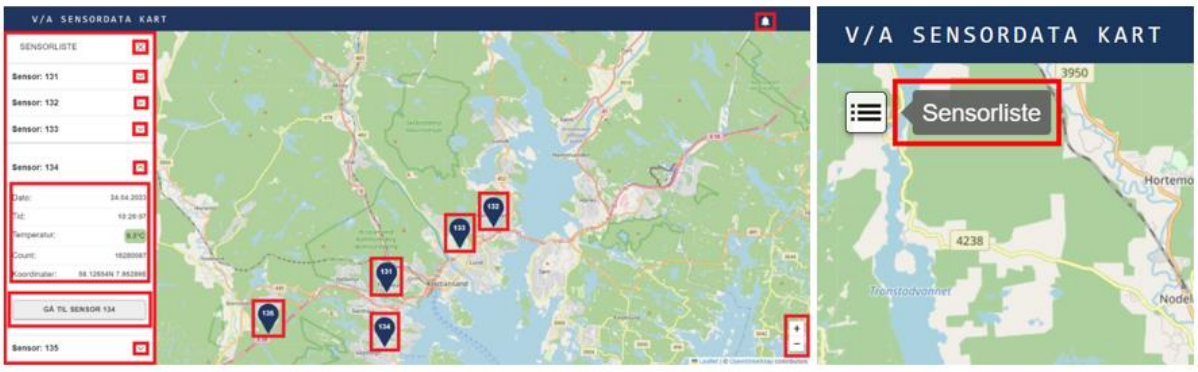
Designprinsipper er etablerte retningslinjer innenfor design som kan bidra til designprosessen ved å evaluere og kritisere designidéer (Benyon, 2013, s. 86). Det er viktig å dekke de fleste designprinsippene ettersom de er med på å skape et interaktivt design som er lett å lære og gir brukeren en form for kontroll. Ekspert innen Human–Computer Interaction (HCI), David Benyon, har utformet en guide til disse designprinsippene som er med på å evaluere designet som blir utviklet (2013, s. 86).

1. Visibility – sørg for at funksjoner og viktige elementer er synlige og tilgjengelige.
2. Consistency – konsekvent bruk av farger, skrifttype, designfunksjoner osv.
3. Familiarity – bruk språk og symboler kjent for publikum.
4. Affordance – sørg for at ting ser ut som det de gjør.
5. Navigation – sikre at folk enkelt kan navigere seg gjennom systemet.
6. Control – klargjør hva som kan kontrolleres og la folk ha kontroll.
7. Feedback – informer folk raskt om effekten av handlingene deres.
8. Recovery – sikre gjenoppretting fra feil og feil.
9. Constraints – restriksjoner for å begrense feil eller upassende handlinger.
10. Flexibility – flere måter å gjøre ting på.
11. Style – attraktiv eller stilig design.
12. Conviviality – høflig og vennlig system.

(Benyon, 2013, ss. 86-87)

De mest relevante designprinsippene for systemet gruppen utviklet har vært: Visibility, Consistency, Familiarity, Affordance, Navigation, Control, Feedback, Flexibility og Style.

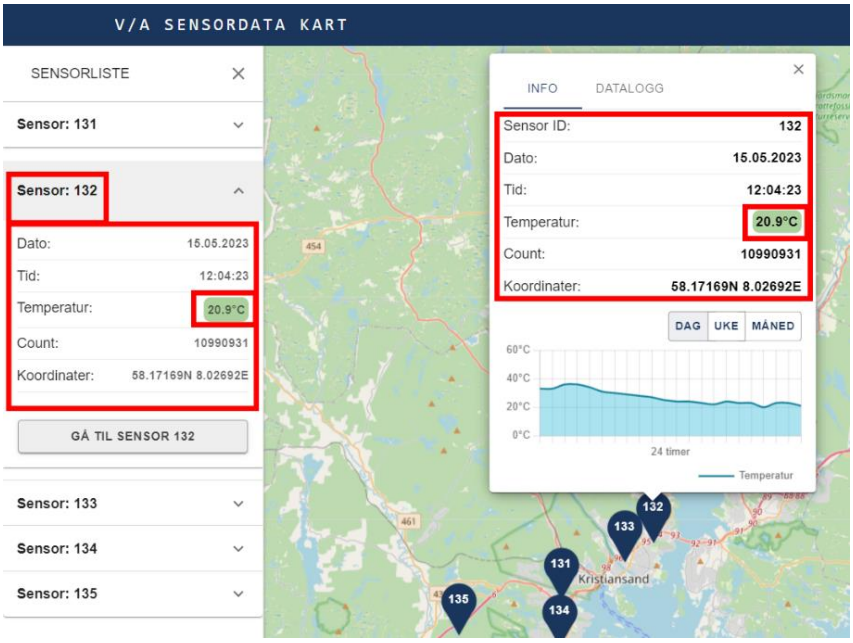
**Visibility -** Refererer til at brukeren kan for eksempel se til elementer som sidepanel, trekkspillkomponenten, knapper for å finne frem til sensor eller zoome inn og ut. Markørene med sensornummer indikerer at det finnes noe informasjon på kartet, se Figur 16. Ettersom disse er både synlige og tilgjengelig, går dette innenfor Visibility.




*Figur 16. Visibility går under synlige og tilgjengelige elementer som brukeren kan se.*

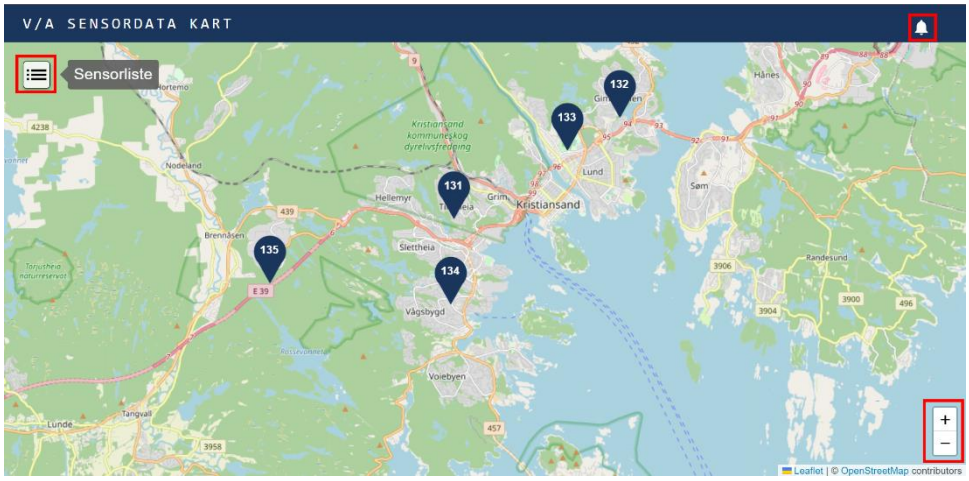
**Consistency** - For at designet skal fremstå tilfredsstillende å se på, var det viktig for gruppen å ha konsekvent bruk av farger, skrifttype og designfunksjoner. Dette har med at Consistency, som

refererer til at brukeren ser et konsekvent designmønster i designet, bidrar til at designet ikke blir forstyrrende og forvirrende (Nikolov, 2017), se Figur 17 for mer detaljer.



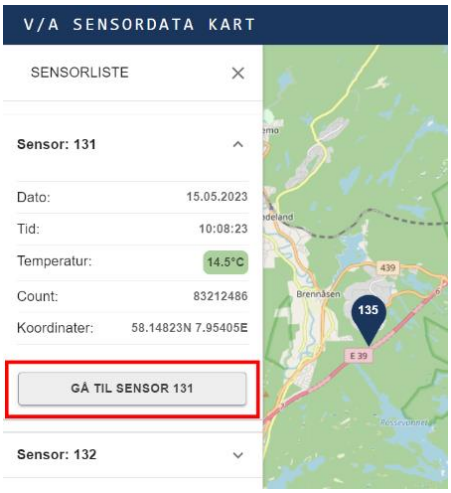

*Figur 17. Consistency med konsekvent designmønster.*

**Familiarity** - Refererer til hva brukeren er kjent for det fleste (Zehra, 2022). I forbindelse med systemet, kjenner brukeren igjen både zoom inn og ut knappene. Brukeren vet også at plussikonet indikerer at kartet kan forstørres, samme gjelder med minusikonet for å forminske. Designet på disse har en likhet med det som benyttes i andre kart. Brukeren kan også se at listeikonet indikerer at det er en type liste som er relatert til sensorene, mens bjellen indikerer varsel eller notifikasjoner, se Figur 18. Disse er også kjent for de fleste.



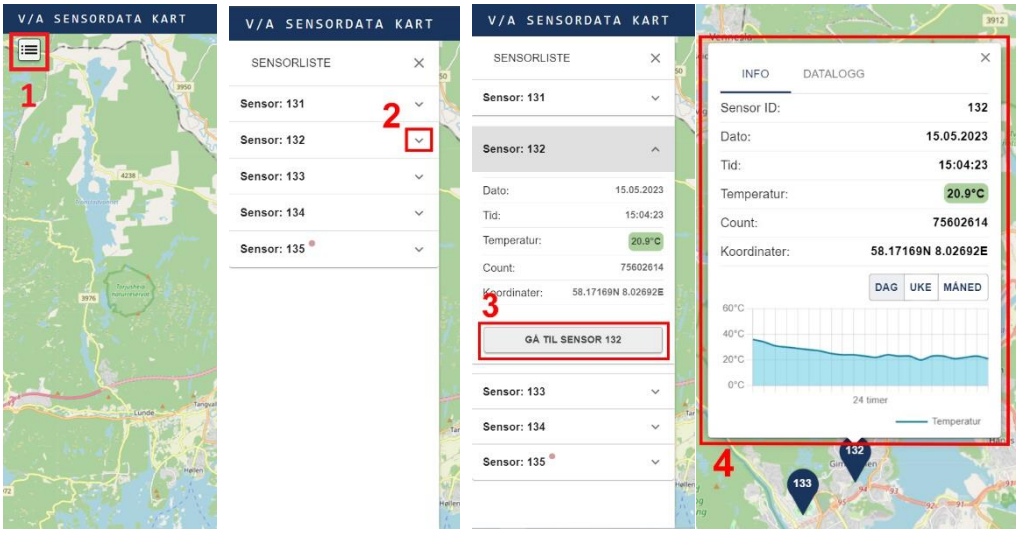

*Figur 18. Eksempel på Familiarity er elementene for å zoome inn og ut og diverse standardikoner.*

**Affordance** - Refererer til måten utformingen av et produkt eller grensesnitt kommuniserer til brukeren om hvilke valg som er mulige eller hensiktsmessige å ta (Uxcel, u.d.). I dette tilfellet har knappen et standard utseende og kommuniser til brukeren at den kan trykkes, se Figur 19.



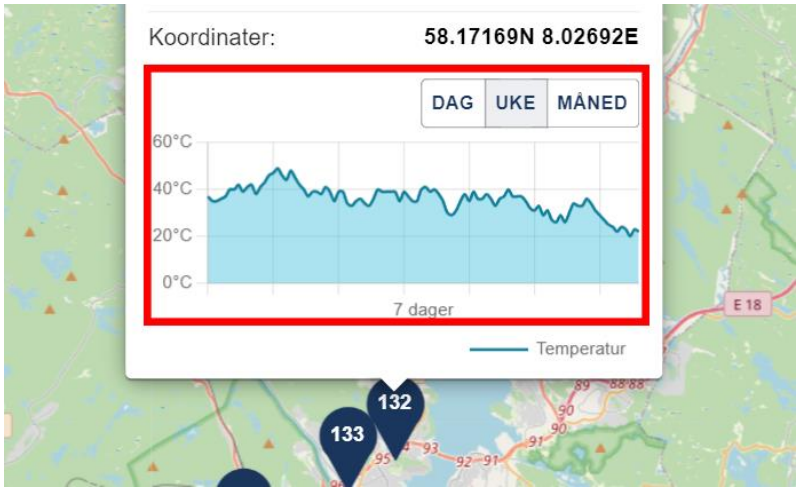

*Figur 19. Knapper er et eksempel på Affordance i systemet.*

**Navigation -** Refererer til hvordan brukeren navigerer seg rundt i systemet. Det skal være en klar respons på interaksjon som er tydelig og indikerer hvor brukeren kan navigere videre (Aaberge, 2021). Et eksempel på dette kan være når en bruker velger å trykke seg inn på sensorlisten, kan brukeren deretter utvide eller skjule informasjonen på listen. Ved å trykke på «gå til sensor»-knappen, vil brukeren videreføres til et popup-vindu på kartet. Deretter kan bruker trykke på krysset, og vinduet vil da forsvinne, se Figur 20. Dette forklarer en av flere måter som kan føre brukeren til informasjon på kartet.




*Figur 20. En av navigasjonsmulighetene brukeren har, fra sidepanel til popup-vinduet.*

**Control** og **Feedback** - Når brukeren trykker seg mellom en dag, en uke eller en måned på grafen, vil den historiske sensordataen endres deretter. Dette er med på å gi brukeren en form for kontroll, da brukeren har bestemt en type handling. Ettersom grafen endrer seg, vil brukeren se en form for tilbakemelding. Dermed går denne delen under feedback, se Figur 21.




*Figur 21. Et eksempel på en form for kontroll kan være som komponenten button group.*

**Flexibility** - Et annet designprinsipp som er essensielt for bruk av systemet, er muligheten til fleksibilitet. Designprinsippet Flexibility refererer til at brukeren får flere måter å gjøre ting på. I dette tilfellet kan brukeren finne frem til sensorene på flere måter. Dette kan brukeren gjøre ved å trykke direkte på markørene i kartet, som fører til et popup-vindu for både informasjon og graf, eller gjennom sensorlisten på sidepanelet. Ved å trykke på knappen i sidepanelet, vil samme popup-vindu vises frem. Det var diskusjon rundt å implementere et søkefelt for sensorene som tredje navigasjonsmulighet, men det ble bestemt at funksjonen var utenfor prosjektets omfang. Etter samtale med brukeren kom gruppen frem til at dersom systemet skaleres opp kan en slik funksjon være essensiell for å sikre at brukeren når sitt mål på nettsiden. Slike valg ble tatt for å unngå Feature Creep, hvor et prosjekt eller et produkt sine krav øker utenfor det som allerede er definert i planleggingsfase (Batterbee, 2020).

**Style -** Henger sterkt sammen med Consistency, og handler om det grafiske utseende. Ettersom systemet tar utgangspunkt i vann og avløp var fargen blå valgt som hovedfargen til systemet, da mennesker generelt assosierer blå med vann og himmel (Cherry, 2022). Fargen blå er i tillegg med på å gi en følelse av ro og fred, symboliserer stabilitet og er best likt blant kvinner og menn (Cain, 2017). Svart og hvit er nøytrale farger som skaper god kontrast og gir dybde (Fatrabbit CREATIVE, u.d.), mens gråtoner som regel passer med de fleste farger (Farley, 2010). Ettersom temperatur både kan stige og synke, ble det bestemt at rødt og grønt skulle skille mellom normal temperatur og høy temperatur. Rødt og grønt er innført som signalfarger, siden rødt vanligvis indikerer fare og symboliserer varme (Holtsmark, 2020). De ble av den grunn valgt for å varsle om godkjent eller for høy temperatur.

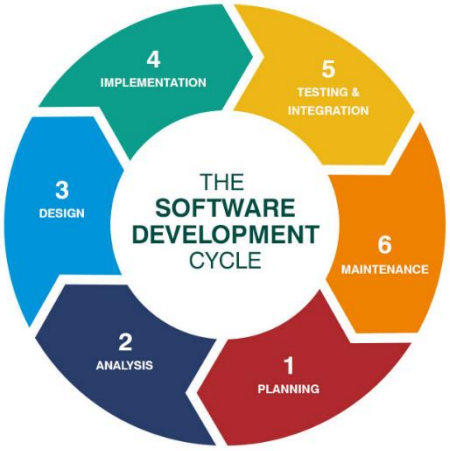
Designprinsippene er brukt som et grunnlag for å utvikle et interaktivt design og gir oss et visuelt utgangspunkt for utvikling av applikasjonen.

## 3.4 Implementering av systemet

I dette kapittelet vil vi presentere vår tilnærming til implementeringen av systemet. Vi vil først beskrive vår metodikk knyttet til selve utviklingen og deretter presentere systemarkitekturen og implementeringen av de ulike programvarekomponentene som utgjør systemet.

### 3.4.1 Systemutviklingslivssyklus

Systemutviklingslivssyklus er ifølge Amazon Web Services en metodikk som brukes i forbindelse med utvikling av programvare (Amazon Web Services, Inc., u.d.). Målet med metodikken er å planlegge utviklingen slik at systemet treffer brukernes behov best mulig, og modellen består av noen forhåndsdefinerte steg som skal hjelpe med dette/gjøre dette mulig. Som vist på Figur 22 går livssyklusen fra det sjette og siste steget og tilbake til første steg igjen. Dette er med på å illustrere en agil måte å forholde seg til metodikken på, hvor man kontinuerlig leverer mindre leveringer og går gjennom stegene på nytt ved for eksempel hver Sprint. Gruppen har til en viss grad fulgt en slik tilnærming til systemutviklingen.




*Figur 22. Systemutviklingslivssyklus. (Big Water Consulting, 2019)*

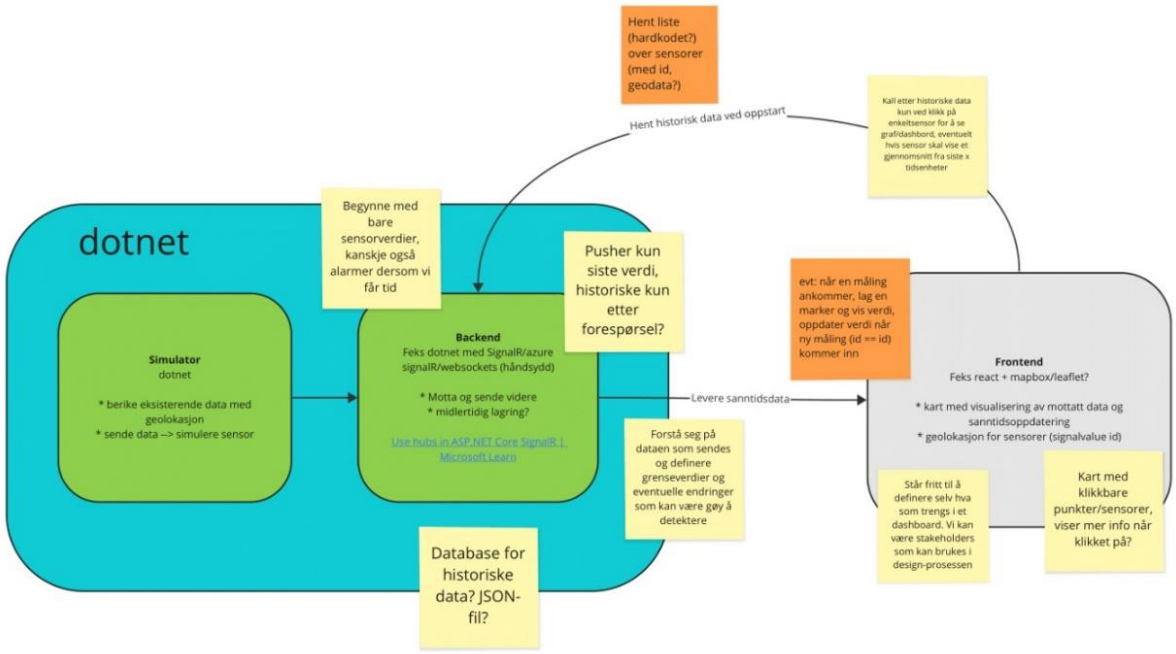
Ved hver ny Sprint har det blitt definert et Product Increment for den aktuelle sprinten, som tar utgangspunkt i en eller flere brukerhistorier. Brukerhistoriene har så blitt analysert og designet for å gjøre delene om til funksjoner og elementer som må til for å oppfylle brukerhistorien, før disse har blitt kodet inn i systemet. Enhetstester har blitt skrevet samtidig som ny kode har blitt laget, så disse stegene overlapper hverandre noe, men det har også blitt utført manuell testing av funksjonalitet i etterkant av implementeringen før det ferdige resultatet/produktet har blitt deployert i Azure. Det siste steget som handler om vedlikehold av den eksisterende programvaren, inkluderer blant annet fiksing av feil og overvåking av systemet for å finne ting som kan gjøres bedre (Amazon Web Services, Inc., u.d.). Ved neste Sprint kan man begynne å planlegge potensielle endringer og forbedringer som ble oppdaget i den foregående sprinten, og på den måten vil systemet kontinuerlig tilpasses for å lage et så godt system som mulig.

### 3.4.2 Systemarkitektur

Systemarkitektur er et begrep som refererer til en konseptuell modell som definerer strukturen og oppførselen til et system (Systems Architecture, 2023). Teknologivalgene som har blitt tatt i forbindelse med implementeringen av systemet er basert både på anbefalinger fra arbeidsgiver,

diskusjon innad i gruppen og undersøkelser knyttet til hva som best kan passe de kriteriene og kravene som er satt for systemet. Det har vært et bevisst valg fra gruppens side å benytte seg av teknologiene som Norkart selv har kjennskap til og bruker til daglig, slik at vi enkelt har hatt mulighet til å spørre om hjelp og få råd.

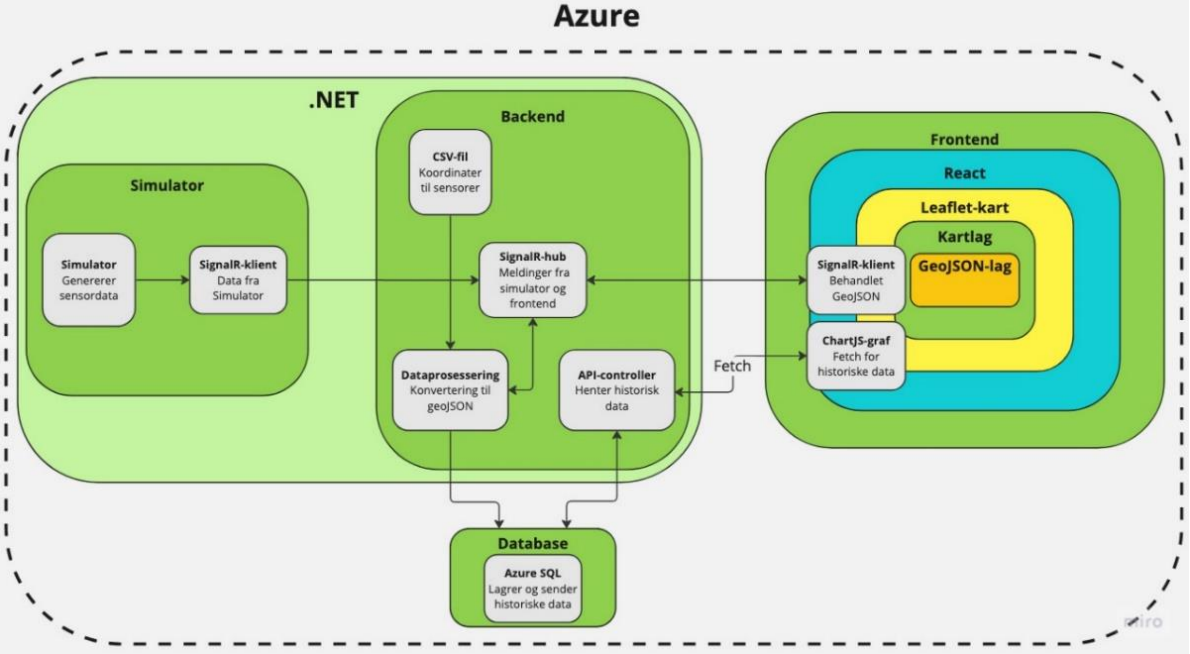
Systemet skulle i utgangspunktet basere seg på sensordata i et test-API fra Normatic. Dette API-et består av sensorverdier splittet i to kategorier; temperaturmålinger og alarmer. Av det originale prosjektomfanget var begge kategoriene valgt ut, men i sluttproduktet er det kun sensorverdiene fra temperaturmålingene som er inkludert. Senere ble det valgt å gå vekk fra Normatic sitt API, og heller lage vårt eget API. Dette valget blir utdypet i kapittel 3.4.3. Det ble holdt et arkitekturmøte sammen med arbeidsgiver under planleggingen av prosjektet. Under møtet ble problemstillingen, teknologier og fremgangsmåter som kunne brukes i prosjektet presentert. Arkitekturen har vært gjenstand for regelmessige endringer spesielt gjennom startfasen av prosjektet ettersom produktet har utviklet seg og nye problemstillinger har dukket opp. Det har blant annet blitt inkludert en database som i den første versjonen av arkitekturskissen ikke var der, se Figur 23.




*Figur 23. Første versjon av arkitekturskisse.*

Databasen ble inkludert i arkitekturen da alle hovedfunksjonene var ferdigstilt og det ikke var flere brukerhistorier knyttet til sanntidsdata, se Figur 24. Den neste prioriterte brukerhistorien handlet om fremvisning av historisk data, og da ble det bestemt å implementere en database for lagring av denne dataen. Av andre endringer kan det nevnes at Azure har blitt lagt til som en omkransende del av systemarkitekturen da prosjektet i sin helhet er deployert via Azure. I tillegg var simulatoren opprinnelig en komponent som kun skulle ta imot data fra Normatic sitt test-API og berike disse dataene med koordinater og eventuelt flere sensorverdier. Da det ble bestemt at Normatic sitt API

ikke skulle brukes i systemet ble simulatoren refaktorert til å generere egne sensordata. Håndteringen av tillegging av koordinater til hver sensor ble også flyttet fra simulator til backend-komponent.




*Figur 24. Endelig systemarkitektur.*

### 3.4.3 Simulator

På det opprinnelige arkitekturmøtet ble det bestemt at man skulle lage en simulator. Hensikten var todelt; hente data fra Normatic sitt API og berike disse med geodata, samt eventuelt produsere ekstra data på samme format, og sende disse videre til backend-applikasjonen. Målet var at backend-applikasjonen ikke skulle måtte utvikles til å håndtere flere typer inputs, og at simulatoren på sikt kunne erstattes med en reell datastrøm fra faktiske sensorer.

Etter hvert som utviklingen kom i gang ble det klart for gruppen at API-et til Normatic ikke var stabilt nok, inneholdt nok data, eller hadde hyppige nok oppdateringer for gruppens formål. Planen ble derfor endret, og kall til Normatic sitt API ble tatt ut av bruk, og applikasjonen baserte seg utelukkende på egne syntetiske data produsert i simulatoren. På denne måten hadde gruppen nå full kontroll over dataflyten i eget system, noe veilederne hos Norkart også anbefalte.
Siden dataene fra hver sensor heller ikke inneholdt noe informasjon om geolokasjon, ble geodataene bestemt flyttet til backend-delen. På denne måten ville man kunne motta data fra en sensor, og slå opp lokasjon ved hjelp av ID. Figur 25 viser hvordan sensordataen ser ut når den kommer fra simulatoren og hvordan den ser ut etter at geolokasjon har blitt lagt til.


*Figur 25. Sensordata fra simulator før og etter bearbeiding.*

Simulatoren består av en sensormodell, metoder for å generere forskjellige typer data for sensoren, en tidtaker, og en SignalR-klient som sender dataene til backend-applikasjonen i form av en JSON-streng. Når simulatoren startes oppretter den et valgfritt antall sensorer og genererer data for hver sensor, som for eksempel id, oppdateringsfrekvens, temperatur, og så videre. På samme tidspunkt starter hver sensor også en tidtaker for seg selv, basert på oppdateringsfrekvensen til hver enkelt sensor. Når denne tidtakeren går ut, vil sensoren oppdatere sin verdi. Når simulatoren startes, og for hver gang en sensor oppdateres, sendes disse dataene via SignalR til backend-applikasjonen. På denne måten kan man simulere en rekke sensorer som eksisterer uavhengig av hverandre, og som kun har ansvar for å sende inn egne data til SignalR-huben i backend-applikasjonen.

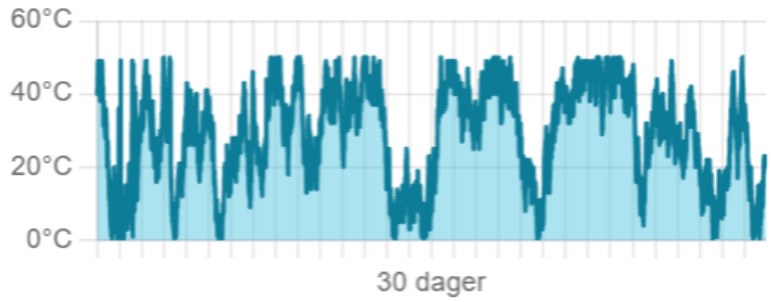
### 3.4.4 Backend

I systemutvikling er det mange lag som skiller bruker og maskinvare, backend er laget som tar for seg forretningslogikk og datatilgang (Frontend and backend, 2023). I klient-server-modellen er backend en del av serveren klientene kobler seg på. I et system inneholder backend ulike teknologier og er som regel kun tilgjengelig for serveren.

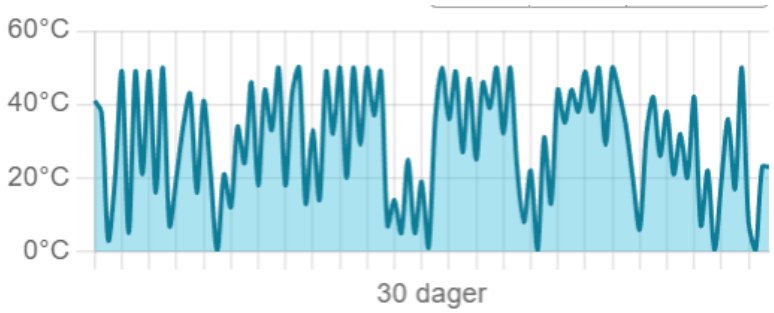
En av de mest essensielle teknologiene som prosjektet baserer seg på er SignalR. SignalR er en del av ASP.NET-rammeverket og brukes til å lage applikasjoner med sanntidsfunksjonalitet. Noe av funksjonaliteten er automatisk håndtering av tilkoblede enheter, utveksling av meldinger til noen spesifikke eller alle koblede enheter, og skalerbarhet for å håndtere økt trafikk. Dette tillater kode på serversiden å sende innhold direkte til klienten i sanntid (Microsoft, 2023). Ett eksempel kan være at SignalR er ofte anvendt i applikasjoner som inkluderer chatteromfunksjonalitet. Ved å koble til chatterommet kan brukere i sanntid utveksle meldinger med andre deltakere uten behov for å oppdatere siden. Ifølge Microsoft er det flere kriterier som burde følges dersom applikasjonen skal være en god kandidat for å ta i bruk SignalR (2023). Kriteriene er applikasjoner med høy oppdateringsfrekvens, dashbord-, monitorering- og samarbeidsapplikasjoner, eller applikasjoner som trenger notifikasjonsvarsler. Prosjektet har som formål å overvåke hyppig oppdatert sensordata som skal vises på et kart. Dette oppfyller flere av de anbefalte kriteriene fra Microsoft for å bruke SignalR-teknologien.

Gjennom arkitekturdiskusjonene ble gruppen rådet av Norkart om å gjøre de ulike delene av applikasjonen så løst koblet som mulig. På grunn av dette ble backend-delen av prosjektet til en selvstendig komponent, som andre systemer kan koble seg til. Dette tillater at programmer som eksempelvis Power BI og QGIS også kan koble til for å hente de samme dataene, på samme måte som frontend i dette prosjektet. For å gjøre dette mulig har det i størst mulig grad blitt tatt i bruk standardiserte formater som GeoJSON og andre etablerte praksiser for hvordan dataen blir hentet fra backend.

Backend består hovedsakelig av to deler. Den ene delen tar imot data, bearbeider den for å legge til geolokasjon og sender det videre via en SignalR-hub. Dette skjer i sanntid. Den andre delen består av et eget API som henter historiske data fra databasen og kjører det gjennom en algoritme. Denne algoritmen er kalt «Largest Triangle Three Buckets» (LTTB) og reduserer antall datapunkter, men greier likevel å beholde mye av den visuelle konturen fra det originale datasettet. Datapunktene vises i appen på en graf, og grunnet den begrensede størrelsen på denne grafen måtte antall datapunkter reduseres for å gjøre det mulig for brukeren å se utviklingen av temperaturverdiene. Hvis man ser for seg at oppdateringsfrekvensen en sensor kan ha er på én gang i minuttet, vil det i løpet av 30 dager være over 40 000 datapunkter som skal visualiseres for brukeren. Reduseringen av dette antallet til færre datapunkter gjør at mengden data som brukeren må laste inn blir mindre og vil føre til både en raskere applikasjon og økt lesbarhet (Steinarsson, 2014).




*Figur 26. Graf uten redusert antall datapunkter*




*Figur 27. Graf med redusert antall datapunkter*

Som vist på Figur 27 blir grafen tydeligere i forhold til Figur 26, mens hovedlinjene fortsatt vises. Algoritmen plukker ut kun den mest sprikende sensorverdien over et forhåndsdefinert område på grafen. Her kan et problem oppstå hvis dette området inneholder mange sprikende verdier som er nyttige å observere, da vil noe av denne dataen ikke vises.

### 3.4.5 Frontend

Brukergrensesnittet for systemet er en webapplikasjon som har blitt utviklet med React. React er et JavaScript-bibliotek som er et av de mest populære rammeverkene for å lage webapplikasjoner, og blir også brukt av Norkart. Gruppen har også valgt å bruke programmeringsspråket TypeScript istedenfor JavaScript, da det både har blitt anbefalt fra arbeidsgiver og andre IT-bedrifter gruppen har vært i kontakt med. En av forskjellene er for eksempel at TypeScript har statiske typedefinisjoner, som gjør at variablene blir definert ved kompileringen av programmet, og ikke under eksekveringen (TypeScript, n.d.). Det vil gjøre det enklere å oppdage syntaktiske feil i koden på et tidligere tidspunkt enn hvis man skulle brukt JavaScript.

Applikasjonen inneholder flere forskjellige biblioteker som har blitt brukt i varierende grad for å bygge opp de forskjellige elementene på nettsiden. Et bibliotek kan beskrives som en del ferdiglagde metoder og kodesnutter som kan tas i bruk istedenfor å lage alt fra bunnen av selv (McFarland, 2012, s. 118).

Kartet er lagd med et React-spesifikt bibliotek som heter React-Leaflet, et påbygg av JavaScript-biblioteket LeafletJS. Avgjørelsen om å bruke dette biblioteket var en prosess som ble diskutert med Norkart fordi biblioteket bruker en lisens som kan være problematisk. React-Leaflet har gått fra å benytte seg av en såkalt MIT-lisens til en «Hippocratic Ethical Licence» som legger noe mer føringer på bruken av produktet, og Norkart har tatt et standpunkt om å ikke bruke produkter med en slik lisens. Til tross for det ble det bestemt at det var greit å bruke biblioteket i dette prosjektet, ettersom produktet er en POC som ikke skal ut til brukere.

For styling og layout har det blant annet blitt brukt Material UI, som er et bibliotek med mye innebygd funksjonalitet og designelementer tilpasset React. Material UI-komponentene har blitt brukt som et grunnlag for flere av elementene i appen, hvor det har blitt gjort forandringer på utseende og funksjonalitet slik at det skal passe med appens kriterier.

Et annet viktig element er grafen som viser den historiske utviklingen av sensorverdiene. Denne er hentet fra React-ChartJS-2, som i likhet med React-leaflet er en React-utvidelse av det noe mer kjente ChartJS. Dette biblioteket lar brukeren se hvordan temperaturen har beveget seg i noen utvalgte tidsintervaller.

For å hente ut sensorverdiene fra backend har det blitt tatt i bruk to forskjellige metoder. Sanntidsdataen for sensorene kobler seg til backend med en SignalR-klient som lytter på ny data

som kommer inn. Den historiske dataen blir hentet fra backend via et API-kall som skiller mellom de ulike sensorene og hvilke tidsrom man skal hente data fra.

# 4. Kvalitet og kvalitetssikring

I dette kapittelet vil gruppen diskutere hvordan man har forsøkt å sikre kvalitet i både prosjektarbeid og produkt, hvordan kvalitet defineres og oppnås.

## 4.1 Definere kvalitet

Utdanningsforbundet hevder at det mangler en felles definisjon for begrepet "kvalitet", både generelt og innenfor utdanningsområdet (Skulberg & Aaslid, 2019). Derfor må begrepet knyttes til noe konkret for å kunne gi det mening. Dersom det dreier seg om en tjeneste eller et produkt, kan man definere "kvalitet" som evnen til å oppfylle brukerens krav og forventninger (Gundersen & Halbo, 2018). I dette prosjektet mener gruppen at kvalitet oppnås når produktet møter kravene og forventningene fra både oppdragsgiveren og de aktuelle brukerne. Med dette i tankene har gruppen valgt å legge vekt på tre områder for å måle oppfyllelsen av disse kravene:

1. Produktet leverer det som forventes
2. Kunde/bruker er fornøyd med det som leveres
3. Koden i seg selv holder så objektivt høy kvalitet som mulig.

### 4.1.1 Produktet tilfredsstiller forventninger

Det er både viktig, og helt grunnleggende, at applikasjonen utfører de oppgavene som forventes. For å sikre at produktet leverer det som forventes har gruppen valgt å benytte seg av to verktøy: systemdefinisjon og brukerhistorier. Systemdefinisjonen har gitt gruppen en generell beskrivelse av hva produktet er tenkt å være, mens brukerhistoriene har gått mer i detalj og spesifisert ønsket funksjonalitet.

Ved å bruke brukerhistoriene, og spesifikt akseptansekriteriene til disse, under utforming av oppgavene i Product Backlog, har gruppen klart å utforme spesifikke kriterier for når hver brukerhistorie er implementert. På denne måten sikrer gruppen at applikasjonen dekker de områdene som er bestemt i samarbeid med produkteier, og at det ikke finnes hull i funksjonaliteten som forventes. Ved å prioritere ved hjelp av MoSCoW-metoden passer også gruppen på at det er den viktigste funksjonaliteten som implementeres først.

### 4.1.2 Fornøyde brukere

At brukeren er fornøyd med opplevelsen av å bruke applikasjonen kan i seg selv være et tegn på kvalitet, selv om det eventuelt skulle være andre mangler ved applikasjonen.

### 4.1.3 Kvalitet i kode

Gruppen har ønsket å skrive så god kode som mulig – kode som er effektiv, lett å lese, lett å vedlikeholde, lett å utvide med ny funksjonalitet, og lett for andre å få nytteverdi av. Gruppen anser det som utenfor scopet av denne oppgaven å kvantifisere disse tingene, men bevissthet rundt dette har likevel vært viktig for gruppen gjennom hele utviklingen.

## 4.2 Kvalitetssikring

I tillegg til å definere hva som bestemmer om kvalitet er oppnådd, er det viktig å også vite hvordan man kommer seg dit. Gruppen har hatt flere aktiviteter, prosesser og metoder for å oppnå den ønskede kvaliteten.

### 4.2.1 Produktet tilfredsstiller forventninger

Ved å være bevisst på brukerhistoriene og akseptansekriteriene under sprintplanleggingen, har gruppen til enhver tid klart å fylle Sprint Backlog med oppgaver som er relevante for å få implementert de viktigste funksjonalitetene i applikasjonen. På denne måten har man sikret en jevn fremgang mot målet om å implementere først brukerhistoriene som var Must Have, og deretter Should Have og Could Have.

Resultatet er at systemet dekker beskrivelsen fra systemdefinisjonen, og alle Must Have og Should Have, i tillegg til noen Could Have fra brukerhistoriene har blitt suksessfullt implementert i applikasjonen. Gruppen mener derfor å ha oppnådd ønsket kvalitet på dette området.

### 4.2.2 Fornøyde brukere

Siden denne applikasjonen er en POC, og ikke har noen faktisk bruker, har det ikke blitt gjennomført intervjuer, brukertesting, eller andre aktiviteter opp mot en sluttbruker. I stedet har veilederne i Norkart tatt på seg rollen som både produkteier og bruker, og det er krav eller ønsker fra disse gruppen har forholdt seg til gjennom hele prosessen. Blant annet er Personas og brukerhistorier utformet basert på tilbakemeldinger fra veilederne.

Gruppen har gjennom hele prosjektet søkt tilbakemeldinger fra veileder, blant annet gjennom Sprint Reviews eller andre presentasjoner underveis. Alt i alt har tilbakemeldingene fra veilederne i Norkart vært gode, og gruppen er tilfreds med at ønsket kvalitet er oppnådd også på dette området.

### 4.2.3 Kvalitet i kode

Som nevnt i kapittel 4.1.3 er dette et punkt som gruppen ikke har prioritert å kvantifisere i dette prosjektet, og det er derfor utfordrende å vurdere om ønsket kvalitet i realiteten er oppnådd. Gruppen har likevel hatt fokus på prosesser og arbeidsmetoder som skal være med på å bidra til at denne kvaliteten skal bli så god som mulig:

**Felles kodestandard**
En kodestandard er et felles sett med regler eller retningslinjer for hvordan kode skal skrives. Hensikten med å ha en felles standard for hvordan man skriver kode er å gjøre den homogen uavhengig av hvem den er skrevet av, lett å lese, lett å vedlikeholde, og skalerbar (University of St. Andrews, u.d.).

Gruppen diskuterte allerede tidlig i prosjektet viktigheten av å ha en felles forståelse for hvordan kode skulle skrives. For enkelte prosjekter kan det utarbeides et dokument som skal guide all skriving av kode. Gruppen anså det ikke som nødvendig å gå til et slikt skritt for dette prosjektet; alle studentene har relativt lik bakgrunn og har samme opplæring i koding fra dette studiet, og har dermed cirka samme grunnlag å jobbe ut fra.

I stedet ble det bestemt noen sentrale prinsipper for fremgangsmåte;

- Bruke etablerte praksiser- gruppen ønsker så langt som mulig å bruke etablerte praksiser på forskjellige områder. Ved å basere arbeidet mest mulig på etablerte praksiser vil man bygge sin egen kode på allerede utprøvde metoder og løsninger, og dermed kunne unngå feil og fallgruver som gruppen ellers kanskje ikke har nok erfaring til å unngå.
- Flittig bruk av veiledere i partnerbedrift – gruppen ønsket å gjøre mest mulig nytte av tilgang til erfarne programmerere i Norkart som kunne gi hjelp og komme med tips.
- Code Reviews – gruppen gjennomførte Code Reviews, både med veiledere i Norkart og internt i gruppen. Ved å få flere øyne på koden er det lettere å fange opp feil eller se bedre måter å gjøre ting på.
- Pull Requests – ved å godkjenne hverandre sine Pull Requests vil man få flere øyne på koden, og sikre at den er forståelig og korrekt.

**Kodetesting**
Testing av applikasjoner er viktig for å sikre at funksjoner virker som planlagt, at data er korrekte, og at kravene til applikasjonen er oppfylt (Microsoft, 2022). Testing er et fagfelt i seg selv, og det finnes ekstremt mange metodikker som kan brukes for å teste programvare – i dette prosjektet har gruppen valgt å anvende både enhetstesting, integrasjonstesting og manuell testing.

Enhetstesting er en type automatisert test som tester funksjonaliteten til en spesifikk bit kode, vanligvis på metode-nivå (Microsoft, 2022). Enhetstester er (når de først har blitt utviklet) veldig tidsbesparende, siden de kan kjøres raskt hver gang man gjør en endring i koden, og slipper å teste manuelt at metoden(e) man har endret har beholdt sin tiltenkte funksjonalitet, og resultatet fra kjøring av metoden(e) blir som forventet. Gruppen har i dette prosjektet utviklet en del enhetstester. I simulatoren er testdekningen begrenset til metodene som genererer data, for å sikre at disse er innenfor ønsket område. Det ble vurdert å være tilstrekkelig dekning, da denne delen av applikasjonen kun skal simulere data som kommer inn fra sensorer plassert i VA-anlegg. I backend-applikasjonen er testregimet en del strengere, da denne delen er mer sentral i utøvelse av applikasjonen sin funksjon.

Det er utviklet integrasjonstester, som er et steg over enhetstester, og har som formål å teste hvordan flere komponenter samsvarer med hverandre, ofte vil de enhetene som har blitt testet slått

sammen til én integrasjonstest (Integration testing, 2023). Denne integrasjonstesten er i backend og kan koble seg på systemet lokalt eller i Azure for å teste hele backend-koden. Testen vil begynne med et data-input som kjører gjennom datakonverteringen og sendes til databasen. Testen vil få et output som er i riktig konvertert dataformat. På denne måten vil hele backend sin dataflyt bli testet.

De delene av prosjektet som det er mindre kurant å teste med enhets- og integrasjonstester, som for eksempel kommunikasjon mellom de forskjellige applikasjonene i Azure, har blitt testet manuelt.

Som nevnt er det vanskelig å si noe objektivt om kodekvalitet uten noen metode for å kvantifisere hva kvalitet i kode er, men subjektivt er gruppen selv relativt fornøyd med den kvaliteten som er oppnådd.

### 4.2.4 Andre kvalitetssikringsaktiviteter

I tillegg til de konkrete områdene nevnt over, er det en del andre ting i forbindelse med prosjektarbeidet som påvirker hvilken kvalitet som oppnås.

**Prosjekttrekanten**

Bevissthet rundt prosjekttrekanten som nevnes i 2.1 har vært viktig for gruppen for å forstå hvordan kvalitet skal kunne oppnås. Gruppen var klar over at blant de tre faktorene i trekanten som påvirker kvalitet, så var det *scope* som i størst grad kunne påvirkes av gruppen selv, og til en viss grad *ressurser*. Det at gruppen var klar over dette mener gruppen selv har bidratt til at man har valgt seg ut et passende scope, og klart å levere et produkt innenfor dette scopet som holder en høy kvalitet. Dersom man ikke hadde gjort seg noen tanker rundt prosjekttrekanten, og da risikerte å se seg ut et for stort scope, kunne dette resultert i en mer omfattende applikasjon, som på mange områder holdt en lavere kvalitet enn det som her har blitt oppnådd.

**Scrum**

Som nevnt i kapittel 2 ble Scrum valgt som gruppens rammeverk for dette prosjektarbeidet. Scrum har blant annet aktiviteter som Sprint Retrospective og Sprint Review, som fokuserer på kontinuerlig forbedring av både prosessene og produktet. Sprint Review har blant annet latt gruppen presentere arbeidet sitt for både Norkart og veileder fra universitetet, hvor man har fått verdifulle tilbakemeldinger som har bidratt til å øke kvaliteten i produktet. Sprint Retrospective har vært viktig for gruppen, da denne aktiviteten har latt gruppen ta opp ting som kunne vært gjort bedre i hver Sprint. Dette har ført til en økende kvalitet i gruppens arbeid fra Sprint til Sprint, etter hvert som problemer har blitt oppdaget og adressert.

**Designprinsipper**

Siden dette prosjektet ikke har en faktisk bruker, har gruppen sett på det som ekstra viktig å følge etablerte designprinsipper for å opprettholde kvaliteten i brukergrensesnittdesignet. Ved å følge etablerte og utprøvde prinsipper øker gruppen sannsynligheten for at applikasjonen får et design som er lett å bruke og enkelt å forstå.

**Architectural Decision Record**
Underveis i prosjektet har gruppen vært nødt til å treffe en del viktige valg angående arkitektur. Veilederne i Norkart anbefalte gruppen å opprette en Architectural Decision Record for å holde styr på begrunnelsen bak de viktigste beslutningene rundt systemets arkitektur.

Gruppen ønsket å lage en ADR både for å selv kunne se tilbake på valgene som ble gjort dersom dette skulle bli nødvendig, men også for å gjøre applikasjonen enklere å forstå for utenforstående som får tilgang til den, på samme måte som kommentarer i kode gjør den enklere å lese og forstå. I dette prosjektet har ikke informasjonen samlet i ADR blitt spesielt mye brukt, da det ikke har oppstått behov for å gå tilbake og friske opp hvorfor spesifikke valg ble tatt, men i et større prosjekt med lenger varighet kan gruppen definitivt se nytteverdien av å føre en slik logg. Dessuten mener gruppen at en ADR øker kvaliteten til den totale løsningen, siden den gjør det lettere for en tredjepart å sette seg inn i valgene som er gjort underveis.

# 5 Refleksjon
Nå, ved prosjektets avslutning, ønsker gruppen å reflektere litt rundt hva som har blitt produsert, om målene for prosjektet er oppnådd, hva som har gått bra, og hva man kunne gjort annerledes.

**Måloppnåelse**
I kapittel én ble det drøftet forskjellige mål for prosjektet. Det første var Norkart sine mål, som var å la gruppen gjøre forsknings- og utviklingsarbeid, og lage en POC som potensielt kunne gi en verdi for bedriften i fremtiden. Hvor stor den faktiske verdien av prosjektet blir for Norkart er vanskelig å si på nåværende tidspunkt, men prosjektet har i alle tilfeller utforsket nyttige verktøy som eksempelvis SignalR, ChartJS og Leaflet, som våre veiledere visstnok har begrenset erfaring med. I disse tilfellene kan gruppens bruk av disse verktøyene tenkes å få en verdi i form av kunnskapsoverføring fra gruppen til Norkart. I systemdefinisjonen står det også at applikasjonen skal bidra til å øke situasjonsforståelse rundt sensorverdier. Gruppen har utforsket måter man vil kunne få sensordata inn i kart, som for eksempel i Norkarts KDV-applikasjon. Med bakgrunn i dette vil gruppen hevde at applikasjonen som er utviklet i dette prosjektet vil kunne bidra til å øke situasjonsforståelsen til brukerne av KDV og Normatic sine sensorer, og at det overordnede målet med selve applikasjonen er oppnådd.

Når det gjelder gruppens egne mål for prosjektet var disse som nevnt tredelt – vi ønsket at prosjektet skulle gi en verdi til Norkart, at vi selv skulle få mengdetrening og utvikle våre ferdigheter innen systemutvikling og prosjektstyring, i tillegg til de akademiske målene. Disse målene føler gruppen at vi har oppnådd. Alle gruppemedlemmene har fått brynt seg på utfordrende oppgaver, utforsket nye teknologier, brukt relevante verktøy, og tilegnet seg ny kunnskap og nye ferdigheter. I tillegg har gruppen, som nevnt over, utforsket biblioteker og teknologier som kan få en nytteverdi for Norkart.

**Hva har gått bra?**
En av tingene som var viktig for gruppen i valg av bachelorprosjekt var at oppgaven passet for alle gruppemedlemmene, altså at hvert gruppemedlem fant arbeidsoppgaver som var innenfor eget interesseområde. Gruppen føler at dette har vært vellykket. Det har vært varierte arbeidsoppgaver, fra analyse og design til rene utviklingsoppgaver. Det at gruppen var bevisst på dette har gjort at hvert gruppemedlem har fått utnyttet sine sterkeste sider, og har fått utvikle videre de ferdighetene som er relevante for den type jobb hver enkelt ønsker seg etter endt studium.

Gruppen er også fornøyde med løsningen som har blitt utviklet. Kartlegging i starten av prosjektet og utvikling av brukerhistorier har gjort at gruppen har kunnet ha kontroll på hvilken funksjonalitet som var høyest prioritert for løsningen. Dette har gjort at man har kunnet begynne med det viktigste først, og deretter tilføre mer funksjonalitet etter hvert. Det ble gjort et valg om å ha et slankt MVP, slik at man skulle unngå Feature Creep og å bite over for mye.

I tillegg er gruppen fornøyd med balansen mellom ønsket om å ha kontroll over prosjektets omfang og kvalitet, og ønsket om å ikke planlegge for langt frem slik at man hadde nærmet seg en fossefallsmodell. For å få til dette har gruppen benyttet Scrum-rammeverket, men gjort egne tilpasninger som passet gruppens behov. Vi har også kontinuerlig evaluert arbeidsprosessene og vurdert endringer. Gruppen føler vi har funnet et fornuftig kompromiss mellom absolutt kontroll og fornuftig tidsbruk på planleggingsaktiviteter.

Gruppen mener selv at man har oppnådd god kvalitet i produktet, både basert på innfrielse av akseptansekriteriene i brukerhistoriene, og på tilbakemeldinger fra Norkart. Kvaliteten har blitt oppnådd ved hjelp av blant annet bruk av Scrum, at man valgte et overkommelig scope, definerte kriterier for kvalitet, og samtaler med «brukeren».

**Utfordringer**
Som i alle prosjekter har man støtt på utfordringer og problemer også i dette. Måten gruppen har avdekket og dermed kunnet forsøke å løse disse problemene på er ved å holde et Sprint Retrospective-møte i slutten av hver Sprint, hvor tre punkter har vært adressert: hva som har gått bra, hva som kunne vært gjort bedre, og eventuelle tiltak for at neste Sprint skal bli mer vellykket enn den foregående.

Noen eksempler på utfordringer som ble avdekket på disse møtene, og hvordan de ble forsøkt løst, er;

- Gruppemedlemmer satt for lenge med samme problem uten å gjøre fremgang – dette ble løst ved at terskelen for å spørre om hjelp ble senket, og det ble et større fokus på å benytte seg av parprogrammering der det var hensiktsmessig.
- Gruppemedlemmene satt i for stor grad i hver sin silo og arbeidet, uten å vite hva de andre holdt på med, til tross for at man hadde standup hver dag – denne utfordringen ble løst ved

at man satte av tid til å ha små presentasjoner og Code Reviews for hverandre, slik at man kom seg ut av sin egen lille boble.

- Estimering – estimering er et vanskelig emne. På den ene siden ønsker man full kontroll over prosjektet, hva som gjenstår, og hvor lang tid det vil ta. På den andre siden er det vanskelig å komme med presise estimater for oppgaver man har lite eller ingen erfaring med, og man ønsker ikke å bruke disproporsjonalt mye tid og krefter på planlegging som kanskje ikke gir et reelt bilde av hva som kommer til å skje når arbeidet settes i gang. Fra start hadde gruppen valgt å legge seg på et grovinndelt estimeringssystem, noe som har fungert greit. Etter hvert som arbeidet skred frem og man fikk en oversikt over cirka hva den reelle kapasiteten til gruppen var i «effort» per Sprint, kunne man også justere inn hvor mange oppgaver man tok inn i Sprint Backlog, for å ha en passe mengde arbeid i hver Sprint. På denne måten har man kunne hatt et tilstrekkelig godt oversiktsbilde over omfanget av arbeidet som gjenstår.
- API – API-et fra Normatic som gruppen skulle hente data fra viste seg å ikke helt dekke gruppens behov. Det var begrenset antall spørringer over en viss tidsperiode, noe som gjorde det vanskelig å jobbe med under utvikling og testing. I tillegg var det få sensorer og sjeldne oppdateringer. Gruppen valgte i stedet å lage sin egen simulator som produserte syntetiske data. Dette gav en mye bedre kontroll over dataflyten i systemet, som kan være fordelaktig i en utviklingsprosess.

**Begrensninger**

En begrensning gruppen har støtt på er at det ikke har eksistert noen faktisk brukergruppe i dette prosjektet, og veilederne i Norkart har derfor tatt på seg denne rollen. De har vært en sparringspartner for gruppen når det gjelder spesifikasjon og utforming av produktet. Denne løsningen har fungert greit, og har gitt gruppen relativt frie tøyler til å prioritere etter hva man selv synes virker viktig, eller som vil være faglig interessant å utvikle. På den annen side er det viktig å være oppmerksom på at det kanskje gir en noe urealistisk prosess sammenlignet med et prosjekt hvor man må forholde seg til en faktisk bruker, og å være bevisst på at prosesser som brukertesting og intervjuer av bruker kan være en viktig del av et IS-prosjekt som gruppen har gått glipp av i løpet av dette prosjektet.

**Hva har vi lært?**

De største lærdommene vi tar med oss fra prosjektet er følgende:

- Nye teknologier og biblioteker – det har vært veldig nyttig for gruppen å få utforske verktøy som Azure, som er utbredt i IT-selskaper, og å utnytte biblioteker som f.eks. SignalR.
- Tilpassing av Scrum-rammeverket for å møte egne behov – det har vært en god erfaring å gå inn i et prosjekt og være bevisst på at det er mulig å tilpasse Scrum for å passe gruppens spesifikke prosjekt og forutsetninger. Dette setter prosjektet og arbeidet i fokus, i stedet for rammeverket. Scrum har også bidratt til å fremme vårt mål om å anvende kritisk tenkning

gjennom prosjektet. Dette har blitt oppnådd blant annet gjennom gjennomføring av aktiviteter som retrospektiv, som har tvunget oss til å reflektere over prosessene våre.

- Balanse mellom oversikt og arbeidsfordeling/spesialisering – gruppen hadde et mål om at hvert medlem skulle få arbeide med oppgaver som man selv syntes var interessante, og som kunne være relevante å få mer erfaring innenfor, blant annet med tanke på fremtidig jobb. Dette førte til at man ble sittende litt for mye å jobbe i egne siloer i starten, og gruppen lærte at det er viktig å gjøre avbøtende tiltak for å rette på dette.
- Ser alt man har lært på studiet i sammenheng – et omfattende prosjekt som dette gjør at gruppen må bruke alt vi har lært på studiet frem til nå, og sy alt sammen til en helhet, noe som har vært svært interessant.
- Verdien av å ha en kontorplass med god tilgang på hjelp og lav terskel for å spørre, og faste arbeidstider og -rutiner – det at gruppen har fått fast plass på Norkarts kontorer, og dermed et sted å samles for å jobbe med prosjektet og enkel tilgang til hjelp fra mer erfarne utviklere, har vært uvurderlig. Det var en av de viktigste grunnene til at gruppen valgte denne oppgaven, og det har vist verdien av å ha folk rundt seg og lett tilgjengelig. Denne faktoren er noe som gruppen kommer til å ta med seg videre inn i arbeidslivet.

**Hva ville vi gjort annerledes?**

Gruppen ønsker også å reflektere litt rundt hva man ville gjort annerledes om prosjektet skulle gjøres på nytt. Det viktigste punktet her er å ha et større fokus på samarbeid og erfaringsutveksling fra start. Hadde det vært fokus på dette fra dag en, kunne man potensielt unngå situasjonen som oppsto hvor man til tider kunne havne i sin egen silo og var mindre oppmerksom på andre sitt arbeid. Dette ble imidlertid imøtegått gjennom Daily Scrum-møter og Code Reviews, der vi presenterte vårt arbeid for hverandre. Generelt hadde gruppen god kommunikasjon og samarbeid. Likevel ble dette identifisert som et område for forbedring under våre Sprint retrospektiver, og det kunne ha vært en fordel å ha hatt fokus på det fra starten av. Det er også nyttig for alle gruppemedlemmene å ha bredere kunnskaper og ferdigheter enn hvert sitt fokusområde, og allsidighet kunne vært en større prioritet fra begynnelsen av.

Det andre punktet vi vil trekke frem er at man kunne hatt et enda større fokus på brukerhistoriene i starten av prosjektet. Brukerhistoriene og prioriteringen av disse gir en slags fasit på hvilken funksjonalitet som skal prioriteres i prosjektet. Siden et kart med markører på er en såpass vanlig ting er det lett å tenke at man vet hva man skal gjøre, så derfor jobbet gruppen litt på automatikk i starten, uten å faktisk gå inn i brukerhistoriene og se hva som skulle prioriteres først. Dette ble justert etter hvert, men gruppen kunne med fordel gjort det helt fra begynnelsen. Det er tross alt akseptansekriteriene fra brukerhistoriene som bestemmer om man har utviklet det man skulle.

# Litteraturliste


Aaberge, A. (2021, desember 8). *Brukeren og navigasjon*. Retrieved from NDLA: https://ndla.no/article/5115

Additive. (n.d.). *Persona*. Retrieved Mai 08, 2023, from Additive: https://www.additive.eu/en/glossary/persona.html

Amazon Web Services, Inc. (u.d.). *What is SDLC (Software Development Lifecycle)?* Hentet mai 11, 2023 fra https://aws.amazon.com/what-is/sdlc/

Amsjø, G. (2022, 08 16). *Det Smidige Hjørne*. Retrieved 05 2023, from Det Smidige Hjørne: https://scrummaster.no/2022/estimering/

Babych, M. (2021, Desember 08). *A Review Of The Minimum Viable Product Approach*. Retrieved Mai 09, 2023, from Forbes: https://www.forbes.com/sites/theyec/2021/12/08/a-review-of-the-minimum-viable-product-approach/

Batterbee, I. (2020, Februar 17). *Feature Creep: what is it, and how does it affect the user experience*. Retrieved April 24, 2023, from Uxdesign: https://uxdesign.cc/features-creepers-the-customer-experience-horror-story-124c8fa73edf

Benyon, D. (2013). Designing Interactive Systems: A Comprehensive Guide to HCI, UX and Interaction Design. I *Designing Interactive Systems: A Comprehensive Guide to HCI, UX and Interaction Design* (3. utg., s. 604). Edinburgh Gate, United Kingdom: Pearson Education Limited. Hentet April 5, 2023

Big Water Consulting. (2019). Software Development Lifecycle. Retrieved from https://bigwater.consulting/2019/04/08/software-development-life-cycle-sdlc/

Bigelow, S. J. (2023, Mars). *Azure DevOps*. Hentet fra TechTarget: https://www.techtarget.com/searchwindowsserver/definition/Azure-DevOps-formerly-Visual-Studio-Team-Services

Cain, A. (2017, August 29). *Why Blue Is the World's Favorite Color*. Retrieved April 24, 2023, from Artsy: https://www.artsy.net/article/artsy-editorial-blue-worlds-favorite-color

Cherry, K. (2022, November 22). *The Color Blue: Meaning and Color Psychology*. Retrieved April 24, 2023, from Verywellmind: https://www.verywellmind.com/the-color-psychology-of-blue-2795815

Datatilsynet. (2019, Juli 16). *Risikovurdering*. Retrieved from Datatilsynet: https://www.datatilsynet.no/rettigheter-og-plikter/virksomhetenes-plikter/informasjonssikkerhet-internkontroll/risikovurdering/

Digdir A. (n.d.). *Om risiko og risikovurdering*. Retrieved Mai 10, 2023, from Digdir: https://www.digdir.no/informasjonssikkerhet/om-risiko-og-risikovurdering/3048

DOGA. (u.d.). *Designprosessen*. Hentet Mai 10, 2023 fra Design og arkitektur Norge: https://doga.no/verktoy/designdrevet-innovasjon/guide-for-designdrevet-innovasjon/2/designprosessen/

Entur. (n.d.). *Brukerhistorier*. Retrieved Mai 09, 2023, from Entur: https://design.entur.no/kom-i-gang/for-designere/brukerhistorier

Farley, J. (2010, Mars 3). *Color In Design: Gray*. Retrieved April 25, 2023, from Sitepoint: https://www.sitepoint.com/color-in-design-grey/

Fatrabbit CREATIVE. (n.d.). *Psychology of Black and White and What They Mean for Your Business*. Retrieved April 25, 2023, from Fatrabbit CREATIVE: https://www.fatrabbitcreative.com/blog/psychology-of-black-and-white-and-what-they-mean-for-your-business

Gundersen, D., & Halbo, L. (2018, Mai 28). *Kvalitet*. Hentet fra Store Norske Leksikon: https://snl.no/kvalitet

Holtsmark, T. (2020, Oktober 26). *Rødt*. Retrieved April 25, 2023, from Store norske Leksikon: https://snl.no/r%C3%B8dt

Justis- og beredskapsdepartementet. (2021, Desember 22). Liste over virksomheter med kritisk samfunnsfunksjon og nøkkelpersonell . Retrieved from https://www.regjeringen.no/no/tema/samfunnssikkerhet-og-beredskap/innsikt/liste-over-kritiske-samfunnsfunksjoner/id2695609/

Mathiassen, L., Munk-Madsen, A., Nielsen, P. A., & Stage, J. (2018). *Object Oriented Analysis & Design* (2nd. utg.). Metodica ApS.

Mørstad, E. (2023, Februar 13). *Skisse*. Retrieved from Store norske leksikon: https://snl.no/skisse

McFarland, D. S. (2012). *JavaScript & jQuery: The Missing Manual* (andre. utg.). O'Reilly Media, Inc.

McKenna, D. (2016). The scrum framework. In D. McKenna, *The Art of Scrum: How Scrum Masters Bind Dev Teams and Unleash Agility* (1 ed., pp. 27-34). Aliquippa, Pennsylvania, USA: Apress. Retrieved Mars 23, 2023

Microsoft. (2022, 04 10). *Testing in .NET*. Retrieved 03 2023, from Microsoft Learn: https://learn.microsoft.com/en-us/dotnet/core/testing/

Microsoft. (2023, Februar 14). *Overview of ASP.NET Core SignalR*. Retrieved from Microsoft: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0

Microsoft. (u.d.). *The project triangle*. Hentet 05 2023 fra Microsoft Support: https://support.microsoft.com/en-us/office/the-project-triangle-8c892e06-d761-4d40-8e1f-17b33fdcf810

Nikolov, A. (2017, April 8). *Design principle: Consistency. The most known and most fragile design principle*. Retrieved April 18, 2023, from UX Collective: https://uxdesign.cc/design-principle-consistency-6b0cf7e7339f

Norkart AS. (u.d.). *Komtek*. Hentet Mai 10, 2023 fra https://www.norkart.no/komtek/

Norkart AS. (n.d.). *Om oss*. Retrieved 02 28, 2023, from Norkart: https://www.norkart.no/om-oss/

Normatic AS. (n.d.). *Om Normatic*. Retrieved 02 28, 2023, from Normatic AS: https://www.normatic.no/om-normatic

Parush, A. (2015). The Navigation and Policy Layer. In A. Parush, *In Conceptual Design for Interactive Systems: Designing for Performance and User Experience* (1 ed., pp. 25-36). Morgan Kaufmann. doi:https://doi.org/10.1016/B978-0-12-419969-9.00006-1

Pedersen, K. (2022, November 29). Prosjektbeskrivelse. Kristiansand, Norge.

Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process.* Addison-Wesley.

Scrum Alliance. (u.d.). *The Three Scrum Artifacts and Their Commitments*. Hentet fra Scrum Alliance: https://resources.scrumalliance.org/Article/scrum-artifacts

Scrum.org. (n.d.). *What is Scrum?* Retrieved April 27, 2023, from Scrum.org: https://www.scrum.org/resources/what-scrum-module

Sketch. (2022, Mars 10). *What is a mockup? The definitive guide*. Retrieved April 13, 2023, from Sketch: https://www.sketch.com/blog/what-is-a-mockup/

Skulberg, H., & Aaslid, E. B. (2019, Juni 12). *Begrepet kvalitet – hva dreier det seg om?* Hentet fra Utdanningsforbundet: https://www.utdanningsforbundet.no/var-politikk/publikasjoner/2019/begrepet-kvalitet--hva-dreier-det-seg-om/

Slickplan. (2022, September 15). *Sitemap vs wireframe comparison: what's the difference?* Retrieved April 25, 2023, from Slickplan: https://slickplan.com/blog/sitemap-vs-wireframe

Steinarsson, S. (2014, Februar 28). Downsampling data - not a trivial task. Hentet fra Downsampling data - not a trivial task: https://web.archive.org/web/20140625052324/https://blog.datamarket.com/2014/02/28/downsampling-data-not-a-trivial-task/

TypeScript. (n.d.). *TypeScript*. Retrieved April 25, 2023, from https://www.typescriptlang.org/

Universitetet i Agder. (n.d.). *Bacheloroppgave i informasjonssystemer*. Retrieved 02 28, 2023, from Universitetet i Agder: https://www.uia.no/studieplaner/topic/IS-304-1?year=2022

University of St. Andrews. (n.d.). *Code style and standards guides*. Retrieved 03 2023, from University of St Andrews: https://www.st-andrews.ac.uk/digital-standards/code-standards/

Upland Software. (u.d.). *Estimated Time To Complete (ETC)*. Hentet fra Upland Software: https://uplandsoftware.com/psa/resources/glossary/estimated-time-to-complete-etc/

Uxcel. (n.d.). *Affordance*. Retrieved April 25, 2023, from Uxcel: https://uxcel.com/glossary/affordance

Vihovde, E. H. (2022, 12 28). *API*. Retrieved 05 2023, from Store Norske Leksikon: https://snl.no/API

Wikipedia. (2022, Desember 20). *MoSCoW Method*. Hentet Mai 2023, 11 fra Wikipedia: https://en.wikipedia.org/wiki/MoSCoW_method

Wikipedia. (2023, April 17). *Frontend and backend*. Retrieved Mai 11, 2023, from WIkipedia: https://en.wikipedia.org/wiki/Frontend_and_backend

Wikipedia. (2023, Mars 17). *Integration testing*. Retrieved Mai 15, 2023, from Wikipedia: https://en.wikipedia.org/wiki/Integration_testing

Wikipedia. (2023, Mars 30). *Website wireframe*. Retrieved Mai 11, 2023, from Wikipedia: https://en.wikipedia.org/wiki/Website_wireframe

Wikipedia.org. (2023, 03 25). *Systems Architecture*. Retrieved 05 2023, from Wikipedia.org: https://en.wikipedia.org/wiki/Systems_architecture

Wood, R. (2019, Desember 9). Risk Assessment Matrix. Hentet fra https://www.safran.com/blog/whats-the-difference-between-qualitative-and-quantitative-risk-analysis

Yin, K. (2022, 06 21). *The Ultimate Guide to Architectural Decision Records*. Retrieved 05 2023, from Better Programming: https://betterprogramming.pub/the-ultimate-guide-to-architectural-decision-records-6d74fd3850ee

Zehra, Z. (2022, Desember 1). *What are learnability principles for usability?* Retrieved April 25, 2023, from Educative.io: https://www.educative.io/answers/what-are-learnability-principles-for-usability

# Vedlegg

## Vedlegg A. Gruppekontrakt

# Gruppekontrakt

**Medlemmer**

Vi, gruppe nr 1. består av følgende medlemmer:


Garcia, Karen Johanna Thorsen: karent13@uia.no
Leite, Johannes: johanl14@uia.no
Lotsberg, Erlend Dyrøy: erlenddl@uia.no
Western, Joachim: joachimw@uia.no
Woo, Tai-Gwen: taigwengw@uia.no


**Skal jobbe sammen om følgende prosjekt:** Bacheloroppgave i Norkart: Visualisering av sensordata fra KOMTEK Drift og Vedlikehold. Oppgaven vil bestå i å nyttiggjøre sensordata fra pumpestasjoner o.l. og vise disse i et kart samt i et Dashboard.

**Bakgrunn og motivasjon**

Vi ønsker å lære om, og mestre fagets læremål, slik de er oppgitt på Canvas, samt tilegne oss erfaring ved å jobbe i et reelt prosjekt i arbeidslivet i samarbeid med profesjonelle aktører.

**Bestemmelser om fremmøte og samarbeid**

Gruppemedlemmene forplikter seg til at de vil:

- Delta i gruppearbeid, samlinger og øvelser, be om/ta imot veiledning, gjøre selvstudium som kreves for å tilegne seg kunnskap og ferdigheter til å klare å fullføre arbeidsoppgaver man blir tildelt.
- Overholde interne og eksterne frister, og levere oppgaver som publiseres på Canvas innen fristene.
- Sørge for at arbeid fordeles jevnt mellom gruppemedlemmer, men samtidig utnytte hver enkelt students spesielle ferdigheter og bakgrunn.

Evt. konflikter rundt samarbeid og innsats søkes løst gjennom diskusjon i gruppen. Fører ikke

dette frem, kontaktes veileder eller foreleser, så snart som mulig, og senest før frister. For konflikter eller saker som ikke eksplisitt er nevnt i denne gruppekontrakten avgjøres saken ved avstemning, hvor flertallet bestemmer.

Gruppemedlemmer som ikke innretter seg etter bestemmelsene i gruppekontrakten vil i første omgang få en muntlig advarsel, deretter en skriftlig advarsel. Ved tredje brudd på gruppekontrakten kan gruppemedlemmet kastes ut av gruppen. Det er flertallet i gruppen som bestemmer om sanksjoner skal iverksettes for konkrete brudd på gruppekontrakten.

**Arbeidstider**

Gruppen har besluttet å se på bacheloroppgaven som en jobb. Som i enhver annen jobb gjelder krav om 100% oppmøte, med mindre fravær kan begrunnes med gyldig grunn.

Arbeidstider er primært 8 timer (inkl. 30 min lunsj) mandag, tirsdag og torsdag, i tillegg kan det kreves ekstraarbeid dersom det viser seg nødvendig. Gruppen praktiserer fleksitid med kjernetid 9 - 14. Det er tillatt å jobbe fra hjemmekontor, men dette skal meldes fra om, aller helst på forhånd.

Uavhengig av om man jobber fra hjemmekontor eller på Norkart-kontoret skal man delta på daily standup. Daily standup tidsfestes til klokken 09:15 på kontordagene.

**Andre bestemmelser**

Til gruppeleder er valgt: Erlend Lotsberg, til viseleder, som også har ansvar for eventuelle planleggingsverktøy er valgt Tai-Gwen Woo.

Gruppeleder leder gruppemøtene og sørger for at alle er med i arbeidet. Er det noe som må diskuteres, skal gruppeleder styre diskusjonen, og sørger for at alle blir hørt. Etter en diskusjon av en lengde som passer til temaets viktighet, avveid mot behovet for å ta en beslutning og komme videre i prosjektet, tas det en omforent beslutning. Er det uenighet, foretas det en avstemming, der gruppeleder har dobbeltstemme ved stemmelikhet. Etter at beslutninger er fattet skal alle forholde seg lojalt til disse.

Gruppeleder er gruppens kontaktperson utad, nestleder er hovedansvarlig for fremdrift og for at frister for innlevering overholdes. Gruppeleder er også hovedansvarlig for å holde kontakt med og arrangere møter med eksterne partnere, og med intervjuobjekter/samtalepartnere. Gruppeleder eller viseleder fører logg/kort møtereferat, om hva vi ble enige om, hva gjorde vi og hva gjør vi neste gang. Av referatet skal fremgå: fremmøte, forfall (meldt på forhånd, med grunn), fravær (ikke møtt/uten oppgitt grunn).

Gruppeleder eller viseleder Erlend Lotsberg/Tai-Gwen Woo har ansvar for å innkalle til møter.

Sted, dato: Kristiansand, 09.01.2023
Underskrifter:

| | |
|---|---|
|  <br>Karen Johanna Thorsen Garcia |  <br>Joachim Western |
|  <br>Johannes Leite |  <br>Tai Gwen Woo |
|  <br>Erlend Dyrøy Lotsberg | |

## Vedlegg B. Sprint Retrospective

# Sprint 1 retrospekt

02.02.2023

Hva har gått bra:

- Fordeling av oppgaver
- Estimering
- Samarbeid innad i gruppen/med Norkart. Bra kontormiljø.
- Gode forberedelser
- Har nådd målet for sprint 1
- Overholdt tidsfrister
- Vi har lært mye, hatt det gøy
- God kontroll
- Utvikling av skisser og design
- Arbeidstidene har fungert fint
- Scrumprossesene

Hva kan vi forbedre:

- Ikke bruke for mye tid på en oppgave, der det lar seg gjøre.
- Være flinke til å ta pauser med jevne mellomrom
- Bruke mindre tid i lunsjpausen
- Spør mer hvis man føler seg helt lost/fortapt/ubrukelig
- Være sjenerøse med bruk av tid på å lære bort til andre. Vi ligger godt an, og har tid til det.
- Skrive mer i bachelorrapporten

Tiltak for å gjøre det bedre til neste gang:

- Ikke bruke ChatGPT
- Ta i bruk dokumentasjon
- Spør mer hvis man føler seg helt lost/fortapt/ubrukelig
- Være sjenerøse med bruk av tid på å lære bort til andre. Vi ligger godt an, og har tid til det.
- Vær fornuftig med tidsbruk i lunsjpausen
- Git blame teacupmonkey -& erlendurr
- Git gooder
- Legge til flere skriveoppgaver i backlog

# Sprint 2 retrospekt

16.02.2023

Hva har gått bra:

- Begrenset og kritisk bruk av ChatGPT. Bruker andre kilder som for eksempel dokumentasjon
- Har blitt bedre på å spørre om hjelp og å hjelpe andre.
- Har fått terpet mer på git og blitt bedre.
- Har nådd product increment for sprint 2
- Fordeling av oppgaver
- Jevnt arbeid på rapportskriving
- Utviklingsutvikling
- Tais nøkkelkort fungerer

Hva kan vi forbedre:

- Kodegjennomgang (code review)
- Møtedisiplin
- Arbeide litt mindre i våre egne siloer
- Ikke sitte for lenge med ett problem
- Ikke nok oppgaver i backlog
- Bli mer bevisst på sammenhengen mellom brukerhistoriene og Product Backlog Items
- Notere ned viktige avgjørelser

Tiltak for å gjøre det bedre til neste gang:

- Sette opp møter for å gå gjennom kode
- Sett opp møte for å utforske neste steg når det begynner å bli tomt i backlog
- Bruke første dag i ny sprint til planlegging og code review
- Bruke brukerhistoriene mer aktivt i sprintplanlegging. Gå gjennom akseptansekriterier.
- Avgjørelsesgrunnlag, oppdater wiki, ta det med i backlog

## Sprint 3 retrospekt

09.03.2023

Hva har gått bra:

- Har blitt bedre på å spørre om hjelp og å hjelpe andre.
- Fordeling av oppgaver basert på interesse og kompetanse
- Jevnt arbeid på rapportskriving
- Code review har fungert fint, bedre innsikt i hva hverandre har gjort

- Sprint planlegging har gått bra, og vi har brukt brukerhistoriene mer aktivt (prioritering og akseptansekriterier)
- Vi har dokumentert beslutningsgrunnlag i en egen ADR
- Karen har sluttet å drikke så mye kaffe
- Flinke til å ta pauser

Hva kan vi forbedre:

- Mer fokus på oppgaver som er prioritert
- Fokus på å nå product increment
- Mindre sosial loffing
- Ikke bruke for lang tid hvis man er fast i en oppgave

Tiltak for å gjøre det bedre til neste gang:

- Plasser de mest prioriterte oppgavene øverst, marker oppgavene med prioriteringstall
- Økt fokus og selvdisiplin, selv om oppgavene i backlog er uinteressante
- Spør om hjelp, eller ta et valg, hvis man sitter fast i en oppgave
- Skille mellom arbeidstid og pausetid

## Sprint 4 retrospekt

23.03.2023

Hva har gått bra:

- Flinke til å spør om hjelp, spurt Cato masse
- Arbeide sammen på oppgaver
- Kuttet ned på matpausen, litt, noen ganger
- Nådd sprint-målet, funksjonalitet er på plass

Hva kan vi forbedre:

- Nivået på Mario Kart
- Fokus
- Bli flinkere til å jobbe med én ting av gangen

Tiltak for å gjøre det bedre til neste gang:

- Plasser de mest prioriterte oppgavene øverst, marker oppgavene med prioriteringstall
- Fokusere på en spesifikk oppgave

## Sprint 5 retrospekt

13.04.2023

Hva har gått bra:

- Vi føler at vi er stabile og har kontroll over prosjektet
- Vi har klare mål og oppgaver fra backlog
- Jevn progresjon i arbeidet vårt
- God oppgavefordeling
- Godt samarbeid
- Godt arbeidsmiljø
- Forbedret oss i Mario Kart, Joachim er ikke lenger først hver eneste runde, JO!!!!!!!!

Hva kan vi forbedre:

- Prioritere rapportskriving

Tiltak for å gjøre det bedre til neste gang:

- Involvere alle i rapportskriving

## Vedlegg C. Risikovurdering

**Risikoanalyse VA-sensorteknologi bachelorprosjekt**

| Hendelse | Sannsynlighet | Konsekvens | Risiko | Avbøtende tiltak |
|---|---|---|---|---|
| Ikke bli ferdig med prosjektet i tide | 2 | 2 | 4 | Se feature creep |
| Ikke bli ferdig med rapporten i tide | 1 | 5 | 5 | Begynt på rapport fra dag 1<br>Satt opp disposisjon<br>Føre "dagbok" for å lette skriving senere<br>Planlegge skriving inn i backlog for å sette av tid |
| Problemer med git | 4 | 2 | 8 | Har etablert felles rutiner for push/pull/branching<br>Git-master<br>Ikke jobbe i samme deler av prosjektet samtidig |
| Støter på problemer som er for vanskelige | 5 | 4 | 20 | Selvstudium/kompetanseheving<br>Lese dokumentasjon og tutorials<br>Samarbeide med andre på gruppen<br>Lav terskel for å spørre om hjelp fra veiledere |
| Feature creep | 2 | 3 | 6 | Fokusere på et slankt MVP<br>Modulær tilnærming<br>Begynne smått, legge til features etter hvert |
| Sosial loffing | 1 | 4 | 4 | Tydelige krav til arbeidsinnsats<br>Fleksitid for å lette reiser o.l.<br>Behandle oppgaven som en jobb<br>Sitte på felles kontorplass |
| Feil estimering | 4 | 2 | 8 | Bruke et mer "grovt" estimerinssystem<br>Være fleksible og endre estimater etter behov<br>Lære av tidligere erfaringer<br>Heller sikte på å ha et "slankt" prosjekt som kan utvides enn å ha et enormt prosjekt som ikke blir ferdig |
| Manglende oppfølging | 2 | 4 | 8 | Være aktive og spørre om hjelp<br>Selv ta initiativ til møter vi trenger<br>Selv styre prosjektet, og ikke bli passive<br>Avtale faste veledningsmøter |
| Dårlig kontroll på om koden virker | 3 | 3 | 9 | Fokus på enhetstesting<br>Har hatt lynkurs i testing<br>Rutiner for at hver programmerer tester egne klasser |
| Dårlig kodekvalitet | 3 | 3 | 9 | Etablere felles standarder for kode<br>Gjøre periodiske code reviews |
| Gruppemedlemmer glemmer avtalte rutiner | 4 | 2 | 8 | Laget Wiki som man kan lese for å huske hva man skal gjøre |

## Vedlegg D. Personas

### Erik Ekstern aktør

Erik eier flere bygg og vil ha en total oversikt på drift og vedlikeholdsområdet. Som eier av byggene, er det viktig at Erik kjenner til områdene og er ansvarlig for å ha denne oversikten. Dersom noe skulle oppstå, vil han vite nøyaktig hvor. Dette gjelder også driftsoppgaver innen vannledninger, avløpsledninger og kummer i eiendommene.

### Julie Rørlegger

Av og til oppstår det lekasjer eller fortettning. Rørlegger Julie må derfor vite nøyaktig hvor feilen ligger. Det gjelder også oversikt på vannledninger, avløpsledninger og kummer, uansett eiendom.

### Nils V/A avdelingen

Nils driver drift og vedlikeholdsområdet for en hel kommune. Som ansvarlig og som må sørge for å ha både oversikt og kontroll, forventes det at Nils kjenner til alle vannledninger, avløpsledninger og kummer kommunen har.

Dersom noe skulle oppstå, vil han også vite nøyaktig hvor. Driftoppgaver og vedlikeholdsopppgavermed historiske data kan fortelle han hvor problemet oppsto eller prediktere noe som kommer til å skje. Med prediksjon, kan en problematisk hendelse forhindres i god tid.

### Pål Tilsynsmann

Tilsynsmannen Pål må foreta kontroll av V/A, samt se på risikoen for forurensning av miljøet og spredning av sykdom ved å sikre godt drikkevann og god vannkvalitet.

Han trenger faktiske sanntidsdata som kan fortelle han noe om vannkvaliteten, men også dersom forurensning skulle oppstå. Som tilsynnsmann, er han ansvarlig at kontrollene blir foretatt; slik at innbyggerne er sikret god vannkvalitet og miljø.

### Tore Kommunen

Drift og vedlikeholdsområdet Tore står ansvarlig for er stort, men er ofte avhengig av å kjøpe hjelp fra eksterne aktører som bistår både i form av teknisk ferdighet og videresending av data, selvom han har tilgang til systemet.

Tore står ovenfor en del krav i arbeidet sitt. For at sjefen hans (inkl. innbyggerne i kommunen) skal være sikre på at Tore gjør det han blir lønnet for, må han ha kontroll over driftsoppgaver innen vannledninger, avløpsledninger og kummer i kommunen.

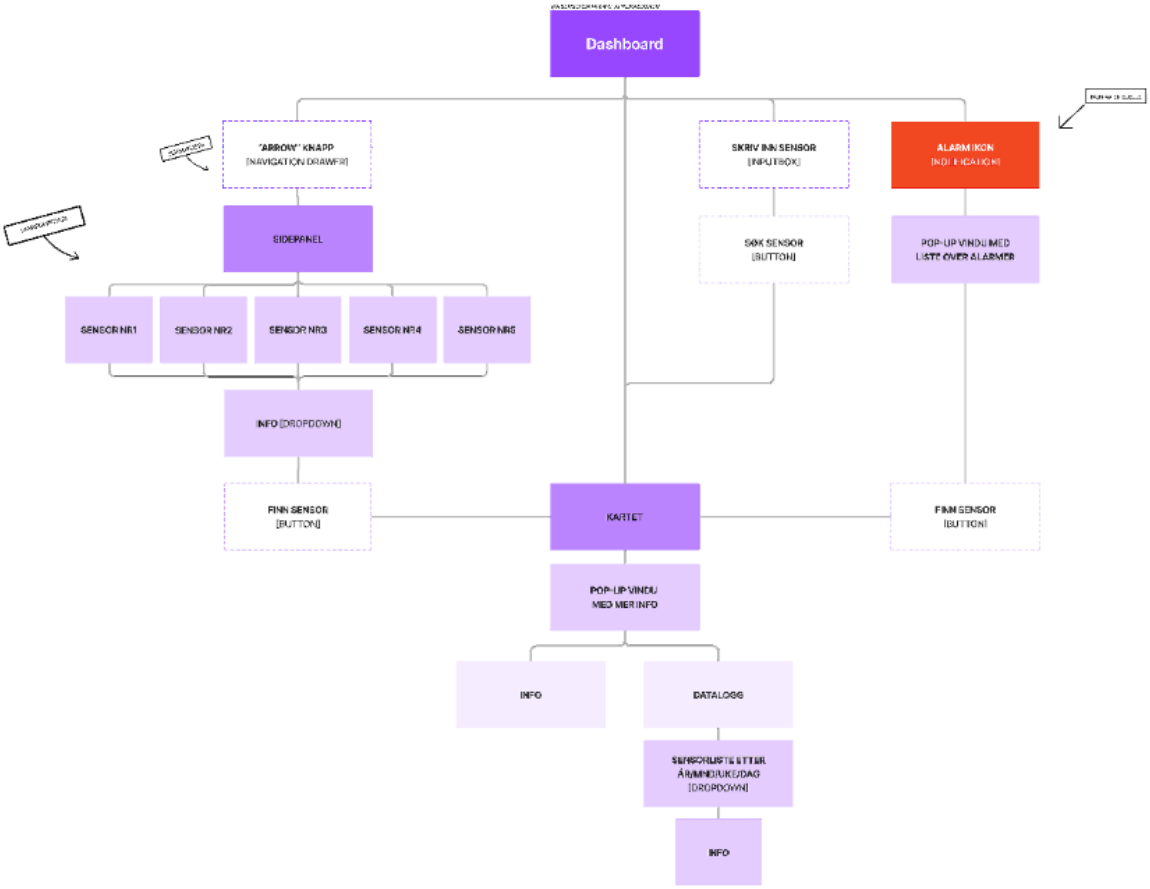
## Vedlegg E. Brukerhistorier

| MoSCoW | Brukerhistorie | Løsningsbeskrivelse | Akseptansekriterium | Argument |
|---|---|---|---|---|
| **Must have** | 1# Som bruker ønsker jeg oversikt over sensorene slik at jeg kan ha kontroll over hvor disse befinner seg. | En web-app som gir informasjon om sensorene og oversikt i form av kart og informasjonsliste. Sensorene blir representert i form av ikoner/popups. | Gitt at jeg bruker web-appen, skal jeg ha:<br>• Liste med sensorer, dropdown panel<br>• Kart med oversikt over sensorer | Enkel oversikt over sensorene er viktig for å lokalisere sensorene og lese av tilhørende sensordata. Anses som must have. |
| **Could have** | 2# Som bruker ønsker jeg å se status på sensorene slik at jeg har oversikt og kontroll på status til hver sensor. | Sensorikoner med ulike farger. De ulike fargene representerer de ulike statusene. Status på sensorene skal også beskrives i form av tekst og vises i et popup-vindu. | Gitt at jeg bruker web-appen, skal jeg ha:<br>• Sensorikoner på kartet med ulike farger som representerer tilhørende status på sensorene.<br>• Mulighet til å trykke inn på enkelte sensor og se status (popup-vindu) | Kritisk informasjon, f.eks. feilmelding fra sensor, kommuniseres gjennom status. Grunnen til at det er satt som could have er fordi vi prioriterer sanntids- og historiske data først, da vi med produkteier har kommet frem til at dette gir mest verdi. |
| **Must have** | 3# Som bruker ønsker jeg å se temperaturmåling (i sanntid) fra hver sensor slik at jeg har oversikt over temperaturen til hver pumpestasjon. | Temperaturmålinger blir vist i et popup-vindu i form av tall/tekst. | Gitt at jeg bruker web-appen, skal jeg ha:<br>• Mulighet til å trykke inn på hver sensor og lese av tilhørende sanntidsmålinger (popup-vindu). | Vannets temperatur sier noe om dens kvalitet og forhold. Oversikt over vannets temperatur er derfor et must have. |
| **Must have** | 4# Som bruker ønsker jeg å kunne se enhetsnummer til sensorene slik at jeg har kontroll | Enhetsnummer blir vist i et popup-vindu i form av tall/tekst. | Gitt at jeg bruker web-appen, skal jeg ha:<br>• Mulighet til å lese av tilhørende | Enhetsnummer er et must have for at brukerne skal kunne gjenkjenne spesifikke sensorer. |

| | | | | |
|---|---|---|---|---|
| | på hvilken sensor som må fikses. | | enhetsnummer på markøren i kartet. | |
| **Should have** | 11# Som bruker ønsker jeg å se historiske data om hver av sensorene slik at jeg kan ha oversikt over svingninger i målinger. | Grafer i en modal hvor man kan bytte mellom ulike tidsperspektiver (uke, dag, måned etc.). | Gitt at jeg bruker web-appen, skal jeg ha:<br>• Enkelt og oversiktlig grensesnitt for å se historikken til hver sensor. | Historikk kan blant annet brukes til å se om det er sammenheng mellom målinger og hendelser. Dette kan ha viktig fremtids- og skaleringsverdi og regnes derfor som en should have. |
| **Should have** | 6# Som bruker ønsker jeg at systemet skal være enkelt å bruke fordi jeg ikke er teknisk anlagt. | Et brukergrensesnitt utviklet med etablerte designprinsipper. | Gitt at jeg bruker web-appen, skal jeg:<br>• Enkelt kunne navigere og bruke applikasjonen | Ettersom prosjektet vårt er et Proof of Concept og vi ikke har reelle brukere, har vi nedprioritert brukeropplevelsen til should have. |
| **Should have** | 7# Som bruker ønsker jeg at systemet skal ha et behagelig utseende, slik at brukeropplevelsen ikke dempes av utseendet. | Et brukergrensesnitt utviklet med etablerte designprinsipper. | Gitt at jeg bruker web-appen, skal den være:<br>• Behagelig å se på<br>• Ha balansert bruk av farger | Ettersom prosjektet vårt er et Proof of Concept og vi ikke har reelle brukere, har vi nedprioritert brukeropplevelsen til should have. |
| **Could have** | 5# Som bruker ønsker jeg å bli varslet om dersom sensorene oppdager noe unormalt slik at jeg har oversikt og kan respondere riktig. | Få varsel i form av notifikasjon gjennom grensesnittet. | Gitt at jeg bruker web-appen, skal jeg ha:<br>• Varselbjelle i grensesnittet som lyser rødt når det er varsel. | Sensorene kan oppdage kritiske forhold som krever umiddelbar handling. Varsler er et must have for å opplyse og gi oversikt over sensorer som trenger oversyn. |

| | | | | |
|---|---|---|---|---|
| **Could have** | 8# Som bruker ønsker jeg å kunne predikere og vite hvilke sensorområder som har sannsynlighet for å oppstå problemer, slik at man kan hindre problemene før de oppstår. | Maskinlæring ved å analysere historisk data. | Gitt at jeg bruker web-appen, skal jeg kunne:<br>• Se historisk data<br>• Se hvis en sensor ofte oppdager problemer | Hadde vært interessant, men ikke noe som er nødvendig eller som bør prioriteres. Er derfor prioritert som could have. |
| **Could have** | 9# Som bruker ønsker jeg å kunne søke opp spesifikke sensorer slik at jeg kan lett finne frem en sensor jeg har i tankene. | Et søkefelt hvor man kan søke opp spesifikke sensorer. | Gitt at jeg bruker web-appen, skal jeg kunne:<br>• Søke opp spesifikke sensorer | Ikke et stort behov ettersom vi har få sensorer. I tillegg skal brukerhistorie 1# gi god nok oversikt over sensorene. Er derfor prioriert som could have. |
| **Will not have** | 10# Som bruker ønsker jeg oversikt over vannledninger, avløpsledninger og kummer, slik at jeg kan ha kontroll over hvor disse befinner seg | Et kart som viser og gir oversikt over vannledninger, avløpsledninger og kummer. Farger og streker for å illustrere hvor de befinner seg. | Gitt at jeg bruker web-appen, skal jeg kunne:<br>• Trykke på en knapp for å se, eller ikke se, vannledninger, avløpsledninger og kummer. | Vi har ikke data over vannledninger, avløpsledninger og kummer. Vi kommer derfor ikke til å inkludere det i denne omgang. |

## Vedlegg F. Navigation Map




Beklager lesbarheten, vi mistet originalfilen.

## Vedlegg G. Egenvurdering

**Samarbeid**: Gruppen føler samarbeidet har fungert meget bra. Den består stort sett av litt eldre personer, hvor de fleste har erfaring fra høyere utdanning fra tidligere. Gruppemedlemmene er åpne og ærlige med hverandre, og er ikke redde for å stille krav til hverandre. Gruppen har også en sammensetning hvor medlemmene har litt forskjellige interesseområder og kompetanser, noe som gjør at medlemmene utfyller hverandre på en god måte. I tillegg kommer medlemmene godt overens sosialt.

**Johannes Leite**: Mitt eget bidrag til bachelorprosjektet har vært variert, men jeg har fokusert litt ekstra på mitt favorittområde innen IT som er frontend-utvikling. Teknologiene vi har valgt å bruke i prosjektet er ting jeg skal bruke i jobb til høsten, så derfor har det vært ekstra spennende og nyttig for meg å holde på med dette hos Norkart gjennom våren.

**Erlend Lotsberg**: Mitt individuelle bidrag til bachelorprosjektet har i hovedsak vært backend-programmering, det er i for det meste der mine interesser ligger. Jeg har likevel også gjort noe frontend-programmering. I tillegg til dette har jeg bidratt til rapportskriving, og har fungert som gruppeleder og kommunikasjonsansvarlig.

**Joachim Western**: I dette prosjektet har jeg vært i størst grad del av utviklingen av applikasjonen. Da prosjektet startet så var jeg med på systemutviklingen, hvor jeg utviklet mest i backend, men også i frontend-delen av prosjektet. Gjennom prosjektet har det vært god stemning i gruppa og dette føler jeg har hatt en stor positiv innvirkning på prosjektet. Det har gjort det lettere for meg å diskutere og komme med innspill, ettersom det har vært en stor likhet i gruppa.

**Karen Thorsen**: Ved dette prosjektet har jeg hatt ansvarsområdet for idéutviklingen av designet. Dette gjelder innen de ulike områdene med personas (for å skaffe en forståelse av brukeren), research, skissering, utvikling av navigation map, wireframes og tilslutt UX-design av mockup; med fokus på designprinsippene og universell utforming. I tillegg til dette har jeg bidratt med noe utvikling innen frontend, samt skriving av rapporten.

**Tai-Gwen Woo**: Har hovedsakelig ledet gruppen som Scrum-Master under hele prosjektet. Samtidig har jeg vært med på å komme med innspill på design og vært noe involvert innen frontend utviklingen. Videre har jeg hatt ansvar for å dokumentere arbeidet som ble gjort.  Mitt største fokus under prosjektet har vært kartlegging av brukerbehov og utviklingen av brukerhistorier. Videre har jeg passet på at brukerhistoriene blir anvendt riktig for å sikre kvalitet.

## Vedlegg H. Uttalelse fra Norkart

Kristiansand 16.05.2023

NORKART


Hovedkontor Skøyen
Hoffsveien 4
0275 Oslo
Telefon: +47 67 55 14 00

Avd.kontor Kristiansand
Markens gate 19
N-4611 Kristiansand
Telefon: +47 67 55 14 00

Avd.kontor Bergen
Inger Bang Lunds vei 12
5059 Bergen
Telefon: +47 67 55 14 00

Avd.kontor Trondheim
Postboks 1261 Sluppen
7462 Trondheim
Telefon: +47 67 55 14 00

Avd.kontor Lillehammer
Fåberggt. 155
2615 Lillehammer
Telefon: +47 67 55 14 00

*Org.nr. NO 934161181 MVA.*
*Bankgiro.*
*9490.05.27025*
*E-post:* info@norkart.no
*Internett:* www.norkart.no


### BACHELORGRUPPE VÅREN 2023

Erlend Dyrøy Lotsberg, Joachim Western, Johannes Leite, Karen Johanna Thorsen Garcia og Tai-Gwen Woo har fra januar 2023 frem til nå tatt Bacheloroppgaven sin hos oss i Norkart Kristiansand. Oppgaven bestod overordnet av å motta sensor data fra et eksternt firma (Normatic) og presentere disse på en hensiktsmessig måte i et kart. Vi har lagt lite føringer på oppgaven utenom at vi ønsket at gruppen brukte samme teknologi som oss (.net, react, leaflet og azure).

Vi fortalte tidlig i prosjektet hvordan produktteam i Norkart jobber, og har med glede registrert at gruppen i stor grad har valgt å jobbe med en tilsvarende metodikk. Dette inkluderer en Discovery-fase i starten av prosjektet for å begrense arbeidet som skulle gjøres. Bygge opp en god backlog i Azure Devops. Regelmessige status-møter, bi ukentlige demo og retrospekt m.m. Gruppen har også fått tildelt nødvendige ressurser i Azure for å kunne lage CI/CD løyper og rulle ut koden automatisk til produksjon.

Vår oppfatning av studentene er at de er en flott gjeng studenter. De har alle ulike preferanser på hva de innenfor et produktteam ønsker å jobbe med og har best kunnskap om, men har samlet løst oppgaven på en meget god måte. De har jobbet selvstendig, men har søk råd og hjelp ved behov. Både funksjonelt og teknologisk har oppgaven blir løst meget bra, og de har anvendt relevant teknologi.

Vi ønsker studentene lykke til med avslutning av studiet og videre ut i arbeidslivet og håper tiden de har vært hos oss har vært lærerik.

I tillegg til undertegnede så har Cato Martinussen (Tech lead) veiledet studentene med teknologiske og prosess-problemstillinger.

Med vennlig hilsen

Kurt Pedersen


**Kurt Pedersen**

*Teamleder*

kurt.pedersen@norkart.no

*+47 98 25 78 95*