UNIVERSITETET I AGDER

Egde Flåtestyringssystem og datainnsamling
IS-304: 2018

Tittel:

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig:	Hallgeir Nilsen
Veileder	Hallgeir Nilsen
Oppdragsgiver:	Egde Consulting

Studenter:

Etternavn	Fornavn
Blomvik	Brynjar
Leonhardsen	Frode
Østern	Erlend Kasin

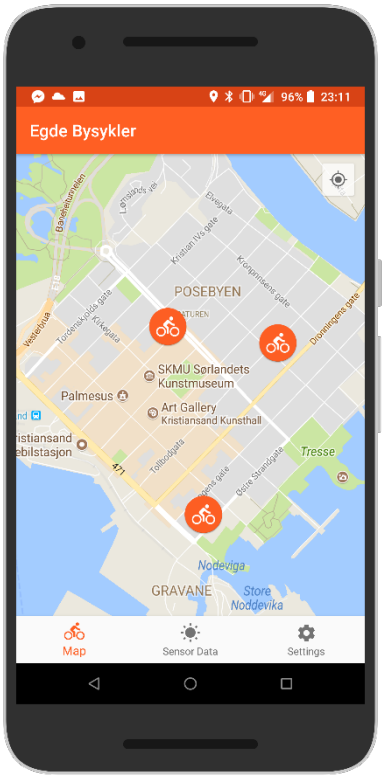
Jeg/vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	JA X	NEI___
Kan besvarelsen brukes til undervisningsformål?	JA X	NEI___
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	JA X	NEI___

Bachelor-rapport IS-304

Egde Consulting

Flåtestyring og innsamling av sensordata

Av gruppe 16
-
Erlend Kasin Østern
Brynjar Blomvik
Frode Leonhardsen



Bitbucket server repo: https://bitbucket.org/bblomvik/egdeflatestyringsystem/src/master/
Bitbucket Android repo: https://bitbucket.org/froleo/egde-flatestyring-android/src/master/

Forord

Denne bacheloroppgaven markerer slutten på vårt treårige løp på IT og informasjonssystemer ved Universitet i Agder. Vi startet forberedelsene allerede høsten 2017 med stiftelse av grupper og valg av oppgave.

Ingen av gruppens medlemmer har hatt nevneverdig erfaring med prosjekter innen systemutvikling utover tidligere emner i studiet. Dette tilsier at det har vært en bratt læringskurve for alle gruppemedlemmer.

Prosjektets mål har vært todelt. Mål nummer én var å designe og utvikle et datasystem for å holde oversikt over og administrere en flåte av kjøretøy. Mål nummer to er å bruke disse kjøretøyene til å samle inn sensordata som for eksempel temperatur, luftkvalitet i det området kjøretøyet befinner seg i og deretter lagre dataene til en database og i etterkant kunne analysere dataene i stor skala.

Rapporten omhandler oppdraget vi har fått, problemene og utfordringene vi har møtt på og våre valg for å overkomme nevnte hindringene.

Vi vil gjerne takke:
Våre samarbeidspartnere i Egde Consulting
Gisle Stavland
Trond Kvarenes
Elin Kilen

Vår veileder ved Universitet i Agder
Hallgeir Nilsen

Vår kontakt i Telia for hjelp med NB IoT
Even Fuglestad

Sammendrag

Denne rapporten er dokumentasjonen av vårt bachelorprosjekt i fagemnet *IS-304: Bachelor i informasjonssystemer ved Universitetet i Agder.*

Vår oppdragsgiver ønsker en multi-funksjonell teknisk løsning som overvåker og administrerer en flåte av kjøretøy samtidig som nevnte kjøretøy samler og registrerer diverse sensor data til en database. Derfor består bachelorprosjektet vårt av to problemstillinger som vi har kombinert til én oppgave.

1. Utvikle et system for å administrere og styre en flåte av kjøretøy.
2. Samle inn diverse/ulike sensordata i de omgivelsene kjøretøyene befinner seg i.

Vi startet med å ta utgangspunkt i bysykler som case-eksempel. I et slikt scenario jobber vi med kritiske begrensninger som blant annet vekt, batteritid, nettilgang/dekning, levetid, fysiske påkjenninger som vær og slitasje) og sikkerhet (tyveri av sykler/maskinvare). Etter hvert som vi jobbet med prosjektet, og i dialog med oppdragsgiver fikk vi spisset omfanget av prosjektet til å omhandle nettopp flåtestyringsadministrasjon med datainnsamling. Vår oppdragsgiver ønsker å få dette systemet implementert som et generisk rammeverk. På den måten kan det tilpasses en rekke ulike bruksområder. Det kan være alt fra kommunale nyttekjøretøy, utrykningskjøretøy, boliger, booking av møterom etc.

I den tidlige fasen av prosjektet ble det viet mye tid til problemstillinger og brukerhistorier som ikke er direkte relevant til den endelige prosjektspesifikasjonen. Dette ser vi på som en del av prosessen å, sammen med oppdragsgiver gjennom god kommunikasjon, komme til bunns i hvilke funksjoner oppdragsgiver vil at systemet skal ha, og at alle involverte parter er på samme side hva angår prosjektets mål og omfang.

Vi benyttet oss i starten av scrum som arbeidsmetode, men endret etter hvert til Kanban. Dette ble endret fordi mange av oppgavene vi jobbet med var lite kompatible med det rigide rammeverket scrum-metoden skilter med, hva angår estimat av tid og kompleksitet. Kanbans fleksibilitet rundt tidsestimering viste seg å gangne oss i dette prosjektet.

Systemet og Android mobilapplikasjonen er utviklet i Java. Databasen er skrevet i MySQL, og kjøretøy/sensorenhetene er Arduinobasert og dermed utviklet i Arduinos egne C/C++ baserte programmeringsspråk Arduino.

Vi har hatt hyppig kontakt med oppdragsgiver gjennom status og demo-møter annenhver uke, og løpende dialog via e-post. Vi er svært fornøyde med Egde som oppdragsgiver da de har vært veldig gode til å gi respons på våre henvendelser og oppfølging underveis i prosjektet.

Vi har tilegnet oss mye ny og nyttig kunnskap, både om nye og aktuelle teknologier, men også innen prosjektstyring, planlegging, kommunikasjon og ikke minst om hvordan det virkelige arbeidslivet fungerer i en bedrift som Egde Consulting.

Innholdsfortegnelse

Figurliste

1. Innledning

1.1 Om gruppen

Vår gruppe består av Brynjar Blomvik, Frode Leonhardsen og Erlend Kasin Østern. Denne gruppen ble etablert ved at to av oss hadde jobbet på gruppe sammen i tidligere faglige oppgaver, og at sistemann var på utkikk etter en gruppe å jobbe med. Siden en bacheloroppgave er et omfattende stykke arbeid, var det ikke ideelt å bare være to personer på en gruppe. Dermed passet det perfekt med et ekstra tilskudd til gruppen.

Gruppens kunnskapsnivå innen applikasjonsutvikling er spredt, der Frode har mest erfaring av oss tre. I tillegg har Brynjar noe erfaring med Arduino og sensorene som vi har tatt i bruk. Men i en så omfattende oppgave, der systemene skal kommunisere med hverandre blir det en bratt læringskurve for oss alle.

1.2 Oppdragsgiver

I denne oppgaven har vi hatt et godt samarbeid med Egde Consulting. De har kontorer både i Kristiansand og i Grimstad, der Grimstad er deres hovedkontor. Vi kom i kontakt med Egde ved at de la ut et oppdrag på Kompetansetorget, som vi syntes var interessant. Hos Egde har vi samarbeidet med Gisle Stavland, Trond Kvarenes og Elin Kilen.

1.3 Prosjektbeskrivelse

1.3.1 Flåtestyring

Egde ønsker å utvikle et rammeverk for flåtestyring av kjøretøy. Målet er at rammeverket skal være fleksibelt og skal kunne brukes for flere typer kjøretøy, men også styring av andre typer ressurser. Eksempel på kjøretøy kan være bysykler, kommunale kjøretøyer som parkvesen, hjemmesykepleie, utrykningskjøretøy, kjøretøy til bedrifter som for eksempel rørlegger og snekkerfirma eller lignende. Andre ressurser kan være for eksempel møteromsbooking, utleie av boliger, kontorlokaler, bygninger helt opp til store geografiske områder. Fra et delingsøkonomisk perspektiv kan rammeverket for eksempel legge grunnlaget for et system der brukere kan registrere sin bolig, hytte, båt eller bil til utleie for andre registrerte brukere.

Selv om rammeverket ideelt sett skal kunne styre et utvidet sett med ressurser kommer vi i dette prosjektet og denne rapporten til å fokusere på styring av kjøretøy og innsamling av sensordata i den forbindelse.

1.3.2 Sensorer

For å kunne styre flåten må alle kjøretøy utstyres med en GPS-enhet som ved jevne mellomrom rapporterer posisjon til systemet. I tillegg til GPS skal kjøretøyene også utstyres med sensorer for å måle forskjellige forhold i omgivelsene kjøretøyet befinner seg i. Vi kommer til å bruke temperatur og luftfuktighet i denne oppgaven. Målet er å utvikle systemet slik at vi med enkelhet kan bytte ut eller tilføye andre typer sensorer etter behov.

Dataene som samles inn fra sensorene har flere potensielle bruksområder. De vil kunne brukes eksempelvis av kommunen til byutvikling. Ved å bruke GPS-data til å analysere reisemønster kan man finne ut hvor det er mest hensiktsmessig å utbedre veinettet, bygge nye veier, sykkelstier, boligfelt osv. De vil kunne bruke data om luftkvalitet til å finne ut hvor luftkvaliteten er lav, og dermed kunne slutte miljøbevisste tiltak deretter.

1.3.3 Systemet som helhet

Den komplette løsningen innebærer flere teknologiske elementer som trådløs kommunikasjon via mobilt nettverk, automatisk rapportering av posisjon og innsamlede sensordata, bruk av database og analyse av big data. Vi har også utviklet en app til Android for at brukerne skal blant annet kunne hente ut kjøretøy. Appen kommer også til å være et sentralt verktøy for alle som jobber i administrasjon og vedlikehold av systemet.

Med andre ord er dette et omfattende prosjekt hvor vi nok ikke har kapasitet til å utvikle en komplett løsning. Vi vil derfor fokusere på å få på plass de største, mest grunnleggende og viktigste elementene i systemet, som er nettverkskommunikasjon, server, database og android app.

1.4 Hvorfor vi valgte denne oppgaven

Vi som gruppe tok en beslutning på å kontakte Egde Consulting når vi så at deres prosjekt lå ute på Kompetansetorget. Vi syntes at det å være med på å gi Kristiansand et tilbud om bysykler hørtes ut som en interessant ide og vi følte at dette var noe byen trengte. Vi syntes også at

applikasjonsutvikling var noe som hørtes spennende ut, og dette var en fin måte å bli kjent med utviklingen av et sånt prosjekt. Etter å ha vært i dialog med Egde Consulting ble vi enige om at denne oppgaven ikke nødvendigvis trengte å handle spesifikt om bysykler, men at vi kunne ta det som utgangspunkt og gjøre det overførbart til flåtestyring av andre kjøretøy.

Ved å være med på et prosjekt av denne typen ville gi oss erfaring innen applikasjonsutvikling med hjelp av en bedrift, som kan bli nyttig for oss innen karrierevalg senere. Vi ville også kunne bli bedre kjent med ulike typer programmeringsspråk, database og kommunikasjonen som må til mellom disse aspektene.

2. Planlegging

I denne seksjonen redegjør vi for planleggingsfasen. Vi opplevde, som man vanligvis gjør, at planene måtte endres underveis i prosjektet. Disse endringene er beskrevet i analysedelen av oppgaven. Dette er fordi nødvendigheten for endringer oppstod under analysene vi har gjort underveis i prosjektet.

2.1 Epics

Vi startet planleggingsfasen med å definere hovedelementene i systemet vi skulle utvikle. Hos Egde kalles disse elementene for epics, og kan ses på som milepæler i utviklingen. Disse milepælene ble rangert etter hva som var viktigst å prioritere i hensyn til Egde sine forventninger av oppgaven.

Vi delte prosjektet inn i tre epics og definerte de som:

Epic 1: Flåtestyring

Epic 2: Vedlikehold av flåten

Epic 3: Datainnsamling/visualisering/analysering

Deretter laget vi en prosjektplan med tidslinje for de tre epicene fordelt mellom prosjektets start og slutt. Både oppdragsgiver og studentgruppen var innstilt på en hyppig møtefrekvens og ble enige om å sette sprintlengden til 2 uker, og å ha et sprint review meeting med tilhørende demo av systemet fra fullført sprint kombinert med et sprint planning meeting for kommende sprint. Denne møtevirksomheten gjorde at oppdragsgiver kunne følge utviklingen tett og gi verdifull tilbakemelding på retningen til prosjektet, omprioriteringer og assistanse i utviklingen.

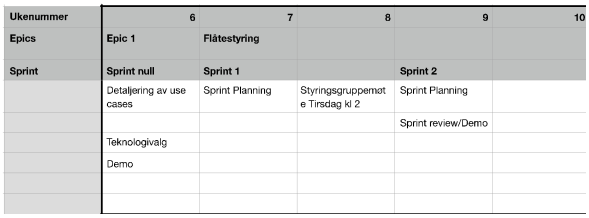
2.2 Prosjektstyringsplanlegging

Vi benyttet scrum som prosjektstyringsrammeverk, da både vi i studentgruppen og Egde som arbeidsgiver hadde kjennskap til og tidligere brukt scrum. Sprintene ble som nevnt satt til to uker, som i utgangspunktet er en ideell lengde. Vi opplevde utfordringer med estimering av vanskelighetsgrad og tidsbruk av de forskjellige oppgavene i de første sprintene. Som resultat av dette måtte vi ta stilling til om scrum var det idéelle styringsverktøyet for vårt prosjekt. Dette har vi gjort rede for i kapittelet om prosjektstyring (kapittel 3).

2.3 Prosjektplanen

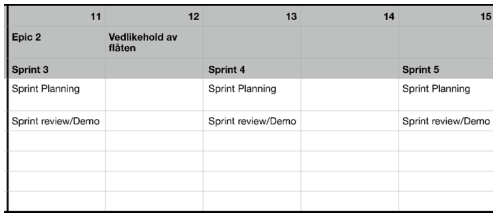
Selv om vi modifiserte prosjektstyringen internt i studentgruppen beholdt vi den oppsatte planen for møter med oppdragsgiver, som hadde forståelse for at noen av oppgavene var svært vanskelig å estimere. Vi forholdt oss også løsere til den definerte planen rundt epics. Etter hvert som prosjektet utviklet seg og planendringene var uungåelig valgte vi å se på epicsene ikke som milepæler som måtte fullføres, men som definisjoner av grunnpilarene systemet er basert på. På den måten kunne vi jobbe med brukerhistorier på tvers av epics, avhengig av hva som måtte utvikles for å få fremgang i prosjektet. Vi innså i ettertid at det var hensiktsmessig, siden utviklingen av flåtestyringssystemet avhenger av sensordata (GPS). Dette står beskrevet i kapittelet om prosjektgjennomføring (kapittel 4).

Vår første versjon av prosjektplanen følger under.



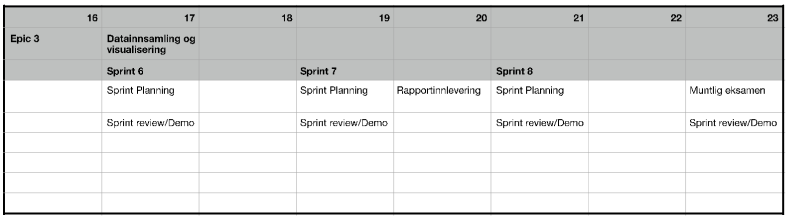
Ukenummer	6	7	8	9	10
Epics	Epic 1	Flåtestyring			
Sprint	Sprint null	Sprint 1		Sprint 2	
	Detaljering av use cases	Sprint Planning	Styringsgruppemøte Tirsdag kl 2	Sprint Planning	
				Sprint review/Demo	
	Teknologivalg				
	Demo				

Figur 1 Prosjektplan del 1



11	12	13	14	15
Epic 2	Vedlikehold av flåten			
Sprint 3		Sprint 4		Sprint 5
Sprint Planning		Sprint Planning		Sprint Planning
Sprint review/Demo		Sprint review/Demo		Sprint review/Demo

Figur 2 Prosjektplan del 2



16	17	18	19	20	21	22	23
Epic 3	Datainnsamling og visualisering						
	Sprint 6		Sprint 7		Sprint 8		
	Sprint Planning		Sprint Planning	Rapportinnlevering	Sprint Planning		Muntlig eksamen
	Sprint review/Demo		Sprint review/Demo		Sprint review/Demo		Sprint review/Demo

Figur 3 Prosjektplan del 3

3. Analyse

Vi er kjent med det faktum at planer og omfang i et prosjekt til stadighet endres, og vårt prosjekt er ingen unntak. Derfor har vi gjennom hele prosjektet gjennomført en kontinuerlig analyse for å evaluere både prosjektstatus og fremgang. Vi vil i den neste seksjonen fortelle om hvordan planen og omfanget av oppgaven har endret seg gjennom prosjektets levetid.

I den innledende fasen definerte oppdragsgiver svært vide rammer for omfanget av prosjektet, og ønsket innspill fra oss i studentgruppen. Dette var utfordrende for oss, siden vi hadde svært begrenset erfaring i slike prosjekter hva angår estimering av kompleksitet og vanskelighetsgrad, tidsbruk for gjennomføring av oppgaver. Etter å ha diskutert omfanget kunne vi snevre inn omfanget til å utvikle et flåtestyringssystem for bysykler, med tilhørende innsamling av sensordata.

Etter en tid med innledende analysearbeid med planlegging av epics (les side 12), brukerhistorier, og estimeringer innså vi etterhvert at oppgaven fortsatt hadde et for stort omfang, og at problemene i de brukerhistoriene vi hadde definert var utfordrende å løse fra et informasjonssystems perspektiv. Mange av brukerhistoriene var tilknyttet den fysiske verden, som låsing av sykler, beholdning av sykler ved de forskjellige parkeringsstativene osv.

Derfor diskuterte vi på nytt med oppdragsgiver prosjektets omfang og kom frem til at det essensielle i prosjektet er det digitale rammeverket for flåtestyringen, register og posisjon til syklene, brukerbasen, og innsamling og lagring av sensordata. Siden det er selve rammeverket for flåtestyringen som er av interesse var det bred enighet om at det var hensiktsmessig å fri oss fra bysykler som et konkret eksempel. Derimot skulle vi fokusere på å lage et generisk, fleksibelt og modulært rammeverk som kan implementeres i en rekke forskjellige sammenhenger.

3.1 Systemets bestanddeler

Med klarhet i det raffinerte omfanget kunne vi spisse prosjektomfanget til følgende elementer og funksjonalitet (En utfyllende liste med teknologier vi har brukt, medfølgende definisjon, bruksområde og begrunnelse for valg kan leses i teknologi-stack, (se side 24).

1. Kjøretøy med sensorenhet: Innsamling av data (sensordata og posisjon)

2. Nettverks- og skykommunikasjon: Overføring av data
3. Back-end: Server med database for mottak, lagring og behandling av data
4. Front-end:
    a. Nettleserbasert grensesnitt for administrasjon
    b. Android-app til sluttbruker/administrasjon/vedlikeholder

3.2 Kravspesifikasjon

Kravspesifikasjonene beskriver hvilke ønsker og krav oppdragsgiver har til systemet vi skal utvikle. Disse spesifikasjonene formuleres før det gjøres innkjøp av maskinvare og utvikling av programvare. Dette er for å sikre tilfredsstillende brukervennlighet, ytelse og funksjonalitet i systemet, både for bruker og administrator.

Basert på prosjektets omfang og bestanddeler kunne oppdragsgiver definere følgende kravspesifikasjoner for systemet. Disse spesifikasjonene har vært grunnlaget for utforming av rikt bilde og funksjonsliste.

1) **Kjernefunksjonalitet**

   Systemet skal være et rammeverk som skal administrere en flåte av kjøretøy som kan lånes/leies/brukes av en spesifikk gruppe brukere eller kunder. Systemet registrer hvem som sjekker ut (låner) hvilket kjøretøy og når det sjekkes inn. Med en flåte av kjøretøy som kan dekke et geografisk område, som for eksempel en hel by, ønskes det at hvert enkelt kjøretøy utstyres med en rekke sensorer som registrerer forskjellige data, som for eksempel temperatur lufttrykk -fuktighet og -forurensning, reisemønster, trafikkflyt osv. Disse dataene skal samles i en database, analyseres og anvendes til samfunnsbedrene tiltak.

2) **Proof of concept**

   Arbeidsgiver vil gjerne ha en bevist bekreftelse for at en gjennomføring av prosjektet med den funksjonaliteten og det bruksområdet som etterspørres faktisk er gjennomførbart, både økonomisk, logistisk og sosialt.

3) **Fungerende kode (funksjonalitet over estetikk)**

   Oppdragsgiver ønsker at vi så fort som mulig, og ved hver demo, skal levere et helhetlig

fungerende system heller enn å bruke mye tid på å finpusse kode.

4) **Høy mobilitet og tilgjengelighet**
   Systemet skal være lett tilgjengelig og brukervennlig både for brukere av systemet og administrasjon og vedlikehold av det digitale systemet og den fysiske bestanddelen som utgjør helheten. Kjøretøy skal være tilgjengelig over nett til alle tider og kommunisere kontinuerlig med serversiden eller samle sammen sensordata og sende pakkevis på tidsintervaller eller ved spesifikke hendelser som ved parkering eller lignende.

5) **Skyløsning**
   Systemet skal lagre og analysere all data i skyen. Dette sikrer tilgjengelighet til alle tider via Internett i tillegg til serverside maskinkraft til analysering av dataene i skyen. Resultatet av de ferdig analyserte dataene blir så fremvist hos brukeren eller administratoren i systemets front end grensesnitt.

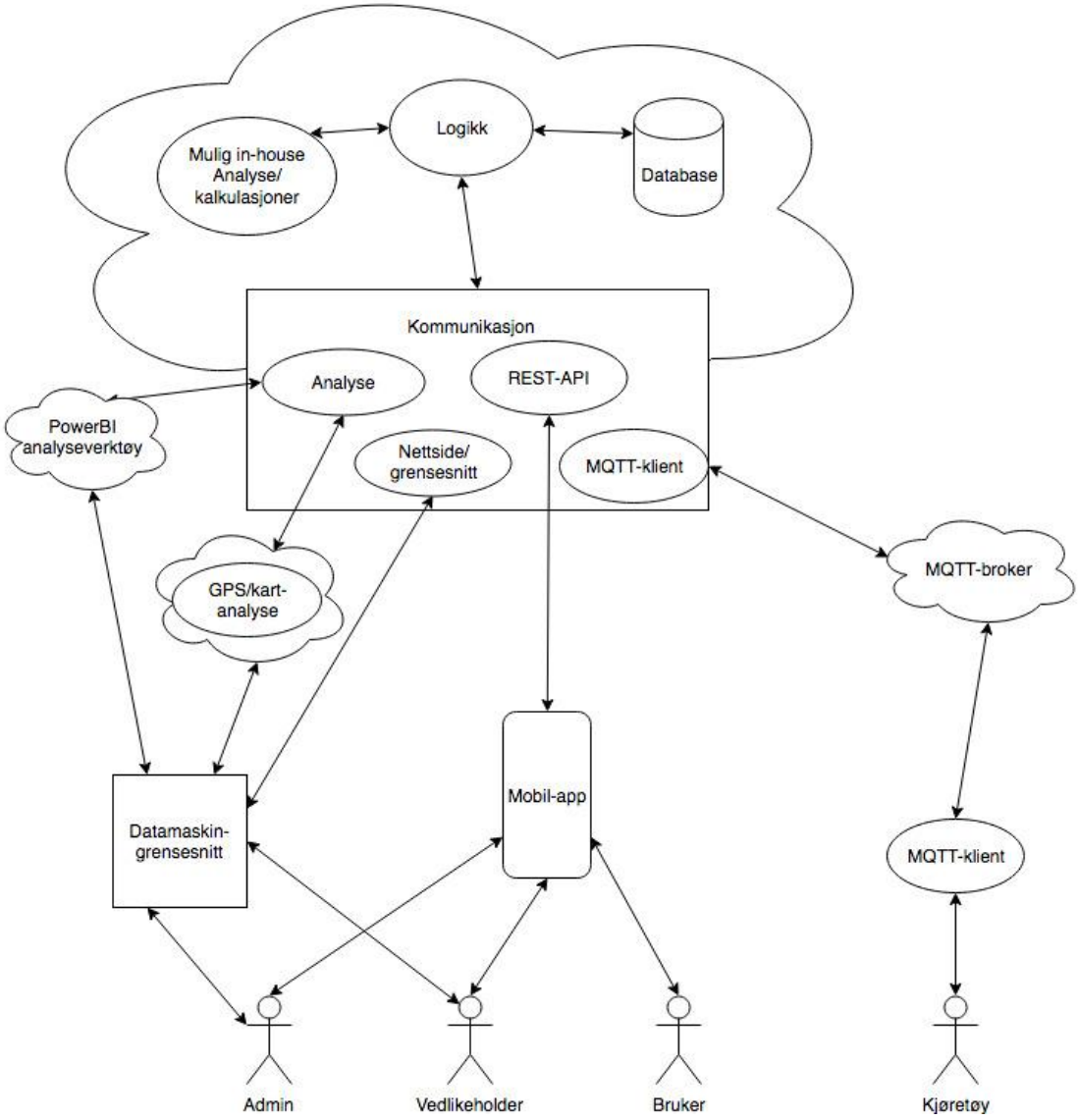
6) **Fleksibilitet/gjenbrukbarhet**
   Systemet som rammeverk skal være lett å tilpasse forskjellige bruksområder. Det kan være styring av andre typer fysiske aktivum som booking av møte/grupperom, bygninger, båter, droner og andre enheter som kan knyttes til internett.

7) **Skalerbarhet**
   Systemet skal enkelt kunne takle å øke antall enheter som knyttes til og leverer data. Systemet testes med et fåtall enheter, men må i en fullt implementert utgave kunne takle tusenvis av tilkoblede enheter.

3.3 Rikt bilde

Et rikt bilde er en grafisk fremstilling av systemet som helhet, med alle dets bestanddeler representert og piler som viser informasjonsflyten mellom de forskjellige delene.

Den første utgaven av det rike bilde ble laget under den innledende analysefasen, før vi hadde valgt kommunikasjons- og analyseteknologier. Versjon to som følger under er en oppdatert utgave som er mer detaljert. Vi har utbrodert modellen med de kommunikasjons- og dataanalyseteknologiene vi har valgt å bruke.



Figur 4 Rikt bilde

3.3.1 Beskrivelse av rikt bilde

Vårt system består av tre hovedkomponenter; en server back-end i skyen, en front-end android applikasjon og en IoT sensor med støtte for lettvekts protokoll MQTT. Det er fire aktører i systemet: admin, vedlikeholder, bruker og kjøretøy.

Backend:

- Database til lagring av alle data - MySQL
- Logikk for informasjonsflyt på serversiden - PHP (Rest) / Java (MQTT)
- En eventuell intern analysemotor (Ikke implementert per 15. mai 2018)
- Kommunikasjonssenter med dedikerte moduler for hver teknologi

- Analyse: Dedikert modul for å sende data til eksterne analyseprogrammer som PowerBI og GPS visualiseringsverktøy - Microsoft PowerBI / Google maps (ikke implementert

Front-end:

- Android applikasjon: Delvis implementert per. 15. mai 2018, med mulighet for innlogging, registrering og visning av telefons lokasjon, samt posisjon statiske kjøretøy. Arbeides med dynamisk visning av kjøretøy fra database på nett.
- Webklient: Modul for fremvisning og administrasjon av all ønskelig data for admin og vedlikehold. (Ikke implementert per 15. mai 2018)

Kommunikasjonsmodul for IoT kort bestående av:

- *MQTT-klient:* To-veis kommunikasjon via broker til alle kjøretøy/enheter som samler informasjon - utviklet klient selv - basert på MQTT API
- MQTT-broker: Bindeleddet mellom MQTT-klientene som kjører på server og i kjøretøyene. - CloudMQTT (www.cloudmqtt.com)
- MQTT-klient i alle kjøretøy. Denne modulen sender posisjon og status på kjøretøyet og sensordata til serveren. - Utviklet selv - basert på MQTT API

3.4 Funksjonsliste

En funksjonsliste er en skjematisk oversikt over de viktigste funksjonene i et system. Vi har ut i fra brukerhistoriene og oppdragsgivers kravspesifikasjon satt opp noen av de viktigste funksjonene for at systemet skal fungere optimalt. Funksjonene i listen er kategorisert etter entiteter/aktører og dereter sortert sortert kronologisk. Det vil si at det første som skjer for en ny bruker er å registrere seg i systemet. Deretter har vi listet "slette bruker" som neste funksjon. På tross av at det kronologisk sett er det siste som skjer for en bruker har vi listet de sammen siden de omhandler samme aktør i systemet og utfører en handling relatert til den forrige, enten en tilsvarende, komplementerende, motsatt handling.

Funksjon i systemet	Kompleksitet	Type/mål	Involverte moduler
Registrere bruker	Lav	Registrering	Admin, system, bruker
Slette bruker	Lav	Oppdatering	Admin, system, bruker
Logge inn	Høy	Avlesning	Admin/bruker, system
Logge ut	Lav	oppdatering	Admin/bruker, system
Booke kjøretøy	Medium	Registrering	System, bruker, kjøretøy
Slette booking	Lav	Oppdatering	Bruker, system
Levere kjøretøy	Lav	Oppdatering	Bruker, system, kjøretøy
Sjekke lokasjon på ett/flere kjøretøy	Medium	Avlesning	Admin, system, database
Vis reisehistorikk på flere kjøretøy	Medium	Avlesning	Admin, system
Registrere sensordata og lagre lokalt i sensorenhet	Medium	Registrering	Sensorenhet
Slette lokalt minne i sensorenhet	Lav	Oppdatering	Sensorenhet
Sende sensordata til database	Medium	Nettverkskommunikasjon	Sensorenhet, nettverkskommunikasjon, Java
Registere sensordata i database	Lav	Registrering	Java, system, database
Hente data fra databasen	Medium	Avlesning	Database, java
Analysere data	Høy	Beregning	Admin system, database, analyseverktøy
Starte/stoppe live feed av sensordata	Høy	Oppdatering	Sensorenhet, nettverkskommunikasjon, admin
registrere/slette booking	Lav	Registrering	Admin, kjøretøy, system, database

Figur 5 Funksjonsliste

3.5 Sikkerhet

Hverken oppdragsgiver eller studenter har satt sikkerhet som et høyt prioritert tema på agendaen siden vi har valgt å fokusere på bruksområde og utvikling av kjernefunksjonalitet og rammeverket for systemet. Vi er utvilsomt klar over viktigheten av sikkerhet i et slikt system. Det er et aspekt som må implementeres før systemet kan tas i bruk i stor skala.

Det kan argumenteres at sikkerhetsfunksjonalitet kunne og burde utvikles og implementeres parallelt med de kjernefunksjonaliteten. Dette er noe vi har reflektert over og vurdert både de positive og negative konsekvensene av. På en side er det hensiktsmessig å implementere

sikkerhet underveis, siden man da får en sikker funksjonalitet i systemet. På den andre siden vil utviklingen av kjernefunksjonaliteten gå saktere. I tillegg vil sikkerhetsimplementasjonen representere enda et element hvor det kan oppstå bugs og problemer, som igjen vil lede til mer tid til bruk på feilsøking. Ved alvorlige uforutsette feil i et slikt tilfelle kan det koste mer tid å feilsøke et helt system heller enn å implementere sikkerhet på en velfungerende kodebase. Til fremtidige prosjekter vil vi ta med oss denne lærdommen når vi i planleggingsfasen vurderer sikkerhetsimplementasjon.

3.6 Brukerhistorier

Brukerhistorier er en kort tekst som fra enten produkteier/oppdragsgiver eller en spesifikk aktørs synspunkt. Denne teksten beskriver en ønsket funksjon i systemet. Historien brukes som grunnlag av utviklerteamet som en oppskrift til å utvikle ny funksjonalitet i systemet.

En stor andel av våre brukerhistorier viste seg å være unyttige etterhvert som omfanget for prosjektet endret seg. Likevel var det mer enn nok arbeid i de brukerhistoriene som gjensto. Et eksempel som på en brukerhistorie som i skrivende stund ikke er 100% fullført følger med en kort beskrivelse under inneværende avsnitt. En komplett liste av våre brukerhistorier ligger som vedlegg 7**.**

Brukerhistorie 13 – Protokoll for dataoverføring	
Brukerhistorie	Som **SYSTEM** ønsker jeg en lettvekts protokoll for overføring av alle sensordata fra sykkelModulen til databasen, for å minimere datastørrelsen som sendes og dermed spare mest mulig batteri i enheten
Beskrivelse	En overføringsprotokoll som er så slank som mulig og overfører kun de nødvendige dataene i et forhåndsdefinert format som leses og forstås på serversiden.
Hvorfor/viktighet	Vi trenger at data overføres fra sykkelen til databasen. Samtidig trenger vi å spare mest mulig batteri. Dermed ønsker vi å skrive en egen slank protokoll som minimerer datamengden og dermed sparer strøm på sykkelenheten
MoSCoW	Must have
PP-score (Fibonacci)	
Akseptansekriterie	Sensordata i sykkelmodulen registreres og lagres i et forhåndsdefinert format. Dette formatet sendes via NB til server som leser tolker og

	registrerer data i databasen. Denne dataen kan deretter fremvises i AdminGUI.
Involverte elementer	SykkelGPS, sykkelmodul, kommunikasjon, database, adminGUI
Testoppskrift	1. Generer data/testdata fra ArduinoNB-enhet 2. Data skrives i protokollen etter predef format 3. Data overføres til server og database 4. Data hentes til og leses av adminGUI

Brukerhistorie: Den faktiske historien om ønsket funksjonalitet

Hvorfor/viktighet: Her forklares det hvorfor funksjonen er ønsket eller viktig

MoSCoW: En vurdering av hvor viktig funksjonen er

PP-score: Planning poker-estimat for fullføring av brukerhistorien. Dette feltet er tomt i de fleste brukerhistoriene våre. Dette er fordi estimering av omfanget foregikk etter alle brukerhistoriene ble skrevet. Deretter ble tidsestimatet ført direkte inn i vår scrumdesk; Jira.

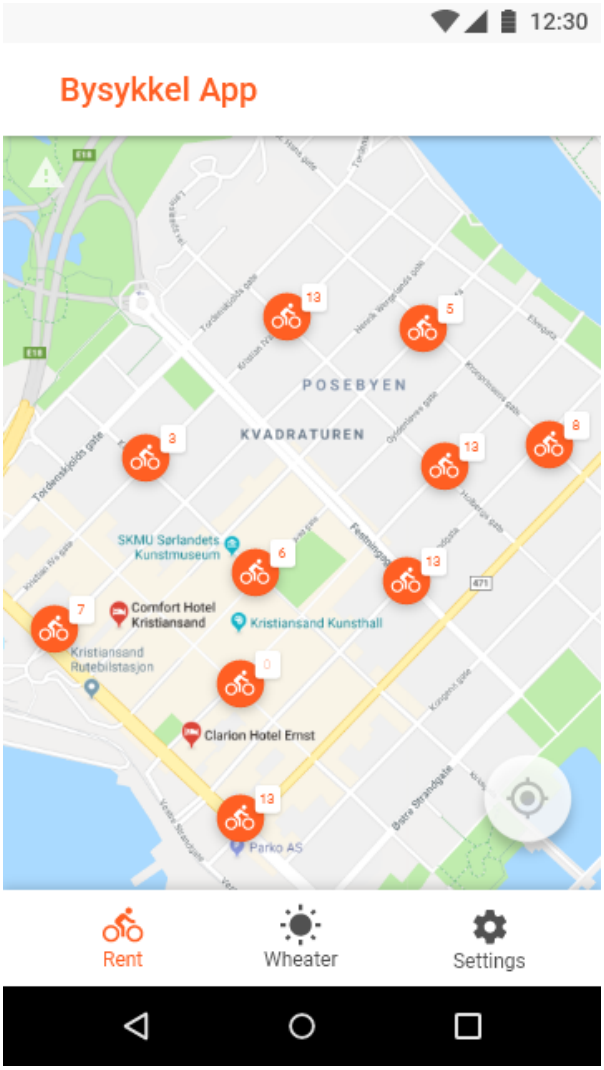
Akseptansekriterier: Hvilke hendelser skal forekomme når ønsket funksjon utføres.

Involverte elementer: Hvilke elementer og/eller aktører som er involvert i hendelsen

Testoppskrift: En oppskrift på hvordan kan vi som utviklere kan forsikre oss om at hendelsen forekommer som planlagt.

3.7 Visuelt design og utvikling av prototype

For å se full oversikt over skjermbilder fra prototypen se vedlegg 2



Figur 6 Applikasjonsprototype

Før vi begynte utviklingen av Android applikasjonen startet vi å lage en prototype for det originale bysykkelkonseptet slik at alle fikk et visuelt mål å gå etter under implementeringen. Prototypen er basert på Google's Material Design som er et design rammeverk som er brukt for de fleste moderne Android applikasjoner. Rammeverket tilbyr informasjon om velfungerende grensesnitt elementer samt tips om hvordan navigering og grensesnitt elementer bør settes opp og kombineres for å skape et vakkert og velfungerende grensesnitt som også er lett gjenkjennelig for nye brukere.

For å lage prototypen brukte vi Adobe Xd CC 2018 som er en relativt ny applikasjon fra Adobe som er skreddersydd for å lage klikkbare prototyper som gjør det raskt å iterere og teste brukbarheten og flyten rundt navigeringen av applikasjonen. Adobe Xd inneholder allerede noen Material Design UI elementer som gjorde det raskt å iterere og tilpasse til vårt eget bruksområde. Vi brukte også ferdiglagde UI elementer fra materdesignkit.com og uxpin.com, samt egenlagde elementer laget i Adobe Photoshop / Adobe Illustrator. Ikoner ble hentet fra material.io/icons.

Om vi var usikre rundt hvordan vi skulle løse flyten og konstruksjonen for noen av grensesnittene ble det laget enkle skisser på papir for å raskere kunne teste forskjellige ideer før vi brukte tid på å lage en digital versjon med høy fidelitet.

I tråd med at størrelser på telefoner de siste årene har blitt større og større gikk for en «bottom navigation bar» som navigasjonsmønster for å bytte mellom hovedviews i grensesnittet. Ved å plassere hovedfunksjoner på bunnen av grensesnittet blir de lettere tilgjengelig mer synlige, og enklere å bruke med en hånd. Mange Android applikasjoner bruker i dag en «hamburger menu»

eller en side-meny for å bytte mellom views, men dette gjør funksjoner vanskeligere å oppdage og nå for bruker. En hamburgermeny kan være nyttig for visse applikasjoner med mange funksjoner, men ut i fra brukerhistoriene våres så vi på «bottom navigation bar» som et bedre valg.

Prototypen var veldig nyttig internt for å teste hvordan brukerhistorier kunne visualiseres i en reell applikasjon uten å bruke mye tid på kode på forhånd.
Prototypen ble presentert for Egde og vi fikk gode tilbakemeldinger, samt tilbakemeldinger om justeringer vi kunne gjøre for å forbedre den. En av tilbakemeldingene var å gjøre om språket til norsk, oppstartsflyt for betalingsinformasjon, og gode kontraster for å sørge for at mennesker med synshemninger også kan lett bruke appen.

Vi anerkjenner at prototypen kan forbedres, spesielt når det kommer til flere funksjoner og flyt for betaling og leie av sykkel. Men vi sa oss fornøyde som et grunnlag for å starte utviklingen. Da målet for prosjektet drastisk endret seg over tid så bestemte vi oss for å ikke bruke for mye tid på flyt for spesifikke grensesnitt for sykkelutleie. Bunnnavigasjonen med «weather» ble lagt til som en placeholder for andre funksjoner og var tenkt til å kunne brukes til å visualisere data fra sensor.
«Settings» med bruker/app-innstillinger kan for eksempel flyttes til en «overflow button» på toppen av grensesnittet om applikasjonen trenger plass til andre viktigere hovedfunksjoner.

3.8 MoSCoW

For å finne ut hvilke funksjoner som var viktigst å prioritere, tok vi i bruk MoSCoW-metoden. Metoden går ut på å dele opp funksjonene etter hva som må være med «Must have», hva som burde være med «Should Have», og hva som kan være med «Could Have» i systemet. Dette hjalp oss med å finne ut hva vi måtte prioritere først og hva som vi etterhvert kan bygge videre på ved en senere utvikling av systemet.

3.9 Planning Poker

Planning poker er en konsensusbasert aktivitet som er mye brukt i agil utvikling for å estimere vanskelighetsgraden eller tidsbruk på forskjellige mål eller oppgaver i prosjektet. Hvert teammedlem har et sett kort med verdiene 1, 2, 3, 5, 8, 13, 20, 40 og 100. Produkteier presenterer så en brukerhistorie eller en ønsket funksjon som skal implementeres. Deretter

velger hvert teammedlem et av kortene basert på personlig anslått vanskelighetsgrad, og legger frem valgt kort med tallverdien ned. Etter dette snus alle kortene samtidig. Dersom alle er enige om estimatet blir det satt. Ved forskjellige verdier på kortene diskuteres det til enighet om et estimat. Ved at alle legger ned kort uten å vite andres estimat unngår man kognitiv bias i form av ankring basert på det første anslaget i en diskusjon. Med andre ord får alle teammedlemmene presentert sitt oppriktige estimat, upåvirket av de andres meninger.

I vårt team benyttet vi oss av en nettbasert versjon av aktiviteten på nettsiden www.planningpoker.com [1] til å estimere antall story points per oppgave. Story points oversatte vi deretter til timer anslått per oppgave å utføre.

3.10 Teknologistack

I et hav av forskjellige teknologiske alternativer til bruk har vi måtte ta visse valg i vårt prosjekt. I de neste avsnitt følger en liste med de forskjellige teknologier vi har brukt sammen med en beskrivelse av teknologien og bruksområdet. Deretter kommer en kort beskrivelse av teknologien sammen med redegjørelse av hvorfor vi har valgt nevnte teknologi.

3.10.1 Maskinvare:

PC/Mac til utvikling av kildekode
Mikroprosessor: Elecrow Crowduino M0-SD +
Nettverksmodul: Sodaq NB-IoT sheild
Server Hosting: Digital Ocean Virtual Private server

3.10.2 Programvare:

VCS: Bitbucket
Scrumverktøy: Jira
Kommunikasjon internt: Facebook/Messenger
Kommunikasjon oppdragsgiver/veileder: Skype
Administrativ dokumentlagring: Office 365 Sharepoint
Rapportskriving: Microsoft Word online
IDE: Java/android utvikling: IntelliJ/Android Studio
Web service stack: LAMP (Linux / Apache / MySQL / PHP)
VPS konsoll tilgang: Putty

Filoverføring til VPS: FileZilla

Rammeverk for kommunikasjon mellom webservice og Android: REST API

Biblioteker Android: Android Volley library

Mikroprosessor/sensorinnsamling: Arduino IDE

Database: MySQL med PHP Admin

Nettverkskommunikasjon: Eclipse Paho MQTT

Design / mockup / prototyping: Adobe XD og Adobe Photoshop CC 2018

3.11 Valg av teknologi

3.11.1 IntelliJ - IDE - Integrated development environment (Utviklingsmiljø)

Vi valgte å gå for JetBrains IntelliJ som utviklingsmiljø primært på grunn av programmet Android Studio, en derivasjon av IntelliJ som er skreddersydd for utvikling av Android-apper. Enkelte av gruppens medlemmer har også tidligere erfaring med IntelliJ og foretrekker det foran Eclipse og NetBeans.

3.11.2 Java - Programmeringsspråk

Java, et av de mest utbredte programmeringsspråk per dags dato. Vi valgte å gå for Java som programmeringsspråk av flere årsaker. Først og fremst er det fortsatt et svært populært og mye brukt språk som Egde selv bruker på mange av sine prosjekter. Det var derfor ønskelig fra oppdragsgivers side å benytte Java. I tillegg har vi utviklet en mobilapp som skal brukes i systemet. Siden flere i bachelorgruppen eier og bruker Android smarttelefon var det et naturlig valg å bruke Java, siden det brukes til alt av Android mobilenheter.

3.11.3 Arduino / Mobil sensorenhet - Programmerings

Mikroprosessor og IoT nettverkskort fikk vi utlevert fra Egde. Crowduino (som er en Arduino klone) fungerer som en vanlig Arduino mikroprosessor og kan dermed programmeres med IDE-programmet (med samme navn) Arduino. Arduino som programmeringsspråk er i sin hovedsak C/C++ pakket inn i Arduinos egen syntaks. Dette er gjort for å senke terskelen for å kunne forstå kodespråk og dermed gjøre programmering av Arduino enklere og mer appellerende for folk uten nevneverdig teknologisk innsikt/erfaring/forkunnskaper/forutsetninger.

3.11.4 MySQL - Database

Når vi skulle velge hvilken database vi skulle bruke gjorde mye undersøkelser for å finne ut hva som var den beste løsningen for oss. I utgangspunktet så vi på serverløs backend gjennom Googles Realtime Firebase som lagrer data utstrukturert i JSON schemaer vi noSQL. Et pluss var at mye funksjonalitet kunne utføres med relativt lite kode pga tett integrasjon med Android Studio, det er skalerbart, samtidig som det støtter real time oppdateringer av data, noe som virket som at det kunne passe bra for vårt originale konsept for bysykler.

En av bakdelene med Firebase var at vi ikke hadde tidligere erfaringer med noSQL, samtidig at det er begrensninger for hvor mye serverlogikk man kan implementere og kompleksiteten på quieries man kan kjøre på data. Dette gjorde at vi anså Firebase som mindre modulært, hvor mye logikken for datahåndtering måtte implementeres på hver enkelt plattform som potensielt skulle støttes. Vi var også litt usikre hvordan det ville håndtere kommunikasjon med iOT enheter.

Når scopet og konseptet for systemet endret seg, og det var et fokus om modularitet og historisk data-analyse, endte vi til slutt med å gå for en MySQL database hostet på en virtuell private, for å forsikre oss om at vi hadde større frihet til å definere serverlogikk ut i fra hva vi trengte. Ut i fra undersøkelser vi gjorde er en strukturert MySQL database med til å håndtere historisk data. Vi hadde og erfaringer med MySQL fra tidligere undervisning, noe som potensielt betydde at vi kunne bruke mindre tid på å sette oss inn teknologi vi hadde mindre kjennskap til.

3.11.4 MQTT - Nettverkskommunikasjon

MQTT som nettverksteknologi ble foreslått først av Trond Kvarenes, en av våre oppdragsgivere. MQTT er en lettvekts meldingsprotokoll som baserer seg på en tre-ledds modell bestående av sender, terminal/mellomledd(broker) og mottaker. Meldingen som sendes består av en overskrift/emne(Topic) som forteller mellomleddet hvor meldingen skal videresendes, og selve meldingen som er fritekst og kan inneholde hva som helst. Denne modellen kan sammenlignes med postsystemet vi har, hvor en pakke sendes fra senderen til en terminal som sorterer og videresender pakken til korrekt mottaker.

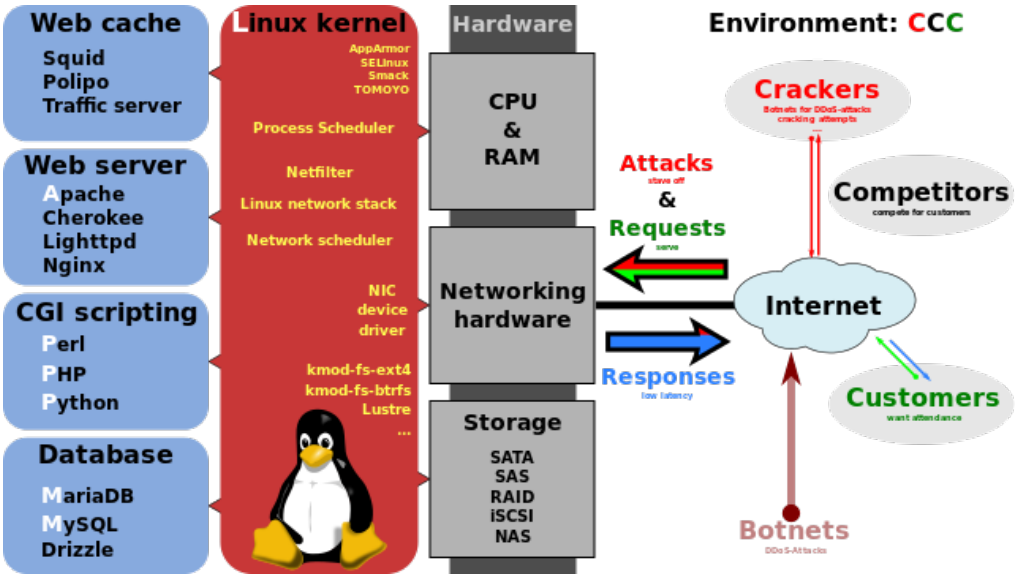
Vi har valgt denne teknologien fordi den er svært lettvekt, med minimalt av unødvendig informasjon og metadata (sammenlignet for eksempel med xml eller json). Vi kan formatere meldingen så kompakt som vi ønsker å skrive egen kode for å tolke meldingene på serversiden.

Infrastrukturen er også svært robust og skalerbar. Det er svært fordelaktig dersom systemet skal implementeres i en kjøretøypark bestående av tusenvis av kjøretøy.

3.11.5 Webserver / Lamp Software Bundle / Rest API

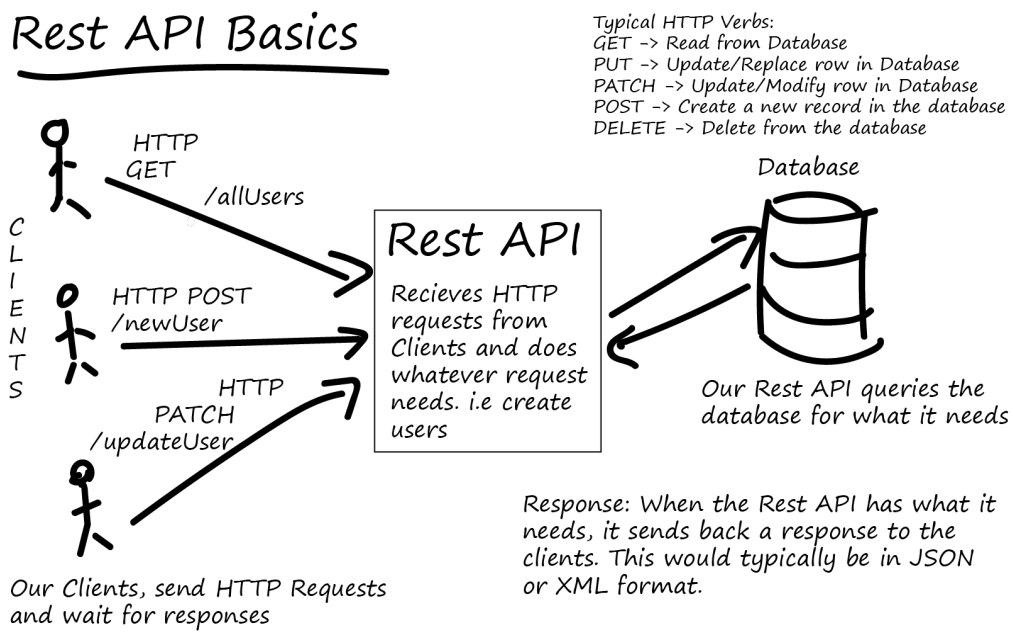
Vi har satt opp en webserver til databasen og backend logikk hos www.digitalocean.com. De er en seriøs tilbyder av virtuelle maskiner og datalagring online. Gjennom GitHub student pack tilbyr de $50 i gratis credits til alle studenter, dermed var de et godt valg for oss til dette prosjektet.

Dropleten ble konfigurert med en instans av Ubunutu som ble konfigurert med LAMP stack, som en komplett software bundle som inneholder Linux, Apache server, MySql og PHP.



Figur 7 LAMP Struktur

Oversikt over LAMP struktur [2]



Figur 8 Rest API Funksjon og struktur

Enkel oversikt over Rest API funksjon og struktur [3]

3.11.6 VCS: BitBucket

VCS - Version Control System er et online lagrings og administrering system for kildekode. Vi valgte BitBucket foran GitHub (som kanskje er et av de mest populære tilbyderne). Noe av grunnlaget for dette var BitBucket tilbyr private repositories gratis, i tillegg til at Egde også bruker dette internt hos seg. BitBucket hadde også intergrasjon mot d vårt scrumboard; Jira som var et pluss. Ved implementering av nye funksjoner branchet vi ut i nye "development branches" med navn på funksjoner som ble arbeidet med. Når funksjonene var ansett som ferdigstilt og fungerende, ble disse merget over til master branchen.

3.11.7 Jira

Egde kunne gi oss tilgang til deres scrumverktøy Jira. De fleste slike programmer er enten gratis og dårlige eller litt dyrere enn vi kunne foretrekke. Derfor takket vi gjerne ja til å kunne bruke Egdes in house scrumverktøy.

3.11.5.3 Facebook/Messenger/Skype

Internt i gruppen har vi brukt Facebook (lukket gruppe til linker og generell kommunikasjon) og (Facebook) Messenger til daglig kommunikasjon og planlegging. Fordelen med å ha en gruppe er at det er lettere å skille forskjellige diskusjoner fra hverandre og å finne frem til statisk

informasjon som lenker og dokumenter. Likevel har enkelheten med Messenger vunnet frem og blitt hovedmediet for kommunikasjon. Det meste av den dagligdagse kommunikasjonen er "ferskvare", altså relevant kun der og da. Som avtaling av møtetidspunkt, booking av grupperom osv. Det meste av informasjonen man trenger finne tilbake til ble lagret enten i Facebookgruppen, på Canvas, Sharepoint eller Bitbucket.

I vår møtevirksomhet med oppdragsgiver har vi ved flere anledninger benyttet oss av Skype. Dette fordi Egde har kontorer i både Kristiansand og Grimstad, og våre tre kontaktpersoner er fordelt på de forskjellige kontorene. Det er også fordelsmessig for oss å kunne bruke Skype fra universitetet istedenfor å måtte reise til Grimstad for korte oppdateringsmøter.

4. Prosjektstyring

I denne seksjonen tar vi for oss prosjektstyring og redegjør for vår bruk av agil utviklingsmetode, prosjektstyringsverktøy, planlegging og sentrale elementer rundt prosjektstyringen. Vi tar utgangspunkt i at leseren har kjennskap til de forskjellige utviklingsmetodikkene som er omtalt i dette kapittelet og går derfor ikke inn for å forklare de forskjellige metodene.

4.1 Agil utvikling

Vi har i dette prosjektet jobbet agilt siden det er fleksibelt og modulært i forhold til for eksempel fossefallsmetoden. Det er fordelaktig siden vårt prosjekt går over en relativt kort periode. Den agile metoden åpner for tilpasning til uforutsett problemer og det gjør at vi kontinuerlig kan vurdere og følge beste praksis underveis i prosjektet.

4.2 Scrum

I agil programvareutvikling er det mest brukte rammeverket scrum. Det var derfor et naturlig valg av utviklingsmetodikk for oss. I tillegg har oppdragsgiver også scrum som foretrukket utviklingsrammeverk.

4.3 Utfordring med estimering og prosjektstyring

Etter hvert som sprintene gikk innså vi utfordringene med å estimere kompleksitet og tidsbruk på mange av våre brukerhistorier og oppgaver. Uten forkunnskaper om teknologiene vi skulle ta i bruk i dette prosjektet krevde det en god del lenger til å fullføre de oppgavene vi hadde

valgt ut per sprint. Dette resulterte i at vi havnet langt på etterskudd i forhold til den opprinnelige planen. Siden scrum-metoden i stor grad baserer seg på estimering og fullføring av et gitt antall oppgaver i et forhåndsdefinert tidsrom var det svært utfordrende å bruke rammeverket effektivt, med tanke på problemene med estimering. Vi fikk heller ikke tatt i bruk det fulle potensialet i prosjektstyringsverktøyet Jira. Etter flere sprinter på konstant etterskudd, og med mye tid og krefter tappet av fortvilelsen rundt scrum begynte vi å vurdere andre styringsmetoder som kunne passe bedre for oss. Vi hadde i starten av prosjektet diskutert ulike rammeverk for prosjektstyring. Etter hvert kom vi frem til at kanban var mer passende for vårt prosjekt.

4.4 Kanban vs scrum

Kanban som også er en agil utviklingsmetodikk har visse likheter med scrum. Derimot er det kanskje forskjellene på scrum og kanban som er viktigst.

Den kanskje største forskjellen er at scrum er en sprintbasert metodikk med flere hvor en sekvens av, planleggingsmøter, oppgavegjennomføring og evalueringsmøter som gjentas, med en ny backlog for hver iterasjon og et strengt regelverk for hvordan ting skal gjøres. Kanban er på den andre siden en fri metodikk med en kontinuerlig arbeidsflyt, én backlog som man kontinuerlig henter nye oppgaver fra. Arbeidslasten reguleres ved at man aldri har flere enn fem aktive oppgaver til ethvert tidspunkt. Det er heller ingen begrensninger på tidsbruk per oppgave eller planendringer underveis i arbeidsflyten.

Det kan argumenteres for at det er hensiktsmessig for oss som uerfarne utviklere å ha et rigid rammeverk for utvikling med klare retningslinjer å forholde seg til, for at ikke arbeidet og strukturen skal flyte ut i ingenting. Likevel mener vi at det på grunnlag av prosjektets mål og omfang er bedre for oss å jobbe med et rammeverk der det byråkratiske aspektet er redusert til et minimum. Det maksimerer dermed tiden vi kan disponere til å løse de faktiske problemene vi står overfor. Denne oppgaven baserer seg på at en fungerende infrastruktur mellom database, javakode og nettverkskommunikasjon. Det er klart definerte oppgaver som må løses før den resterende funksjonaliteten kan bygges på toppen av denne infrastrukturen. Vi så det som fordelaktig å kunne dedikere all vår tid til å implementere en fungerende base, heller enn å måtte bruke tid på planendring og dokumentasjon av endringene.

4.5 Bruk av kanban

Etter å ha konvertert til kanban gikk vi i stor grad vekk fra å bruke Jira som styringsverktøy. Istedenfor tok vi utgangspunkt i funksjonslisten som definerte de viktigste funksjonene i systemet. Siden vi er et relativt lite team kunne vi med enkelhet holde hverandre oppdatert etterhvert som vi fullførte forskjellige oppgaver.

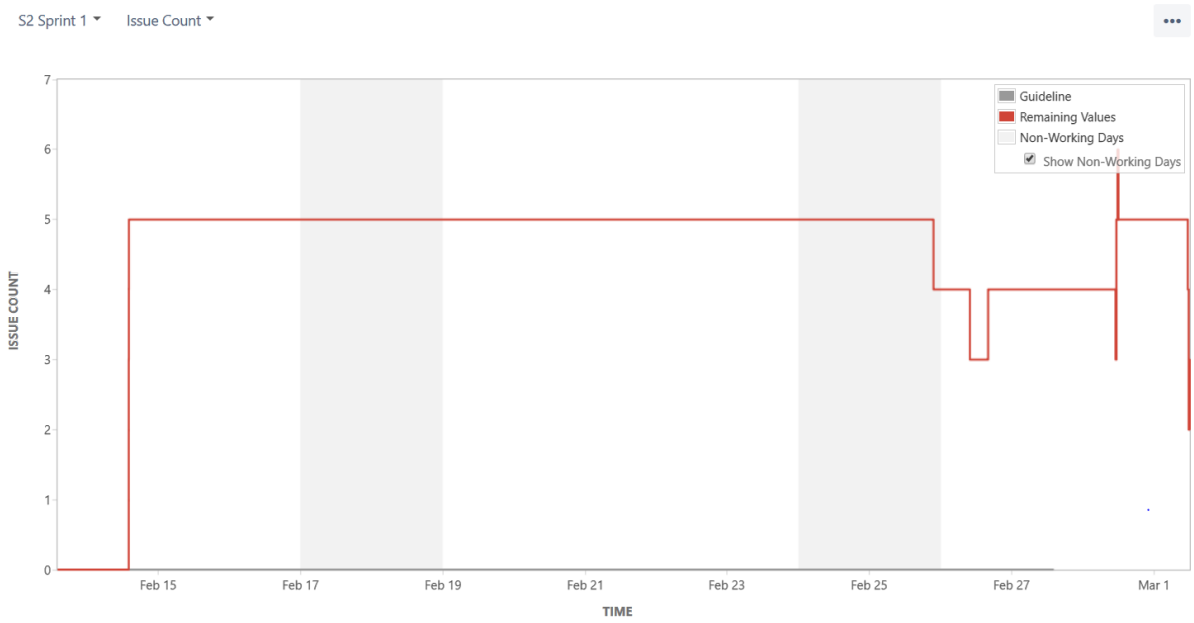
Selv om vi gikk vekk fra scrum som styringsrammeverk beholdt vi likevel planen for møter og demoer med oppdragsgiver. På den måten kunne vi effektivisere fremgangen i utviklingen av systemet i tillegg til å avholde demonstrasjoner og få tilbakemeldinger på utført arbeid og diskutere fremtidige planer og nødvendige endringer.

4.6 Hvordan har vi styrt prosjektet

Vi har gjennom hele prosjektet styrt fremdriften basert på flere elementer. Møtefrekvens og tidspunkt er bestemt av den opprinnelige prosjektplanen, og de overordnede målene for utvikling har alltid vært basert i kravspesifikasjonene fra Egde, og brukerhistoriene vi har skrevet.

Da vi forholdt oss til scrum som rammeverk styrte vi sprintplanleggingen via product backlogen og prioriterte de viktigste funksjonen i dialog med oppdragsgiver. Etter som vi byttet til Kanban som rammeverk gikk vi vekk fra brukerhistorier og fokuserte på å kunne ferdigstille fungerende funksjonalitet basert på funksjonslisten.

4.7 Burndown chart



Figur 9 Burndown chart

Her vises et utdrag fra burndown chart fra sprint 1 basert på issues som er lagt til og markert som ferdig. I denne sprinten tok vi ikke i bruk tidsestimater. Noe av grunnen til dette er at vi ikke hadde god nok kjennskap til Jira og hvordan Jira håndterer estimering og timelogging i forhold til hva vi er vant med fra tidligere verktøy som f.eks Scrumdesk. Jira bruker story points, i stedet for faktiske timer som gjorde det vrient å estimere og dokumentere tidsbruk. I senere tid fant vi mulighet for å logge timer, men det ser ikke ut til at det er mulighet for å generere grafer som viser forhold mellom loggførte timer og storypoints på en god måte

De første dagene av sprinten viser en flat framgang på grafen. Dette kan skyldes flere mulige faktorer.

- Vi hadde ikke satt oss godt nok inn i Jira på forhånd
- Oppgaver som var ferdige ble ikke merket som ferdig når de faktisk var
- Oppgaver tok flere dager å gjennomføre
- Vi jobbet ikke jevnt nok med oppgaver
- Vi overestimerte scopet av sprinten

Mot slutten av sprinten ser man antall oppgaver begynner å synke litt, for å så stige igjen, før å så synke ved sprintslutt. Vi prøvde å bruke story points i følgende sprint, men det ble ofte estimert underveis siden prosjektets fokus endret seg mye noe, som igjen førte til dårlige burndown grafer. Dette strider også i mot retningslinjene til Scrum-rammeverket. Backlogen

økte siden prosjektet endret seg og viktige stories hang igjen videre fra de forrige sprinten. I de første sprintene brukte vi relativt store stories som vi ikke fikk gjort ferdig. Vi prøvde derfor å legge til mindre spesifikke subtasks til hver story, men problemet vi møtte på da var at det ikke var mulighet for å estimere storypoints til hver subtasks. Dette førte til at selv om mange subtasks var utført, så fikk de ikke uttelling på burndown grafen. Vi valgte på grunnlag av dette å gå over til å bruke Jira scrumboardet som et Kanban board for oversikt over oppgaver i arbeidet videre.

5. Prosjektgjennomføring (Implementering)

5.1 Planlegging

Siden bacheloroppgaven løper over kun ett semester spiller de tidsmessige restriksjonene inn på hva og hvor mye det er mulig å få utviklet og implementert. Vi ble derfor enige med oppdragsgiver om å fokusere på å lage fungerende kode på ønskede funksjoner, og da gå videre til å utvikle neste del heller enn å perfeksjonere hver enkelt bit av systemet.
Ved å arbeide etter denne metodikken får vi så fort som mulig en fungerende helhet som vi da kan forbedre bit for bit etter at systemet vi bygger er i tråd med oppdragsgivers kravspesifikasjoner.

5.2 Styringsmøter

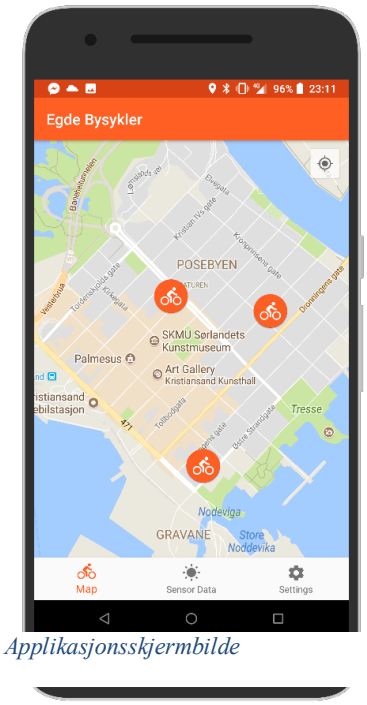
I løpet av denne perioden har vi hatt to styringsmøter i Grimstad med veileder og Egde Consulting. På disse møtene gikk vi gjennom prosjektstatus, utfordringer og hva vi måtte fokusere på videre. Vi har avklart med veileder at siden vi har hatt møter kontinuerlig med Egde gjennom hele prosjektet, så var det nok med to styringsmøter. Referat fra disse styringsmøtene finnes som vedlegg 8.

5.3 Utvikling av database

Til nå har vi arbeidet med separerte databaser for Android applikasjonen og sensor. Dette ble gjort for enkelhet og testings skyld slik at vi kunne prøve å hente ut og lagre data, når de forskjellige modulene ble utviklet separert over samme tidsperiode. Vi har laget et utkast av et EER diagram for det originale bysykkelkonseptet som du kan se i vedlegg 3. Dette trenger visse justeringer for å tilpasses det nye flåtestyringskonseptet. Målet fram mot presentasjonen er å

samle tabeller relatert til brukere og sensordata til en felles database med nødvendige relasjoner, samt Rest APIer som brukes.

5.4 Implementasjon av Android applikasjon



Figur 10 Applikasjonsskjermbilde

En oversikt flere foreløpige skjermbilder kan sees i vedlegg 6.

Et av de første stegene i implementeringsfasen startet med utvikling av grensesnittet for Android applikasjonen i Android Studio. Grensesnittet som ble utviklet var basert på designprototypen for bysykkelkonseptet som ble laget i planleggingsfasen. Grensesnittet er satt opp med en «bottom navigation bar» som er et moderne og gjenkjennelig layout for moderne applikasjoner.

Videre ble det implementert Google Maps funksjonalitet med grunnleggende funksjoner:

- Kartet zoomer automatisk inn på Kristiansand sentrum ved oppstart av applikasjon
- Brukers lokasjon hentes fra telefonens GPS og vises på kartet
- Plassering av statiske markører med plassering for tenkte sykkelstasjoner

Da det var en del usikkerhet rundt teknologivalg i oppstartsfasen ble det i første omgang implementert et fungerende innloggings grensesnitt med brukerautentisering via Firebase Auth for å teste innloggingsfunksjonalitet, siden dette kunne gjøres med relativt lite kode. Når valget for databaselagring til slutt falt på MySQL viste det seg også å være problemfullt å kombinere Firebase Auths nosql-baserte brukerlagring med vår MySQL database. Firebase Auth har kun støtte for å lagre e-mail og passord om brukere, så vi valgte å bytte dette ut med en egen autentiseringsløsning som fungerte bedre og var mer tilpasset webstacken vår, da det potensielt åpnet for flere muligheter for informasjonsfelter vi kunne kreve for registrering og innlogging, som f.eks telefonnummer adresse etc. om vi skulle trenge det.

Til syvende og sist førte endringen til en del mindre funksjonalitet relatert til innlogging og registrering, da vi måtte manuelt skrive REST APIer på webserveren i PHP, i stedet for et par linjer med kode på app siden ved hjelp av Firebase. Blant annet inkluderte dette tap mulighet for tilbakestilling av passord, registreringsbekreftelse på mail samt redusert sikkerhet siden vi måtte ta ansvar for å håndtere brukernavn og passord selv. Fordelen med å bruke egenutviklede web baserte APIer er at viktig kode kjører på en sentral webservice, som sørger for at vi må skrive mindre kode om vi skulle utvidet systemet til andre plattformer som iOS eller web.

Den nye innloggingsdelen bruker REST og Android Volley Library for å kommunisere med databasen på nett. Den inneholder også en SQLite database som brukes for å large en midlertidig aktiv brukersesjon. På serversiden blir brukerdata sendt og lagret til database ved hjelp av REST APIer. Passord blir behandlet gjennom en SHA hashing funksjon kombinert med en unik SALT nøkkel for hver bruker for å kunne lagre passord med en viss grad av sikkerhet som sørger for at personer med tilgang til databasen ikke kan lese passord i klartekst. Dette beskytter passord for rainbow table angrep, og minsker sjansen for brute force angrep. Basert på videre undersøkelse er ikke SHA + salt den sikreste løsningen for passordsikkerhet i 2018, da er for eksempel bCrypt en mer foretrukket krypteringsalgoritme.

Forbedret passordkryptering med bCrypt samt flere funksjoner rundt innlogging og registrering er funksjoner som kan implementeres senere i fremtiden, men vi valgte i stedet å fokusere tiden

på andre deler av systemet da Egde så på dette som viktigere for å kunne bevise konseptet for flåtestyring. Vi kunne også tatt i bruk autentiseringsrammerverk som OAuth for å håndtere brukerautentisering bedre.

Vi har til nå ikke kommet helt i mål med å hente lokasjonsdata fra databasen for dynamisk visning av sensorer på kart i appen, samt innsjekk/utsjekk av kjøretøy. Det som gjenstår er å gjøre ferdig arbeid med APIer på webserveren som sender denne dataen inn i en funksjon som aktivt viser lokasjoner på Android applikasjonen Dette er noe vi vil jobbe med og håper på å få på plass fram til presentasjon. Vi satser også på å få rebrandet applikasjonen fra bysykkelkonseptet til et mer generelt flåtestyringskonsept til presentasjon.

5.5 Implementering av MQTT funksjonalitet i Java

Med nevnte plan for utviklingen vil ikke kildekoden vår være robust. Det vil si at funksjonaliteten vi er ute etter fungerer, men systemet har ingen sikkerhetssjekker for overskriding av grenseverdier eller feil input fra brukeren. Altså ingen funksjonalitet for håndtering av uforutsette feil. Stabiliserende refaktorering vil derimot bli høyt prioritert etter hvert som mer og mer av kjernefunksjonaliteten er implementert og klar for omfattende automatisert testing. Av samme grunn er strukturen og oppbyggingen av klassene heller ikke er optimal. Vi er klare over at det eksisterer forbedringspotensiale som skal adresseres.

5.5.1 Kjernefunksjonalitet

I korte trekk består javaprogrammet av følgende klasser med funksjonaliteten:

- **MQTTSubscriberClient**: Denne klassen er ansvarlig for å lytte etter og motta MQTT-meldinger fra alle registrerte kjøretøy i systemet. Ved mottatt melding sjekker kjøres en funksjon som sjekker avsender og innhold, og basert dette bestemmes det hvilken funksjon i databaseklassen meldingen skal sendes til for videre behandling.
- **MQTTPublisherClient**: En egen klient for å publisere meldinger til kjøretøy. Dette kan være beskjeder som synkronisering av timestamp, ønske om at noen sensordata skal sendes på nytt eller en forespørsel om å endre rutine for sending av data. For eksempel å starte en direkte rapportering av sensordata direkte til administrasjon.
- **DatabaseHandler**: Mottar meldinger fra MQTTSubscriberClient. Deretter håndteres meldingen basert på dens innhold. Meldinger med sensordata er formatert etter en spesifikk konvensjon og rådataene blir delt opp til rådata. Deretter konstrueres et

SQL-statement, en kommando som databasen forstår og kan håndtere, og rådataene settes inn i denne kommandoen før den blir sendt videre til databasen.

5.5.2 Databasekommunikasjon

For å få Java / MQTT og SQL til å kommunisere bruker vi en Java DataBase Connectivity(JDBC) API. Dette grensesnittet sørger for at SQL-kommandoen man genererer i java kommer frem til og blir satt inn i databasen. [3]

For å få Android app til å kommunisere og oppdatere database bruker vi serverside script i from av RESTful services APIer skrevet med PHP. Rest bruker et standard sett med HTTP metoder (CRUD operations) de vanligste metodene innebærer POST, GET, PUT, PATCH og DELETE. [4] Skal f.eks informasjon om en bruker hentes brukes en GET http metode, og det generes en JSON respons som sendes til og tolkes av Android appen.

5.5.3 Aktører i systemet

Vi har ikke ferdigstilt klasser som skal representere de forskjellige brukere og kjøretøy i Java. Vi er usikker på om dette er nødvendig siden all relevant informasjon om kjøretøy (ID, posisjon, tilstand, sensordata) og brukere (ID, rettigheter, kontaktinformasjon) lagres direkte i databasen. Deretter blir informasjonen hentet ut og analysert (i utgangspunktet) av det eksterne analyseverktøyet PowerBI. Det må heller implementeres metoder for å endre tilstanden på hver enkelt aktør i databasen. Det kan tenkes at vi implementerer javafunksjonalitet for å hente og fremvise denne informasjonen direkte.

5.6 Utvikling av Arduino IoT mobil enhet

Vi fikk tildelt et mikroprosessor utviklingskort av typen UNO (Arduino Uno R3 klone), og et Sodaq IoT-NB (Internet of Things - NarrowBand) nettverkskort med tilhørende SIM-kort fra teleoperatøren Telia. NarrowBand er en relativt ny teknologi som er utviklet nettopp for å kunne koble IoT-enheter til et eksisterende 4G mobilnett og dermed kunne kommunisere trådløst i høy hastighet. [7]

Etter første runde med testing, og en del tråling av ressurser på nett kom vi frem til at Uno-kortet ikke var godt egnet til vårt bruk. Til dels på grunn av begrenset lagringsplass på mikroprosessoren, men også siden nevnte kort er utstyrt med kun én seriell buss (kanal) for kommunikasjon. Siden kortet må betjenes via en datamaskin holdes bussen opptatt via USB.

ino M0-SD,)Vi får dermed ikke kontakt med mobilnettet gjennom NB-IoT kortet så lenge mikroprosessoren er tilkoblet en datamaskin. Det førte til at vi ikke kunne teste funksjonalitet og avlese resultat samtidig.

På grunn av dette ble det bestilt av Egde et nyere og mer passende mikroprosessor (Elecrow Crowdu M0-SD, Arduino Zero klone) Utvikling av den mobile enheten ble dermed satt på vent til vi fikk tak i den nye mikroprosessoren. I tillegg til flere serielle busser har dette kortet mulighet for å sette inn et micro SD kort. Det er noe som er svært hensiktsmessig, siden vi jobber med innsamling av data, og dermed fort kan lide av begrenset lagringskapasitet.

Samtidig som det ble klart at det måtte bestilles en ny mikroprosessor fant vi en alternativ plan for å kunne fortsette å utvikle de andre delene av systemet. Vi konstruerte simulert sensordata i samme konvensjon som vår IoT-enhet ville generert og brukte disse istedenfor å få data direkte fra IoT-enheten.

Da Crowduino-kortet endelig ankom måtte vi etter et par dager med prøving og feiling med Elecrows og Sodaqs eksempelkodebibliotek kontakte Telia for å få konstatert at SIM-kortet vi var i besittelse av ikke var kompatibelt med Narrowband-teknologien. Dermed måtte vi vente halvannen uke på at et fungerende SIM-kort skulle komme i posten.
Etter å ha mottatt det nye SIM-kortet fikk vi fortsatt ikke koblet enheten til NB-nettet. I eksempelkoden fra Sodaq krevde tre forskjellige parametere for å kunne koble til mobilnettet (APN, CDP og forceOperator). Telia, som ikke hadde kjennskap til eller kunnskap om Sodaq sine kort kunne ikke gi oss annen informasjon enn APN-adressen og det faktum at Telia benyttet UDP istedenfor CDP. Vi fant en henvendelse på Sodaqs støtteforum om ett tilfelle likt vårt eget, der det heller ikke var presentert en løsning.

Vi valgte derfor å legge hele NB-IoT enheten på is og bruke den resterende tiden vår på å videreutvikle de andre delene av systemet.

6. Kvalitet

Kvalitet som konsept kan defineres i mange forskjellige retninger avhengig av kontekst. I vårt tilfelle definerer vi kvalitet som:

"Systemets evne til å raskt og feilfritt utføre de oppgaver som kreves fra både eier og forbruker."

Kvaliteten på et prosjekt eller produkt blir påvirket av eksterne faktorer som vi kan se i prosjektsstyringstriangelet. De tre faktorene er penger tid og omfang av prosjektet. Justeringer på en av de tre faktorene påvirker de to andre faktorene. Dersom det kuttes i budsjettet vil det påvirke tiden det tar å fullføre, og nødvendigvis tvinge omfanget av prosjektet ned. Dersom prosjektet skaleres opp eller ned vil det føre til økt eller redusert behov for finansiering og tid. På samme måte, dersom en tidsfrist flyttes frem vil det kreve flere økonomiske midler og også kreve en nedskalering av omfanget. [7]

Figur 11 Kvalitetstrekant

6.1 Kvalitetssikring

I vårt prosjekt har vi jobbet med store begrensninger i penger (ressurser) og tid, og med i utgangspunktet et ganske stort omfang. Tidsrammen har vært på rundt regnet fire måneder (ett semester) og budsjettet vårt har i utgangspunktet vært på kroner 0,-. Gjennom prosjektgjennomføringen har vi som tidligere nevnt justert omfanget opp til flere ganger. Dette har lettet på tidspresset rundt prosjektet, men ikke hatt så mye å si for ressursbruken siden budsjettet vårt har vært ikke-eksisterende. Vi har fått tildelt det vi har hatt behov for av programvarelisenser og maskinvare vi ikke har hatt tilgang til selv fra vår oppdragsgiver.

6.1.1 Omfang(scope)

Vi har fått i oppgave å utvikle et flåtestyrings- og datainnsamlingssystem med trådløs kommunikasjon via mobilnett med en fungerende mobilapplikasjon og et web/javabasert grensesnitt. Selve systemet består av ikke mindre enn seks forskjellige teknologier (Java, MySQL, Arduino, Android, NarrowBand, MQTT). Alle de forskjellige bestanddelene i systemet har sine egne rutiner regler og syntakser. Det impliserer at det kreves fem ganger så mye studering for å lære seg å bruke teknologien. Vi har valgt akkurat disse teknologiene fordi vi kjenner til og har jobbet med flere av teknologiene tidligere. Det reduserer dermed arbeidsmengden det kreves for å lære seg og dermed kunne bruke teknologiene. I tillegg har vi flere ganger i løpet av prosjektet raffinert og omdefinert omfanget av oppgaven og dermed gjort prosjektet nærmere gjennomførbart.

6.1.2 Ressurser(money)

Vår oppdragsgiver har vært behjelpelig med å tilby oss de verktøy som har vært nødvendig for å kunne jobbe med prosjektet. Disse verktøyene er i vår situasjon synonymt med penger eller ressurser, fordi det er noe som i et tradisjonelt prosjekt koster penger. De nevnte verktøy er både programvare og maskinvare som for eksempel

- Lisens til prosjektstyringsprogrammet Jira (scrum), analyseverktøyet Microsoft PowerBI, kildekodekontroller BitBucket
- Crowduino M0-SD (Arduino-klone) og Sodaq NB-IoT nettverkskort.

6.1.3 Tid

Tidsaspektet har vært det mest utfordrende å forholde seg til. Med i overkant av fire måneder til rådighet har vi måttet gjøre harde prioriteringer for å fullføre de mest vesentlige delene av systemet. Vi har fokusert på å få en fungerende nettverkskommunikasjon med tilhørende databasetilkobling, og utvikling av android-app til sluttbruker og admin-grensesnitt.

6.2 Prosesskvalitet

Prosesskvaliteten forteller oss i hvor stor grad vi som et utviklerteam sørger for at produktet eller systemet vi utvikler får de riktige egenskapene. [8]

For å sikre prosesskvalitet i vårt prosjekt har vi gjort følgende:

- Kontinuerlig dialog med oppdragsgiver om status fremgang og retning på prosjektet. Hvilket aspekt av systemet er det mest interessante, og dermed viktigst å prioritere.

- Godt definerte og spesifikke arbeidsoppgaver for alle medlemmer i gruppen og kontinuerlig god kommunikasjon innad i gruppen.
- En grundig analyse og planleggingsfase i starten av prosjektet med kontinuerlig justeringer basert på tilbakemelding fra oppdragsgiver.
- Demonstrasjon av systemet til oppdragsgiver underveis i utviklingsprosessen der vi viser fungerende funksjonalitet etter hvert som den er klar og jobber videre på tilbakemeldinger og innspill fra oppdragsgiver.

6.3 Produktkvalitet

Produktkvaliteten forteller i hvor stor grad systemet i seg selv har de riktige egenskapene og produserer de ønskede resultater. Denne kvaliteten måles ut fra kriterier som for eksempel sikkerhet, brukervennlighet, pålitelighet, ytelse og vedlikeholdsbehov

For å sikre produktkvaliteten i vårt prosjekt har vi gjort følgende:

- Kontinuerlig kommunikasjon med oppdragsgiver angående ønsket oppførsel og resultat i systemet.
- Analysert bruksområde og tilpasset sluttproduktet til å ha størst mulig nedslagsfelt blant potensielle brukergrupper.
- Undersøkt og valgt de teknologier som er best egnet for vårt system. Vi valgte å gå i størst mulig grad for teknologi vi har tidligere erfaring med eller kjennskap til for å minimere tid brukt på å lære ny teknologi.
- Øremerket tid til opplæring av ny teknologi som er nødvendig for å fullføre prosjektet

6.4 Kvalitet i vårt prosjekt

Det var vært vanskelig å definere kvalitet i et prosjekt som skal fungere som et proof of concept av et datainnsamlingsrammeverk. Man kan se på kvalitet fra flere perspektiv, en erfaren utvikler kan ha høyere krav for kvalitet, enn oss som relativt uerfarne utviklere. Kvaliteten har i stor grad blitt definert gjennom kommunikasjon med vår arbeidsgiver Egde, basert på ønsker de hadde for systemet, og vår erfaring. Til sammen dannet dette kriterier og mål vi kunne strekke oss etter, noe som til slutt definerte kvaliteten på prosjektet for å nå et minimum viable product.

I løpet av prosessen så vi oss nødt til å gjøre jevnlige justeringer av scope, noe som også endret kriterier for kvaliteten på produktet. Det viktigste kriteriet vi har for kvalitet er å kunne demonstrere et konsept av et modulert rammeverk for flåtestyring og datainnsamling.

Disse kvalitetskravene har vi brutt ned til mindre kriterier:

- Lage et system som fungerer som proof of concept
  Modulært system som i utgangspunktet kan tilpasses til andre klienter
- Systemet kan samle inn og sende sensordata over lettvektsprotokoll
- Skytilkobling med datalogikk og behandling på sentral server
- Fungerende dataoverføring og behandling over nett mellom sensor, server og applikasjon
- Mindre fokus på detaljer i grensesnitt, funksjon over design
- Visualisering av data gjennom verktøy som PowerBI

I skrivende stund er ikke alle disse kravene nådd. Vi er nesten der, men det er et par justeringer igjen med kommunikasjon mellom enheter i systemet for å demonstrere flyt og framvisning og visualisering av data. Dette er noe som vil bli arbeidet med i siste innspurt før presentasjon.

6.5 Risikoanalyse

I en systemutvikling fase er vurdering av risiko en viktig del for å opprettholde kvaliteten i produktet. Derfor tok vi i bruk en risikomatrise for å vurdere hvilke aspekter av prosjektet som var viktig å fokusere på og prioritere, og for å finne ut av diverse problemer som kunne dukke opp. Vi tok utgangspunkt i en risikomatrise som var tilgjengelig for oss på Canvas. Denne matrisen evaluerer sannsynligheten i sammenheng med konsekvensen for å finne ut risikoen av den problemstillingen. Vi bruker resultatet av dette for å finne ut alvorlighetsgraden rundt det, slik at vi kan ta tak i problemet så tidlig som mulig. Grunnen til at vi bruker risikomatrisen er for å kunne opprette tiltak for at problemene ikke skal skje, og for å planlegge en prosedyre å iverksette dersom problemet likevel skulle oppstå.
Formel: Sannsynlighet x Konsekvens = Risiko

Lite sannsynlig:	1	Lite farlig:	1
Sannsynlig:	2	Farlig:	2
Meget sannsynlig:	3	Kritisk	3
Svært sannsynlig:	4	Katastrofal	4

Risiko	Konse kvens	Sannsynlig-het	RISIK O	Tiltak for at ikke skje	Tiltak, redusere skade
Usikkerhet rundt krav, funksjonalitet	4	3	12	Hyppig kontakt med Egde	Være i dialog med Egde om noe er usikkert
Manglende kompetanse i prosjektet	4	3	12	Bygge kompetanse, spørre om hjelp fra Egde så tidlig som mulig	Finne tilsvarende prosjekter på nettet, og gjøre det samme, dokumentere det
Ikke svar, oppfølging fra oppdragsgiver	3	1	3	Hyppig kontakt med Egde, fokus på verdi til Egde	Ta egne valg hvis ikke svar. Dokumentere dette.
Uklart i overgangen fra analyse/design til implementering	4	4	16	- Involvere analyse, design tett inn mot implementering - hyppige leveranser, sjekk mot analyse/design	Hyppige leveranser, ærlig feedback fra brukere og Egde
Gnisninger i samarbeidet	4	1	4	Gi hyppige tilbakemeldinger, alle bidrar.	Være lojale ovenfor gruppemedlemmene rundt oppgaven

7. Testing

Testing er en viktig del av programvareutvikling og gjøres for å forsikre at programmet kjører optimalt og er i stand til å håndtere de feil og eventualiteter som måtte oppstå. For eksempel hvis en bruker prøver å sette inn informasjon som er duplikat, feil format eller utenfor grenseverdiene til systemet, eller dersom en eller flere moduler i systemet mister tilkobling eller krasjer.

Målet for prosjektet vårt har vært å lage et rammeverk for flåtestyring som et proof of concept. Vi har derfor ikke hatt et stort fokus på å teste robusthet, men heller fokusert på å få kritiske aspektet rammeverket til å vise at faktisk fungere for å skape et «minimum viable product» som kan itereres på for å forbedres videre. Med faktorer som erfaring, tidspress, omfang og kompleksitet, har vi ikke prioritert å sette opp et rammeverk for testing. Vi er likevel klar over viktigheten og fordelene med automatisert testing. Istedenfor å sette opp et testmiljø har vi kontinuerlig kjørt manuelle tester underveis i utviklingen.

Testingen vi har utført per dags dato omhandler i stor grad å få godkjent fungerende funksjonalitet. I en eventuell videreutvikling av systemet må det kjøres tester på grenseverdier, sikkerhet, datakompatibilitet og feilhåndtering.

7.1 Testing av MQTT og IOT-modul

Siden systemet vårt, og dermed testingen av systemet, består av minimum 4 forskjellige teknologier (database, java, MQTT og Android IoT-modul), og kombinasjoner av nevnte teknologier (java/SQL-bro og MQTT-javaklient) vurderte vi det som hensiktsmessig å kjøre tester manuelt. Å skrive automatiserte tester som sømløst integrerer alle de forskjellige teknologiene antok vi til å være like tidkrevende og komplekst som utviklingen av selve systemet. Vi valgte derfor å kjøre manuelle tester der vi hadde full kontroll på hvor når og hvordan de forskjellige hendelsene i systemet forekom.

Vi kjørte manuelle tester blant annet på:

- Kommunikasjon mellom alle entiteter.
- Separasjon av meldingsinnhold; meldinger som ikke inneholder sensordata (eller inneholder sensordata i feil format) blir behandlet annerledes enn meldinger med sensordata.

- Registrering av dupliserte meldinger (med samme primary key) i databasen.
- Mottak av meldinger med flere linjer sensordata.

7.2 Testing av Android applikasjon

For å kvalitetsteste android applikasjonen utførte vi manuelle tester etter endringer for å forsikre oss om at ting fungerte som det skulle. Vi tok også i bruk Android Robo Test som er en del av Firebase. Med Android Robo Test kunne vi kunne lage automatiserte grensesnitt tester og prøve applikasjonen på flere virtuelle enheter i skyen. Vi tok også i bruk Android Studios innebygde debugger og konsoll for å rette opp feil som hindret koden i å kompilere korrekt. Vi fikk dessverre ikke tid til å sette oss inn i å skrive manuelle Junit tester for å teste applikasjonen, men dette er noe vi ville fokusert på om vi hadde kommet lengre med prosjektet.

7.3 Testing av Server APIer

På serversiden brukte vi et program som heter Postman for å teste at API kallene til og fra serveren fungerte som forventet, og at informasjon som ble sendt og mottatt ble behandlet på forventet måte. Dette sparte oss for mye tid i stedet for å bruke tid på å implementere et REST API kall i Android appen som inneholdt feil eller mangler.

8. Refleksjon

I de neste avsnittene tar vi for oss refleksjon rundt prosjektet. Refleksjonen tar utgangspunkt i våre forventninger til prosjektet som emne, resultater, samarbeid, metodikk og kontakt med oppdragsgiver.

8.1 Prosjektstyring

Vi ser på scrum og andre agile prosjektledelse praksiser som svært effektive metoder for å styre et prosjekt. Vi har opplevd at det krever mye disiplin, dedikasjon og egeninnsats for at scrum-rammeverket skal komme til sin fulle rett. Både med tanke på administrasjon, og alle de forskjellige møter og papirarbeid som medfølger. Vi har opplevd at scrum tar mye fokus, og at om man ikke har full kontroll på alle aspekter i rammeverket er det lett å miste inspirasjon og motivasjon, og ganske lett falle av lasset. Vårt valg om å kjøre en flat maktstruktur heller enn å dedikere en scrum-master har også videreført til noe ineffektivitet i prosjektstyringen. Vi har

innsett at en scrum master med et overordnet ansvar for det daglige virket i gruppen er essensielt for en trygg, tydelig og effektiv prosjektgjennomføring.

Til forbedring kan det være hensiktsmessig å friske opp teorigrunnlaget og strukturen i scrum-rammeverket. Dette kan naturligvis gjøres individuelt, likevel kan det lønne seg å dedikere en eller flere forelesninger om scrum, etterfulgt av en liten obligatorisk oppgave eller test i starten av semesteret.

8.2 Kommunikasjon

I dette prosjektet har vi lært at kommunikasjon er essensielt i en systemutviklingsprosess. Internt i gruppen har vi som regel brukt Facebook som kommunikasjonsmiddel når vi har planlagt møter, eller om det var noe vi lurte på når vi jobbet individuelt. Siden alle tre er relativt aktive på Facebook, har det fungert helt utmerket.
Kommunikasjonen med Egde har utenom faste møter gått enten via Skype eller mail. Dersom vi lurte på noe utenom våre planlagte møter, kunne vi sende mail og få raskt svar.

8.3 Tid og produktivitet

Å jobbe med en oppgave med så stort omfang var nytt for oss alle, så var det vanskelig å estimere hvor lang tid ting vil ta. I tillegg til å komme litt sent i gang, feilestimerte vi hvor lang tid oppgavene ville ta, og derfor ble vi fort hengende litt etter og måtte ta med oss oppgavene fra gamle sprinter inn i nye sprinter. Selv om vi prøvde å hente oss inn, rakk vi aldri det, så vi ble kontinuerlig hengende etter på sprintene. Det er lett å si at om produktiviteten hadde vært enda høyere hadde vi klart det, men vi følte at vi ikke hadde mer å gå på.

8.4 Utfordringer

Vi merket at det har vært utfordrende og tidkrevende å lære oss nye språk og programmer for å få fremgang i prosjektet. Et slikt prosjekt krever at enkelte ting må implementeres og fungere først for at vi skal komme videre i prosjektet, og vi merket at en slik flaskehals var med på å hindre fremdrift i vårt prosjekt. I begynnelsen slet vi med å finne riktig kort som vi kunne bruke, og det var også utfordrende å se sette seg inn i backend delen med rest API og MQTT.

8.5 Samarbeid med oppdragsgiver

Egde har hele veien vært klare på at dersom det er noe de kan gjøre for at vi skal ha fremgang i prosjektet, så var det bare for oss å spørre. De har vært villige til å bistå med sensorer, fikse adganger til systemer og gjort alt klart for at vi skal kunne jobbe med oppgaven. Dersom det var tekniske ting vi lurte på, så kunne vi spørre Trond som var ansvarlig for den tekniske delen. Vi hadde også mulighet til å bruke deres lokale for arbeid med oppgaven, men siden det var i Grimstad, så var det lettere for oss å jobbe her i Kristiansand.

8.6 Samarbeid med veileder

Vi har brukt store deler av tiden til å jobbe med den tekniske delen av oppgaven, der har vi stort sett brukt Egde til veiledning. Det er ikke før på slutten, når rapportskriving tar større del at vi har hatt ordentlig bruk for veiledning. Vi har vært i kontakt med vår veileder Hallgeir i tidligere faser av prosjektet, men vi burde ha vært i hyppigere kontakt med han i løpet av denne perioden.

9. Resultat

På det tidspunktet rapporten ferdigstilles har vi ikke lyktes med å nå de målene vi har satt oss. Det er flere årsaker til dette, blant annet teknologi som ikke har fungert som håpet, kontinuerlige endringer i prosjektplanen, oppgaver langt mer tidkrevende enn estimert og noe ujevnt prosjektstyring. Derimot har vi deler av systemet oppe og gå.

- Fungerende nettverkskommunikasjon for systemet (med unntak av NarrowBand-tilkobling) via REST API for tilkobling mellom Android app og database og MQTT til resterende kommunikasjon.
- Serverlogikk for å lese, tolke og behandle innkommende meldinger korrekt, og videre sende data til database. MySQL database som tar imot og lagrer all data.
- Android applikasjon som har mulighet for innlogging og registrering av bruker. Implementasjon av kart funksjonalitet (trenger noen justeringer i backend APIER og app for å hente dynamisk data fra sensor)

Vi har fortsatt noe som gjenstår for å få et helhetlig fungerende system som er

- Nettleser/web-GUI
- PowerBI dataanalyse

Vi kommer til å jobbe videre med prosjektet etter at rapporten er levert og forhåpentligvis ha en versjon klar for demonstrasjon til muntlig eksamen 5. juni.

På tross av en svært bratt læringskurve, og alle problemer og utfordringer vi har støtt på er vi alle enige om at dette har vært et interessant og spennende prosjekt som vi gjerne kunne tenke oss å lære mer om og jobbe videre med.

Litteraturliste

[1]Mountain Goat Software. *Planning poker*

Hentet 10. mai 2018 fra

https://www.mountaingoatsoftware.com/agile/planning-poker

[2] Wikipedia. Lamp Software Bundle

Hentet 15. mai 2018 fra

https://upload.wikimedia.org/wikipedia/commons/thumb/8/82/LAMP_software_bundle.svg/640px-LAMP_software_bundle.svg.png

[3] TutorialEdge. *Rest API Basics*

Hentet 15. mai 2018 fra

https://tutorialedge.net/uploads/rest-api.png

[4] The Server side. (september 2005). *Java database connectivity-JDBC*

Hentet 11. mai 2018 fra

https://www.theserverside.com/definition/Java-Database-Connectivity-JDBC

[5] RESTAPITutorial. Using http Methods for RESTful Services
Hentet 15. Mai 2018 fra
http://www.restapitutorial.com/lessons/httpmethods.html

[6] Telia. (desember 2016) *Nyhet! Tingene dine har fått sitt eget nettverk*

Hentet 13. mai 2018 fra

https://telia.no/magasinet/tingene-dine-har-fatt-eget-nett

[7] Microsoft. (2007). *Every project plan is a triangle.*

Hentet 9. mai 2018 fra

https://support.office.com/en-us/article/Every-Project-plan-is-a-triangle-2B74C21B-A406-4727-8D74-26648A56924A#bkmk_whatistheprojecttriangle

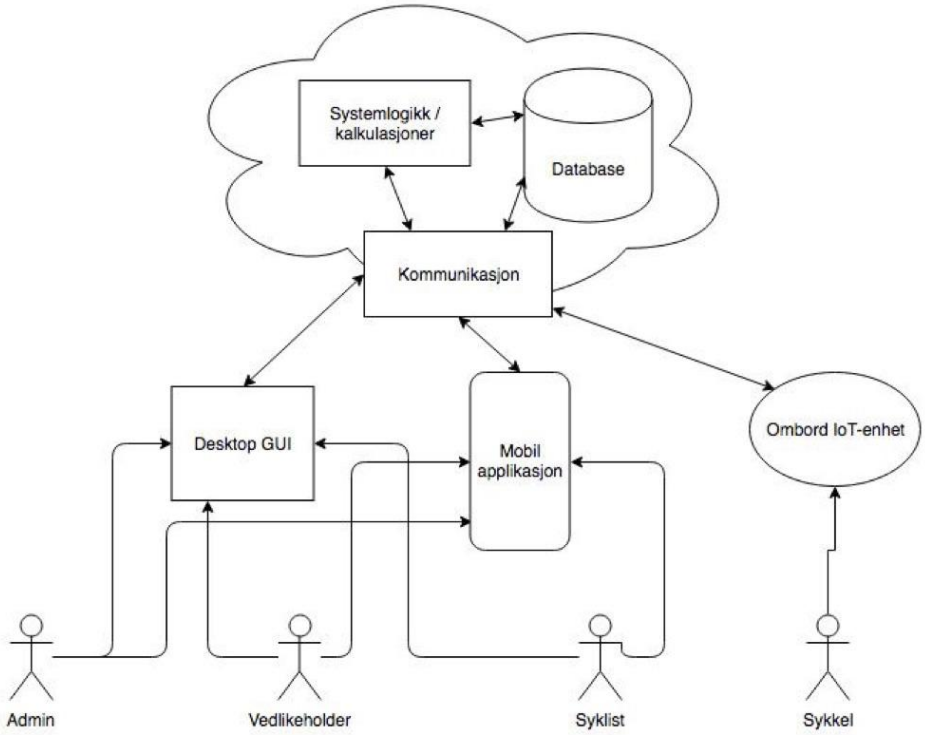
[8] TaskManagementGuide. *What is process quality*

Hentet 9. mai 2018 fra

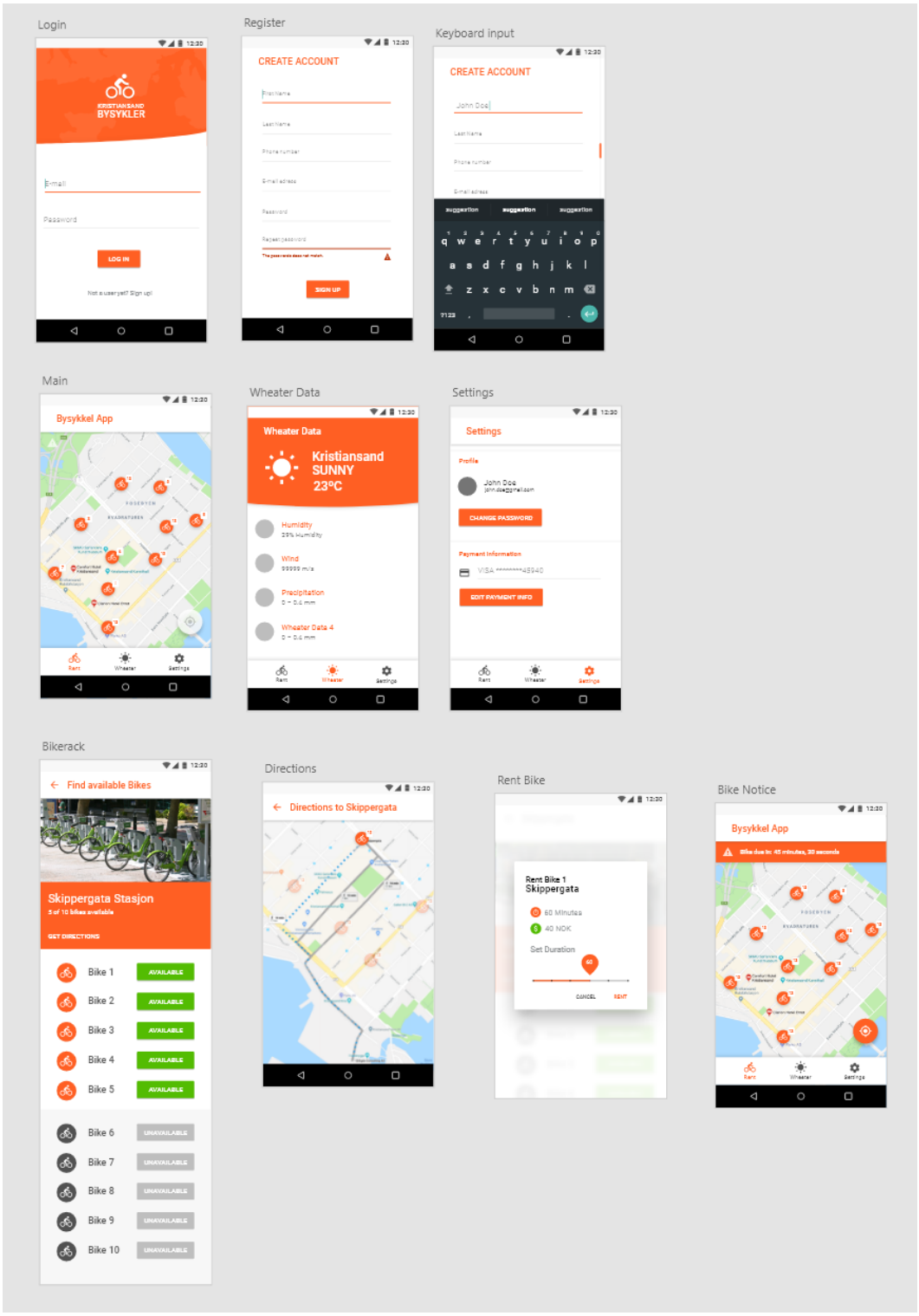
http://www.taskmanagementguide.com/glossary/what-is-process-quality.php

Vedlegg

Vedlegg 1 – Rikt Bilde versjon 1

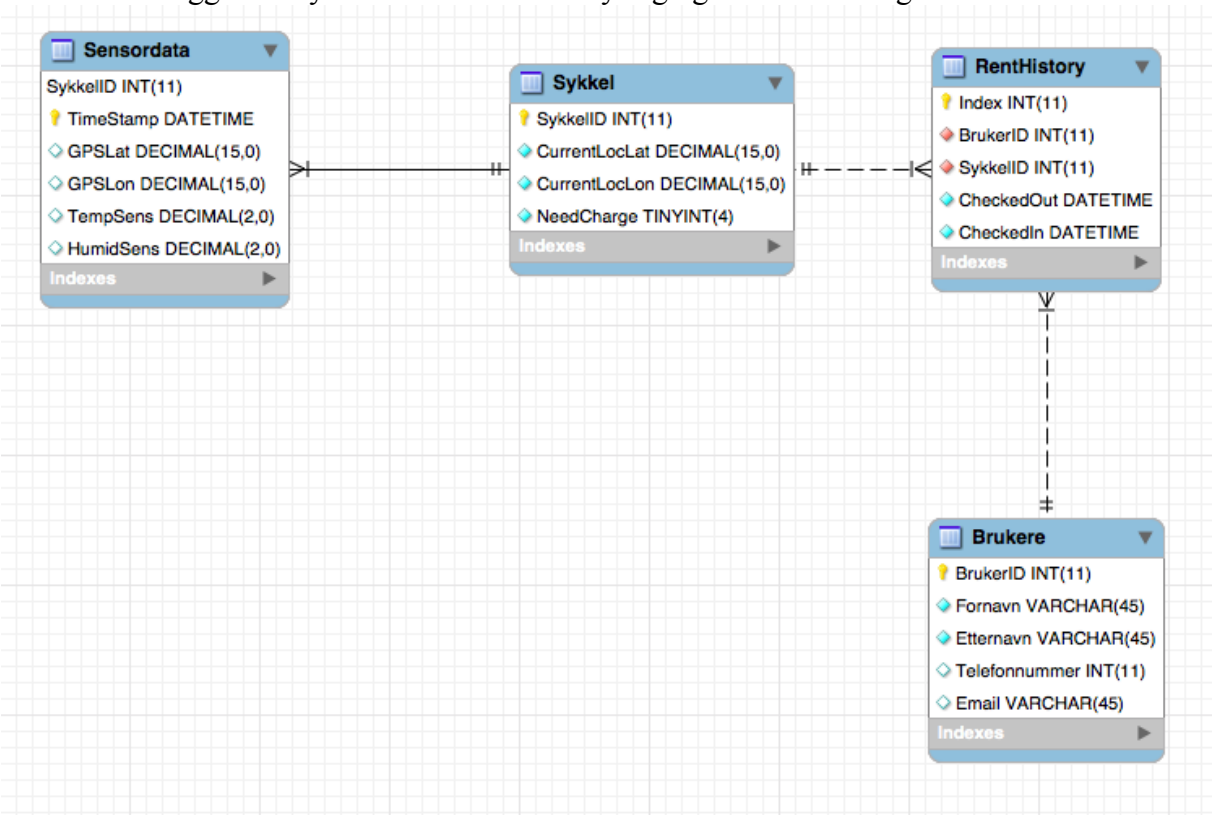


Vedlegg 2 – Skjermbilder av prototypen



Vedlegg 3 – EER Diagram

Dette er et tidlig utkast av EER diagram brukt for å konstruere database. Dette ble i utgangspunktet laget for bysykkel konseptet, og vil bli oppdatert fram mot presentasjon for å bedre tilrettelegges de nye kriteriene for flåtestyring og datainnsamling.



Vedlegg 4 – Funksjonsliste

Funksjon i systemet	Kompleksitet	Type/mål	Involverte moduler
Registrere bruker	Lav	Registrering	Admin, system, bruker
Slette bruker	Lav	Oppdatering	Admin, system, bruker
Logge inn	Høy	Avlesning	Admin/bruker, system
Logge ut	Lav	oppdatering	Admin/bruker, system
Booke kjøretøy	Medium	Registrering	System, bruker, kjøretøy
Slette booking	Lav	Oppdatering	Bruker, system
Levere kjøretøy	Lav	Oppdatering	Bruker, system, kjøretøy
Sjekke lokasjon på ett/flere kjøretøy	Medium	Avlesning	Admin, system, database
Vis reisehistorikk på flere kjøretøy	Medium	Avlesning	Admin, system
Registrere sensordata og lagre lokalt i sensorenhet	Medium	Registrering	Sensorenhet
Slette lokalt minne i sensorenhet	Lav	Oppdatering	Sensorenhet
Sende sensordata til database	Medium	Nettverkskommunikasjon	Sensorenhet, nettverkskommunikasjon, Java
Registere sensordata i database	Lav	Registrering	Java, system, database
Hente data fra databasen	Medium	Avlesning	Database, java
Analysere data	Høy	Beregning	Admin system, database, analyseverktøy
Starte/stoppe live feed av sensordata	Høy	Oppdatering	Sensorenhet, nettverkskommunikasjon, admin
registrere/slette booking	Lav	Registrering	Admin, kjøretøy, system, database

Vedlegg 5 - Uttalelse fra oppdragsgiver

Uttalelse fra Egde Consulting

Oppdragsbeskrivelse
Egde Consulting er Sørlandets ledende selskap innen digitale tjenester og er spesialisert innen teknologi, rådgivning, utvikling og design. Vår digitale og forretningsmessige kompetanse bidrar til verdiskapning ved å forbedre, fornye, og forenkle våre kunders virksomhet. I forbindelse med flere ulike kundecaser ønsket Egde at studentene utvikle en løsning for flåtestyring av bysykler i kombinasjon med innsamling av sensordata som temperatur/luftkvalitet/støy osv. Det viktigste for Egde var å ta frem en konseptuell løsning for en helhetlig verdikjede som inneholdt sensorer, datainnsamling, flåtestyring, lokasjon og brukerinteraksjon.
Vi ønsket særlig at prosjektet var Javabasert, og inneholdt NB-IoT, MQTT som innsamlingsprotokoll, skyløsning for datalagring, PowerBI for datavisualisering og App utvikling.

Påkrevd funksjonalitet

- *Trådløs nettverkskommunikasjon*
- *Automatisk rapportering av posisjon og sensordata*
- *Lagring av data i database i skyløsning*
- *Visualisering av data*
- *Mobil App*

Inntrykk av arbeidet til gruppen
Studentene har vært nysgjerrige og ivrig på å ta i bruk ny teknologi og har vært ambisiøse med tanke på at oppgaven spenner over veldig mange fagfelt og teknologier innen SW-utvikling. App-delen av oppgaven er som veileder kanskje den enkleste å evaluere. Denne ble tidlig implementert til en meget god kvalitet med god forståelse for brukerinteraksjon.
Vil også trekke frem at studentene har jobbet etter etablerte beste praksis ved å skissere Use caser, brukerhistorier og systemarkitektur. De har også fått erfaring i å bruke smidig metodikk og bruke Jira som prosjektadministrasjonsverktøy.

Forbedringspotensialet i oppgaven deles mellom studentene og veiledere som har sett at oppgave definisjonen var litt for løs. Her burde vi jobbet mer i starten for å definere oppgaven.

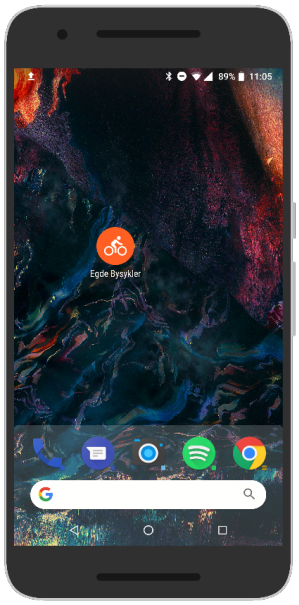
Studentene har levert en meget allsidig og vid oppgave som bør gi dem en god byggestein for videre arbeidsliv. De har også lært bruk av beste praksis metodikk og verktøy.

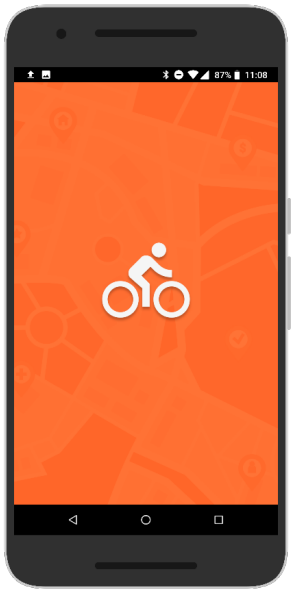
Konklusjon
Vi i Egde Consulting er fornøyd med hvordan studentene har løst oppgavene

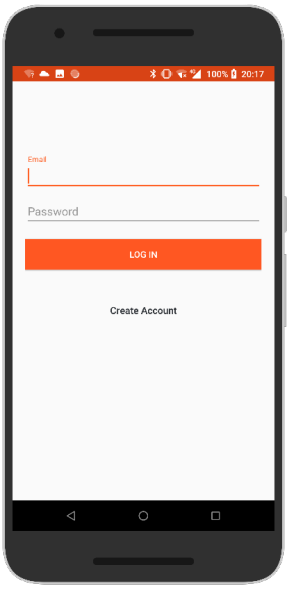
Dato / Sted	Elin Kilen
16.05.2018	Elin Kilen

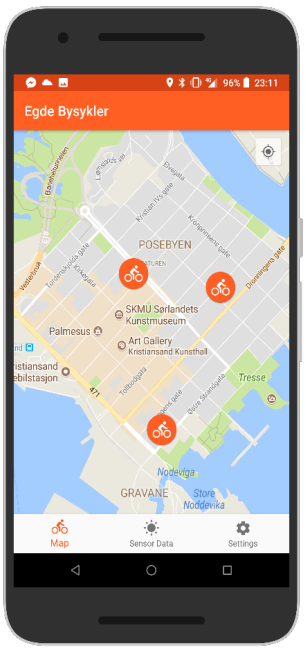
Vedlegg 6 – Skjermbilder fra app

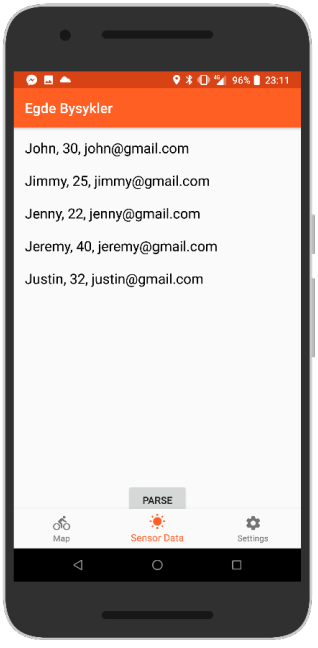
Skjermbilde 1: Launcher ikon på hjemskjerm
Skjermbilde 2: Splashscreen
Skjermbilde 3: Innloggings GUI
Skjermbilde 4: Hovedskjerm med kart
Skjermbilde 5: Sensordata visning (foreløpig er det testet med JSON parsing av en tilfeldig test JSON fil) Trenger noen justeringer av API på server for å sende JSON respons av sensordata.
Skjermbilde 6: Veldig simpel settingside med log ut funksjon, her er det plass for å plassere flere eventuelle innstillinger.

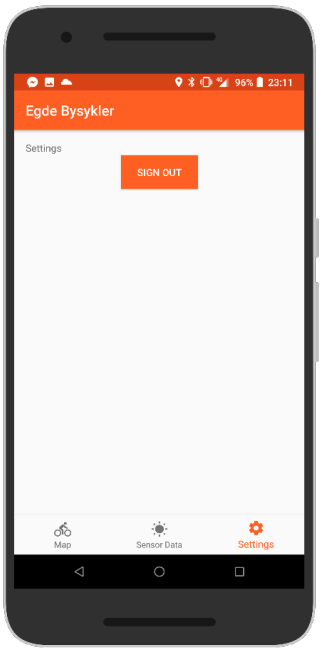












Vedlegg 7 – Brukerhistorier

Brukerhistorie 01 – Oversikt over sykler	
Brukerhistorie	Som **SYKLIST** ønsker jeg å se hvor og hvor mange sykler som er tilgjengelig i mitt nærområde til en hver tid slik at jeg enkelt kan finne en sykkel jeg vil leie.
Beskrivelse	Når jeg skal hente en sykkel vil jeg vite hvor jeg må gå for å finne en sykkel
Hvorfor/viktighet	Det er avgjørende å vite hvor man kan finne nærmeste ledige sykkel. Kritisk funksjon for demonstrasjon av systemet
MoSCoW	Must have
PP-score (Fibonacci)	
Akseptansekriterie	Minimum: Vis ledige sykler ut i fra fri lokasjon på kart. Utvidet: Sykler grupperes i stasjoner
Involverte elementer	GUI med kartfunksjonalitet, Database, kommunikasjon mellom entiteter
Testoppskrift	

Brukerhistorie 02 - Reiserute	
Brukerhistorie	Som **SYKLIST** vil jeg kunne legge opp en reiserute i kart på telefonen som viser den beste ruten.
Beskrivelse	Man skal kunne trykke på en knapp som fører til en visning for den raskeste ruten fra nåværende lokasjon til nærmeste eller valgt stasjon.
Hvorfor/viktighet	Dette er en grei funksjon å ha, men er ikke nødvendig for at hovedformålet med systemet skal fungere. Det finnes andre løsninger som f.eks Google Maps som kan tas i bruk som substitutt.
MoSCoW	Could Have
PP-score (Fibonacci)	

Akseptansekriterie	Dersom bruk av kollektivtransport er hensiktsmessig skal det inkluderes. Bruker får beskjed når destinasjonen er nådd
Involverte elementer	GUI, Lokasjonsinformasjon fra database og brukers mobil. Google Maps API
Testoppskrift	Innlogget bruker velger ønsket stasjon og får mulighet til å velge «Få veibeskrivelse» Bruker får beskjed om at destinasjonen er nådd og får mulighet til å avslutte navigasjonshjelpen

Brukerhistorie 03 – Oversikt over sykkelbeholding	
Brukerhistorie	Som **ADMINISTRATOR (og VEDLIKEHOLDER)** vil jeg ha en oversikt over beholdningen av sykler på hver enkelt stasjon, hvor hver enkelt sykkel befinner seg og om sykkelen er ledig eller i bruk.
Beskrivelse	Systemet skal samle data om alle sykler og ruter og presentere det på en enkel måte for administrator.
Hvorfor/viktighet	Analysere trafikk mellom stasjoner, administrere og planlegge tømming og etterfylling av stasjoner. Det må være lett å se at alle ressurser i systemet er gjort rede for og fungerer optimalt. Dette for å raskest mulig kunne finne feil og mangler for å utbedre disse
MoSCoW	Must Have
PP-score (Fibonacci)	
Akseptansekriterie	Systemet leverer en oversikt over stasjoner med antall sykler på hver stasjon, samt tilstand for hver enkelt sykkel. Samt statistikk for trafikk til og fra stasjon og rutehistorikk for alle sykler på den aktuelle stasjonen.
Involverte elementer	GUI, kommunikasjon, database, sykkelGPS, stasjon
Testoppskrift	

Brukerhistorie 04 - Stasjonsvarsling	
Brukerhistorie	Som **VEDLIKEHOLDER** vil jeg ha et automatisk varsel når sykkelbestanden på en stasjon går over eller under en grense, som indikerer at stasjonen må tømmes eller fylles på.

Beskrivelse	Vedlikeholder har behov for å vite hvor og hvilke stasjoner som har behov for ettersyn. Dersom en stasjon på egenhånd får justert sin sykkelbeholdning innenfor grenseverdiene må vedlikeholder også få beskjed om dette.
Hvorfor/viktighet	At stasjonene sender varsel når sykkelbestanden nærmer seg full eller tom hjelper vedlikeholder til å planlegge sine arbeidsoppgaver ift tidsbruk og reiserute. Dette er også viktig for at syklist alltid skal ha mulighet til å hente tilgjengelige sykler og levere sykler på alle stasjoner.
MoSCoW	Must have
PP-score (Fibonacci)	
Akseptansekriterie	Stasjonene sjekker sin egen sykkelbeholdning opp mot predefinerte grenseverdier. Dersom grenseneverdien overskrides sendes et varsel til systemet som blir distibuert til vedlikeholder og admin, som igjen handler på grunnlag av det.
Involverte elementer	Stasjon, kommunikasjon, vedlikehold
Testoppskrift	1. grenseverdier overskrides i begge retninger 2. sjekk at beskjed blir sendt i riktig tid og format 3. sjekk at beskjed reiser riktig gjennom systemet 4. sjekk at aktuelle parter mottar varsel på riktig måte (lyd, vibrasjon, format etc) 5. sjekk at beskjed kan bekreftes mottatt.

Brukerhistorie 05 – Historikk over sykkeldata	
Brukerhistorie	Som **ADMINISTRATOR** vil jeg kunne se en historikk over alle ruter en sykkel har tatt, og hvilken bruker ID som har brukt sykkelen på det aktuelle strekket. Hvor ofte, hvor mange og når på døgnet hentes og leveres sykler til hver enkelt stasjon.
Beskrivelse	Administrator skal kunne se i kart og listevisning hvilke sykler som har tatt hvilke ruter. (med mulighet for å filtrere på timestamp, område, sykkelID, brukerID etc).
Hvorfor/viktighet	Forebygge tyveri og hærverk på sykler ved at den som henter ut sykkel kan identifiseres. Også for å kunne analyser brukermøster/sykkelruter for å optimalisere logistikk tømming/etterfylling av stasjoner.
MoSCoW	Must have
PP-score (Fibonacci)	

Akseptansekriterie	Admin skal kunne se en detaljert oversikt med mulighet for filtrering over alle sykkelturer som har blitt foretatt, med tilhørende metadata som sykkelID, start-/slutttidspunkt, GPSdata, brukerID etc.
Involverte elementer	Bruker, sykkelGPS, kommunikasjon, database, admin GUI
Testoppskrift	1. Relevant data må eksistere i databasen 2. RD hentes fra database 3. RD må overføres komplet og korrekt til adm GUI 4. RD må fremvises korrekt i adm GUI 5. RD må kunne sorteres og filtreres etter ønske

Brukerhistorie 06 – Historisk oversikt over sykkeldata	
Brukerhistorie	Som **ADMINISTRATOR** vil jeg kunne se en historisk oversikt over alle ruter en sykkel har tatt, og hvilken bruker ID som har brukt sykkelen på det aktuelle strekket.
Beskrivelse	Oversikten vises gjennom et web-grensesnitt med utvidet funksjonalitet for administrator
Hvorfor/viktighet	Viktig for å forebygge og overvåke tyveri og herverk på sykler samt for å kunne analysere brukemønster for å kunne optimalisere logistikk og påfyll/tømming av stasjoner. Viktig for å kunne administrere og luke bort brukere som misbruker systemet.
MoSCoW	
PP-score (Fibonacci)	
Akseptansekriterie	Syklers historikk blir lagret i database, men dataene er kun tilgjengelig for bruker med administratorrettigheter. Kriterie1: Vise sykkelhistorikk for hver sykkel Kriterie 2: Mulighet for å fjerne / blokkere bruker fra systemet
Involverte elementer	WebGUI, Database, Kommunikasjon mellom sykkelsensor og system
Testoppskrift	

Brukerhistorie 07 – Talestyring	
Brukerhistorie	Som **SYKLIST** vil jeg kunne talestyre applikasjonen og få beskjeder fra appen med tale/lyd for å med enkelhet kunne sykle og bruke appen samtidig og uten hender for å bedre trafikksikkerheten.

Beskrivelse	Mulighet for å kontrollere sentrale funksjoner i appen Finn nærmeste sykkelstasjon Opplesning av rute veiledning (sving høyre om 200m osv) Stopp sykkelleie
Hvorfor/viktighet	Å kunne bruke nødvendige funksjoner i appen uten å bruke hendene eller ta øynene vekk fra trafikkbildet er kritisk for å kunne ha god trafikksikkerhet. Mobilbruk er forbudt i bil, og det bør også gjelde på sykkel Brukerhistorien er prioritert som Could Have, da funksjonen ikke er en grunnleggende funksjon av systemet, men noe som kunne vært med på å løfte brukeropplevelsen ytterligere. Det er også sjanse for at funksjonen blir av et mindretall av brukere.
MoSCoW	Could Have
PP-score (Fibonacci)	
Akseptansekriterie	Appen gir output om reiserute og annen info, og kan ta talekommandoer for noen/alle funksjoner.
Involverte elementer	
Testoppskrift	

Brukerhistorie 08	
Brukerhistorie	Som **SYKLIST** vil jeg kunne booke en sykkel på forhånd for å forsikre med om at jeg har en tilgjengelig sykkel fra A til B på det tidspunkt som passer meg. Jeg vil også at det skal være ledig parkering på min endestasjon.
Beskrivelse	Syklist skal kunne reservere en sykkel på forhånd dersom det er behov for å ha en sykkel tilgjengelig til et visst klokkeslett på en bestemt stasjon. Ved å implementere denne funksjonen kan flere mennesker ønske å benytte tilbudet siden de vet at det er sykler tilgjengelig når det trengs. Syklist må registrere både start og endestasjon for å sørge for at det er sykler tilgjengelig ved start og ledig parkering ved stopp.
Hvorfor/viktighet	Mange mennesker har avtaler og ting de skal rekke. Ved å booke sykkel på forhånd kan de være sikker på at en sykkel er tilgjengelig ved avreisestasjon, og at det er en ledig parkering på endestasjonen.
MoSCoW	Should have
PP-score (Fibonacci)	

Akseptansekriterie	Systemet registrerer Ledige sykler kan reserveres ved at systemet/stasjonen registrer reservasjonen og reserverer én av de tilgjengelige syklene. Når brukeren sjekker inn på stasjonen frigjøres en sykkel fra stasjonen og reservasjonen slettes. Dersom sykler må fylles eller tømmes i forbindelse med booking sendes et prioritert varsel til vedlikeholder.
Involverte elementer	Syklist, brukerGUI, stasjon, kommunikasjon, database, vedlikehold
Testoppskrift	1. Bruker reserverer sykkel i GUI 2. sjekk at forespørsel kommer frem 3. startstasjon reserverer én av de tilgjengelige sykler (eventuelt melder vedlikehold om behov for påfyll) 4. stasjon sørger for at det er minimum én reservesykkel til reservasjon (i tilfelle behov for reparasjon på reservert sykkel) 5. endestasjon reserverer en ledig parkering til booket sykkel (eventuelt melder vedlikehold om behov for tømming) 6. etter sykkel hentes slettes reservasjon og turen registreres som vanlig tur? 7. etter sykkel leveres slettes parkeringsreservasjon.

Brukerhistorie 09	
Brukerhistorie	Som **SYKLIST** vil jeg kunne hente ut sykkel uten å bruke appen, i tilfelle jeg har mistet telefonen eller er tom for batteri.
Beskrivelse	Dersom syklisten ikke har strøm på sin mobiltelefon, eller ikke eier en mobiltelefon skal det fortsatt være mulig å registrere seg og hente og levere sykler. Det må være mulighet for innlogging og registrering på hver stasjon gjennom et touchskjerm interface
Hvorfor/viktighet	Det må alltid være mulig å hente sykkel selv om brukeren ikke har telefonen sin tilgjengelig. Telefonen er et verktøy for å gjøre bruken at tilbudet så enkelt som mulig for syklisten
MoSCoW	Should have
PP-score (Fibonacci)	
Akseptansekriterie	Bruker uten tilgang til smarttelefon må kunne logge inn med brukernavn/brukerID og en 4-6siffer PINkode for å kunne hente ut sykkel. For å benytte seg av dette må syklisten ha en bruker registrert med et aktivt abonnement.
Involverte elementer	Syklist, stasjon, kommunikasjon, database

Testoppskrift	1. syklist taster inn ID og pin på skjerm ved stasjon 2. stasjon sender forespørsel om bekreftelse til databasen 3. hvis ok registreres turen i database som normalt og sykkel løses ut. 4. sykkel leveres ved endestasjon

Brukerhistorie 10 – Registrering og Innlogging	
Brukerhistorie	Som **SYKLIST** vil jeg enkelt registrere meg og logge inn i applikasjonen for å kunne ta i bruk bysykler
Beskrivelse	Brukere må ha mulighet til å registrere seg i systemet med: Navn (Fornavn, Etternavn) Telefonnummer Mail Adresse
Hvorfor/viktighet	Registering og lagring og innlogging er kritisk for å demonstrere prosjektet
MoSCoW	Must Have
PP-score (Fibonacci)	
Akseptansekriterie	Minimum: Brukeren kan registrere seg og logge inn gjennom app med database i skyen Utvidelse: 1. Brukeren logges inn gjennom sikker autentisering (2. Brukerbekreftelse på mail 3. Ved innlogging får bruker får beskjed om; • Bruker eksisterer ikke • Inntastet passord er feil 4. Ved registrering får bruker får beskjed om; • Bruker er allerede registrert • Feil data format / ugyldige tegn • Gjentatt passord er feil
Involverte elementer	Database, Databasekommunikasjon, Firebase Autentisering, App GUI,
Testoppskrift	

Brukerhistorie 11: Låse sykkel under tur	
Brukerhistorie	Som **SYKLIST** vil jeg ha mulighet til å parkere og låse sykkel frittstående dersom jeg skal handle på en butikk eller gjøre andre ærend mens jeg låner sykkelen fordi av og til må jeg handle
Beskrivelse	Det må være mulig for syklisten å sette fra seg sykkelen under leietiden.
Hvorfor/viktighet	Syklistene må ha mulighet til å kunne parkere sykkelen i lånetiden, med minimal risiko for at den blir stjelt eller utsatt for hærverk.
MoSCoW	Could have
PP-score (Fibonacci)	
Akseptansekriterie	En trygg løsning for å kunne sette fra seg sykkelen for å for eksempel handle eller spise lunsj. Alternativer kan være: Fastmontert kodelås/nøkkellås på sykkelen Fastmontert programmerbar elektronisk lås Frittstående kode/nøkkellås som henger på hver stasjon Syklist sørger for å eie og medbringe egen sykkellås
Involverte elementer	Syklist, sykkel
Testoppskrift	1. Gå for et alternativ 2. Skriv ny testoppskrift for valgt alternativ.

Brukerhistorie 12 – Betalingsløsning og Abonnent	
Brukerhistorie	Som syklist ønsker jeg mulighet for å legge til betalingsinformasjon og mulighet for forskjellige alternativer for sykkel-leie.
Beskrivelse	Mulighet for forskjellige alternativer for leie av sykkel: Abonnement; Årlig, Måned Timeleie Betaling kan utføres med: Kredittkort, Vipps
Hvorfor/viktighet	Fungerende betalingssystem er nedprioritert nå i prosjektfasen, men vil være kritisk ved et eventuelt endelig sluttprodukt.
MoSCoW	Should Have (Must have ved endelig system)

PP-score (Fibonacci)	
Akseptansekriterie	Brukere kan velge ønsket abonnement, årlig / måned Utføre betalinger og leie sykler gjennom kredittkort og Vipps Ved timeleie vil bruker få beskjed om gjenstående tid, og eventuell takst for utvidet tid.
Involverte elementer	Database, Kommunikasjon, Betalings API
Testoppskrift	

Brukerhistorie 13 – Protokoll for dataoverføring	
Brukerhistorie	Som **SYSTEM** ønsker jeg en lettvekts protokoll for overføring av alle sensordata fra sykkelModulen til databasen, for å minimere datastørrelsen som sendes og dermed spare mest mulig batteri i enheten
Beskrivelse	En overføringsprotokoll som er så slank som mulig og overfører kun de nødvendige dataene i et forhåndsdefinert format som leses og forstås på serversiden.
Hvorfor/viktighet	Vi trenger at data overføres fra sykkelen til databasen. Samtidig trenger vi å spare mest mulig batteri. Dermed ønsker vi å skrive en egen slank protokoll som minimerer datamengden og dermed sparer strøm på sykkelenheten
MoSCoW	Must have
PP-score (Fibonacci)	
Akseptansekriterie	Sensordata i sykkelmodulen registreres og lagres i et forhandsdefinert format. Dette formatet sendes via NB til server som leser tolker og registrerer data i databasen. Denne dataen kan deretter fremvises i AdminGUI.
Involverte elementer	SykkelGPS, sykkelmodul, kommunikasjon, database, adminGUI
Testoppskrift	3. Generer data/testdata fra ArduinoNB-enhet 4. Data skrives i protokollen etter predef format 4. Data overføres til server og database 5. Data hentes til og leses av adminGUI

Brukerhistorie 14 – Vise sykkelrute i kart	
Brukerhistorie	Som **SYKLIST** ønsker jeg å kunne få en foreslått rute i kart basert på min start og endestasjon for å få raskeste/beste/fineste reiserute til min destinasjon.
Beskrivelse	Hvis du er ukjent i byen eller vil ha raskeste eller mest gunstige rute basert på hvilken vei tidliger syklister har tatt før deg. Denne ruten / start- og sluttpunkt kan genereres av vårt system basert på våre data. For enkelhets skyld bør denne ruten kunne eksporteres til google maps, for å holde vår egen app så slank og velfungerende som mulig.
Hvorfor/viktighet	Man vil komme fra A til B og vite hvor man skal sykle
MoSCoW	Could have
PP-score (Fibonacci)	
Akseptansekriterie	Fra og tilpunkter (eller komplett rute)defineres i vår app. Disse data sendes til google maps hvor bruker får oppgitt rute og veibeskrivelse.
Involverte elementer	Syklist, app, kommunikasjon?, database?, GOOGLE MAPS
Testoppskrift	1. syklist setter start og sluttstasjon/punkt i vår app (kart eller inntasting/velg fra liste/favoritter) 2. vår app sender disse dataene til google maps (app) 3. google maps (app) viser veien.

Brukerhistorie 15 - Poengsystem	
Brukerhistorie	Som **SYKLIST** ønsker jeg å kunne motta poeng(credits) når jeg parkerer sykkelen på mindre populære stasjoner.
Beskrivelse	Mindre populære stasjoner får sjeldent tilbake syklene etter bruk, derav blir disse stasjonene fort tomme. For å forminske behovet for etterfylling av sykler, gir vi syklistene et insentiv for å sette tilbake syklene på disse stasjonene. Poengene syklistene mottar blir brukt til gratismåneder på betaling av abonnementet.
Hvorfor/viktighet	F. eks sykkelstasjoner som er i en oppoverbakke blir syklet ned bakken, men parkert på en annen stasjon. For at vi skal minske ressursbruken til vedlikeholder på etterfylling av stasjonene, bruker vi syklistene til å gjøre dette for oss.
MoSCoW	Could have

PP-score (Fibonacci)	
Akseptansekriterie	Stasjonen registrer sykkelID som er knyttet til syklisten som bruker den, og applikasjonen gir ut poeng basert på hvor populær stasjonen er.
Involverte elementer	Syklist, app, database, kommunikasjon
Testoppskrift	1. Syklist parkerer sykkelen på en stasjon 2. Applikasjon deler ut poeng til syklisten basert på popularitet av stasjon 3. Applikasjon kommuniserer med database og registrer poeng til syklisten

Brukerhistorie 16 – Lån flere sykler i gangen	
Brukerhistorie	Som **SYKLIST** ønsker jeg å kunne låne en ekstra sykkel dersom jeg har besøk av familie eller venner som er på besøk og som ikke ønsker å opprette en bruker i systemet, siden de kun skal sykle én eller to ganger.
Beskrivelse	En enkel måte å kunne sykle sammen med din venn som er på besøk fra utenbys. «Lån en ekstra sykkel» kan være inkludert i et premium-abbonement, eller at «vanlige brukere» eller brukere med månedskort betaler en ekstra sum, belastes dobbelt på et eventuelt klippekort, belastes i reward credits eller i kreditt som trekkes på neste betaling. Eventuelt kan gjesten betale direkte med vipps ved å skrive inn brukerID på personen med bruker i systemet og angitt sum.
Hvorfor/viktighet	Dersom du bare er på besøk en dag eller en helg og ønsker å benytte deg av bysykkeltilbudet kan din venn med bruker i systemet enkelt hente ut en ekstra sykkel mot ekstra betaling eller belasning på sin konto. Det er fordelaktig fordi 1. Det går raskere og enklere enn å registrere en ny bruker 2. Betaling for leie av ekstra sykkel går gjennom eksisterende bruker 3. Vi unngår registrering av engangs- og inaktive brukere i databasen (plassbesparende, mer oversiklig og ryddig DB)
MoSCoW	Could have
PP-score (Fibonacci)	
Akseptansekriterie	Syklist registrerer at hen ønsker å låne en ekstra sykkel. Systemet behandler forespørselen og klipper/trekker automatisk et ekstra beløp for tjenesten, eller tar imot vippsbetaling med registrering av brukerID

	til registrert bruker. To sykler løses ut og registreres i systemet som to enkeltstående turer.
Involverte elementer	Syklist, app, kommunikasjon, database
Testoppskrift	1. Bruker registrer ønske om å låne ekstra sykkel 2. Systemet registrerer og behandler forespørsel 3. Korrekt betaling/belastning registreres 4. To sykler løses ut og registreres som to enkeltstående turer 5. Sykler leveres og turene avsluttes

Vedlegg 8: Møtereferat Styringsgruppemøter

Styringsgruppemøte 1, 27. februar 2018

Møtereferat

Den 27. Februar 2018 ble det holdt styremøte hos Egde i Grimstad.

Til stede: Brynjar Blomvik, Elin Kilen (telefon), Frode Leonhardsen, Hallgeir Nilsen, Gisle Stavland, Erlend Kasin Østern,
Fraværende: Delvis Trond Kvarenes

Sak 1. Gjennomgang av prosjektsstatus

Det ble gitt tilbakemelding på prosessen så langt, vi ble enige med Egde om at vi skal legge mer vekt på risiko og avvik fremover. Til neste periode skal vi ta i bruk risikomatrise. Mal på dette skal ligge i vårt skolesystem(Canvas), eventuelt kunne vi få en mal fra Gisle.

Sak 2. Mål for oppgave

Vi ble enige om å sette en mer konkret ramme for hva oppgaven skal handle om. Sånn status var nå, var oppgaven litt for stor i sin helhet, så vi skal klargjøre rammen for prosjektet vårt. Vi ble enige om at vi skal prøve å estimere hvor lang tid vi bruker på ulike oppgaver, for så å justere oss etter dette i fremtiden. Etter litt ustrukturert hverdag hittil, kom vi frem til at vi skal bruke daily scrum fremover, og velge en prosjektleder til å kontrollere dette.

Sak 3. Sprint Demo

Egde og veileder var fornøyde med grensesnittet så langt på UI mockup, men det ble kommentert at vi manglet registrering betalingsinfo ved registrering av ny bruker. Vi kunne også endre visning av tilgjengelige sykler i applikasjonen, siden det ikke er interessant for brukerne å vite hvilke sykler som ikke er tilgjengelige.

Sak 4. Diskusjon om funksjoner

- Credits for belønning av bruker som bidrar til effektiviserende drift.
    - Bruker setter av sykler til mindre populære stasjoner
- Sykkel blir brukt som innsamlingsobjekt av data, trenger derfor ikke vise dette i applikasjon.
- Applikasjon kan heller muligens ha weather forecast.
- Applikasjon kan også inneholde geodata rettet mot turister
- Innsamling av data til andre formål enn det som er sykkelrelevant.

Sak 5. Neste møte

Vi skal ha en kontordag hos Egde mandag, 5.mars. Der vi skal få veiledning på vår oppgave, og få svar eventuelle spørsmål vi har.

Neste styremøte: Dato er ikke satt, men rundt uke 14.

Egde Consulting / UiA Flåtestyringsprosjekt

Kl. 12:30
14. mai 2018

Beskrivelse
Styringsgruppemøte 2
Kommunikasjon: Fysisk oppmøte + Skype

Deltagere
Gisle Stavland
Elin Kilen
Erlend Kasin Østern
Frode Leonhardsen
Brynjar Blomvik
Hallgeir Nilsen (Skype)

Ikke tilstede
Trond Kvarenes

Agenda

1. Gjennomgang av tidligere møter, og status rundt fremgang i prosjektet
    - Styringsmøtemøtereferat (27. februar 2018)
    - Forrige møte mellom Egde og studenter (27. april 2018)
2. Problemer og utfordringer i utviklingen
    - Arduino/NB-IoT
3. Videre utvikling og nye funksjoner
4. Spørsmål
5. Eventuelt

Gjennomgang av agenda

Sak 1: Gjennomgang av tidligere møter, status for prosjektet
Mye har endret seg siden første styringsmøte, under forrige møte var det mindre klare linjer for hva vi skulle utvikle.

I løpet av de siste møtene med Egde har vi justert mer fokus på rammeverk for flåtestyring og innsamling av data

Visualisering av data skal gjøres gjennom verktøyet Power BI

Selv om oppgaven og fokus har endret seg, har det vært lærerikt og det representerer hvordan ting ofte fungerer i arbeidslivet.

Sak 2: Problemer og utfordringer
Vi har hatt en del tekniske utfordringer med iot enheten. Vi har ikke fått til å koble den på nett via Telias narrowband nettverk. Dette skyldes at vi mangler en del informasjon for å koble oss til Tela sitt nettverk. Kortet kommer er fra en nederlandsk produsent, så det ser ut til at dette ikke er 100% kompatibelt med Telia sitt nettverk.

Egde foreslår å simulere data, og heller legge IoT kortet på is.

Telia har sine egne devkit som vi kunne ha prøvd om vi skulle fortsatt med videre utvikling

Diskusjon rundt rapporten:
Rapporten og andre fag har tatt mye fokus de siste ukene, dette har forårsaket at utviklingen har fått lite fremgang. Dette ble varslet om på forrige møte med Egde.

Elen rådet om at vi skulle beskrive detaljer rundt tilkoblingen i rapporten, flaskehalsen det skapte, hva man kunne gjort annerledes.

Elin foreslår videre at vi ikke bruker for mye tid på flaskehalser, sett fokus på ting som er proof of concept for systemet.

Problemer rundt prosjektstyring:
Vi som gruppe har hatt problemer med å forholde oss til Scrum rammeverket og Jira. Teknisk utfordrende flaskehalser og jevnlige endringer har holdt igjen mye av utviklingen, som har gjort at fokus på å oppdatere framgang har blitt falt litt bort. Vi har endt opp med et reelt prosjekt rammeverk som ligner mer på Kanban enn Scrum.

Sak 3: Videre utvikling og nye funksjoner

Etter rapporten er levert starter siste innspurt av utviklingen, dette skal vi fokusere på:

- Hente posisjon fra system og vise det i applikasjon
- Bruke mobil app til innsjekk og utsjekk kjøretøy
- Lage et enkelt webgui med historikk over reiser / PowerBI
- Rapportere status fra kjøretøy

Sak 4: Spørsmål/diskusjon
I siste innspurt av prosjektet foreslår Gisle å se på og diskutere fordeling av resterende oppgaver.

Lag en plan for siste utviklingsfase

Sak 5: Eventuelt

Ingen

Nye saker
Se sak 2: problemer og utfordringer

Avslutningsvis
Møtet ble avsluttet *(ca.)* kl. 13.10

Referert av: Frode Leonhardsen

Godkjent av: Brynjar Blomvik

Vedlegg 9: Møtereferat øvrige møter

MØTEREFERAT 3 06 FEB
GRIMSTAD

Sak 1: Saker av interesse

- Flåtestyring, Vedlikehold, sensordata
- Lande på bruk av sensor. Egde undersøker Raspberry Pi/Arduino
    - Finne ut hvilke sensorer som trengs
- Logistikk og system rundt rapportering av data
    - Realtime? Hvor/hvilken sensor/enhet skal rapportere og når?

Sak 2: Planlegging av Scrum:
Vi ble enige om følgende:

- To ukers sprint
- Sprint start slutt og demo på tirsdager

Sak 3: Kriterier for prosjektet

- Proof of concept (funksjonalitet over (grafisk) estetikk)
- Hvordan skal vi jobbe (Scrum)
- Teste metodikk
- Skalerbar/fleksibel løsning (1-5000-10'000 sykler)
- Høy mobilititet – undersøke og vurdereteknologivalg
- Integrasjon av løsning
- Skyløsning – kjent/lett å bruke (Google Firebase)
- Overførbarhet til andre felt(gjenbrukbarhet)
- Scrum administrasjonsverktøy: Jira
- Brukerundersøkelser (Spørreskjema)
- Brukertest av appen
    - Hvordan gjorde vi testen
- Undersøke eksisterende løsninger
- Innsamling av data
    - Posisjonsdata
        Eksempler:
        SmartebyerNorge nettverk
        AKT – GPSdata på buss
- Temperatur/luftkvalitet
    - Stavanger open.stavanger.kommune.no

Meshtech: potensiell leverandør av sensor/samarbeidspartner flåtestyring

Sak 4: Epics
Epos/epics-nivå
Hovedfunksjonksjonalitet

1. flåtestyringen
    a. Bruker sykkel(use cases)
    b. Stasjonsforvaltning(use case)

c. Apputvikling
2. Vedlikehold av flåten/syklene
    a. Vedlikehod (use case)
3. Datainnsamling/visualisering
    a. (use case)

Sak 5: Styringsgruppemøte

Få mal for styringsgruppemøter fra Egde med liste over roller

Sette roller

- Styringsgurppeleder
- Senior user/salgsperson
- Senior supplier

Sak 6: eventuelt

Forslag/tanker:

Vedlikeholdsbehov på kjøretøy / sjekk av tilstand

Sjekk tid siden sykkelen ble sist brukt → kanskje den er i dårlig stand?

Forslag til turer/ruter: sightseeing for turister. Turstier / ruter / severdigheter

MØTEREFERAT 13. mars

Sak 1: Nye funksjoner til systemet
Betalingsløsning for brukere
- ulike alternativer

Overføringsprotokoll
- MQTT
- Mulighet for bruk av Raspberry Pi som MQTT-broker

Sak 2: Fremtidige møter
Mulighet for å avholde møter på skype
Flytte møtedag fra tirsdag til onsdag eller torsdag. Grunnen er at Elin har fri på tirsdager.

Møtereferat

Den 11. April hadde vi et møte med Egde via Skype.

Til stede: Brynjar Blomvik, Erlend Kasin Østern, Trond Kvarenes

Fraværende: Frode Leonhardsen, Elin Kilen, Gisle Stavland

Sak 1. Databasestruktur

Vi ble enige om hvordan databasen burde struktureres, at vi burde ha all sensordata i en tabell, som skal referere til en sykkeltabell. Vi snakket også om hvor hyppig data skulle sendes til databasen. Vi trenger ikke sende live data, men heller kanskje sende hver halvtime.

Sak 2. Analyse

Vi snakket om hvordan vi skulle analysere dataen og kom til enighet om å bruke Power BI, og at Egde skulle stå for lisenser for dette om det var nødvendig.

Sak 3. Sprintdemo

Vi hadde ikke så mye nytt teknisk å vise, men det skal vi ta med neste gang for å gi Egde et innblikk i hvor vi ligger an per dags dato. Men vi ble enige om å prioritere flåtestyring og strukturering av data fremover.

Sak 4. Neste møte

Neste Sprintdemo-møte blir holdt 25. April, der vi skal vise hva vi har så langt. Vi snakket også om å ta et møte over Skype onsdag, 18. April.

Egde Consulting / UiA Flåtestyringsprosjekt

Kl. 10:00
25. april 2018

Beskrivelse
Sprint- og demomøte mellom oppdragsgiver (Egde Consulting) og utviklerteam (bachelorstudentene)
Kommunikasjon: Skype videosamtale

Deltagere
Trond Kvarenes
Elin Kilen
Brynjar Blomvik
Frode Leonhardsen

Ikke tilstede
Gisle Stavland
Erlend Kasin (sykdom)

Agenda
- Sak 1: Demonstrasjon av systemet MQTT/SQL/Java
- Sak 2: Dataanalyse – Power BI
- Sak 3: Eventuelt

Innledning
Skype fungerte ikke som ønsket. Det resulterte i at vi (studentene) ikke kunne kommunisere med Trond eller Elin samtidig. Møtet ble i stor grad brukt til feilsøking i Skype/Skype for business. Etter at Elin måtte forlate oss hadde vi en kort oppsummering med Trond.

Gjennomgang av agenda
Sak 1: Demo
Vi opplevde store problemer med å få Skype til å fungere med gruppesamtale. Skype for business viste seg å være problematisk.
Mulig løsning: Alle deltagere bruker "vanlig" Skype konto. Vi testet med Trond Frode og Brynjar, det fungerte med lyd video og skjermdeling
Grunnet de tekniske problemer og tidsbegrensninger utsettes demo til fredag 27/04/18.
Elin kommer tilbake med klokkeslett for møtet.

Sak 2:Dataanalyse
Vi setter i gang med bruk av Power BI. Trond kan sette studentene i kontakt med folk i Egde som har erfaring med bruk av Power BI.

Sak 3: Eventuelt
Kort diskusjon angående flåtestyringsbiten. Mer om dette på neste møte

Nye saker
Ingen nye saker

Agenda for neste møte
- Gjennomføring av demo
- Flåtestyring:
    - Egdes ønsker
    - Design/funksjonalitet
- Power BI – hva og hvordan
- Eventuelt

Avslutningsvis
Møtet ble avsluttet (ca.) kl.10.54. Neste møte blir på Egdes kontor i Kristiansand 27.04.18. Elin undersøker klokkeslett og melder alle involverte parter.

Referert av: Brynjar Blomvik

Godkjent av: Frode Leonhardsen

Egde Consulting / UiA Flåtestyringsprosjekt

Egde Consulting, Gravane Kristiansand
27. april 2018

Beskrivelse
Demo av systemet og ønske om videre funksjonalitet
Kommunikasjon: Fysisk 3D + Trond på Skype

Deltagere
Elin Kilen
Trond Kvarenes (Skype)
Brynjar Blomvik
Frode Leonhardsen

Ikke tilstede
Gisle Stavland
Erlend Kasin (Sykdom)

Agenda
- Sak 1: Demo
- Sak 2: Videre funksjonalitet/flåtestyring
- Sak 3: Dataanalyse
- Sak 4: Prioritering av oppgaver

Innledning
Møtet foregikk på Egdes kontorer i Kristiansand. Etter hils og rombytte gikk vi i gang med dagens agenda.

Gjennomgang av agenda
Sak 1: Demo
Demo gikk smertefritt og som planlagt. Alle parter fornøyd med funksjonaliteten så langt.

Sak 2: videre funksjonalitet/flåtestyring
Av funksjonalitet i flåtestyringen ønskes følgende

- Se posisjon/status/øvrig på kjøretøy i sanntid.
    - Hvem har sjekket ut spesifikt kjøretøy.
    - Hvor finnes det ledige kjøretøy.
- Et register over tidligere reiser/turer.
- Vedlikehold og administrativ logistikk.

    - Hvilke kjøretøy trenger ettersyn/reparasjon.
- Grensesnitt for administrasjon.
- Bruke mobilapp til utsjekking av kjøretøy.
    - QR-kode som enkel implementasjon.

Sak 3: Dataanalyse
Av dataanalyse ønskes følgende:

- Benytte Microsoft PowerBI. Trond kan sette oss i kontakt med Gunnar i Egde som har erfaring med bruk av PowerBI.
- Fremvisning av GPS-data i kart – analysere reisemønster.
- Vise temperatursvingninger over tid.
- Bruksmønser for utleie – Når er det størst pågang? Rushtider?

Sak 4: Prioritering av oppgaver
Vi diskuterte hvorvidt vi videre skal prioriteres flåtestyring eller dataanalyse. Vi kom frem til at en del av funksjonaliteten i de forskjellige retningene baserer seg på den samme funksjonaliteten (uthenting, behandling og fremvisning av sensordata). Vi vil derfor prioritere å få på plass den funksjonaliteten som er nødvendig i begge tilfeller.

Nye saker
Neste styringsmøte
Vi diskuterte tidspunkt for neste styringsmøte med vår studieveileder.
De mest gunstige tidspunkt for Egde er enten 7. eller 14. mai.

Agenda for neste møte
- Styringsmøte med studieveileder. Tidspunkt er ennå ikke satt.
- Status for prosjektet og veien videre.
- Status på Arduino/NB IoT
- Eventuell demonstrasjon av ny funksjonalitet.

Avslutningsvis
Møtet ble avsluttet ca. kl.12.35. Neste møte blir styringsmøte med studieveileder Hallgeir Nilsen. Møtet holdes etter all sannsynlighet på Egdes kontorer i Grimstad. Foreslått dato er enten 7. eller 14. mai.

Referert av: Brynjar Blomvik

Godkjent av: Brynjar Blomvik

Vedlegg 10: Selvevaluering

Gruppeevaluering

Gruppens samarbeid har fungert godt gjennom semesteret, og alle gruppemedlemmene har vært tilstede og bidratt til prosjektets fremgang. Vi har operert uten faste daglige oppmøtetider da flere av gruppens medlemmer kombinerer bachelorstudiet med deltidsarbeid og andre forpliktelser. Vi har stort sett arbeidet individuelt, men har hatt løpende kommunikasjon over internett i tillegg til å møtes ukentlig for å kunne jobbe sammen og å oppdatere hverandre på status.

Alle medlemmene har tidligere erfaringer og styrker på forskjellige felt, og dette har vi utnyttet i arbeidsfordelingen. (Medlemmer med erfaring fra Android-utvikling har fått ansvaret for appen, Javabakgrunn, serverlogikk osv.) Selv om vi har erfaring med noen av teknologiene vi har benyttet oss av har det gått mye tid til å gjøre research på teknologi vi benytter oss av for første gang.

Sosialt har gruppen også fungert veldig godt. To av medlemmene har jobbet sammen og kjenner hverandre fra før, og de har tatt imot det nye medlemmet med åpne armer og et varmt sinn.

Vedlegg 11: Individuell evaluering

Brynjar Blomvik

Jeg har i utviklingsdelen av prosjektet vært ansvarlig for utvikling av java kildekode, database og java-kommunikasjon og nettverkskommunikasjon via MQTT. Jeg har ikke hatt tidligere kjennskap til MQTT, så her har det gått med mye tid til å utforske ulike alternativer for å finne beste praksis i vårt tilfelle. I tillegg har jeg hatt ansvaret for research og utvikling av Arduino og NarrowBand-teknologien.Sistnevnte har jeg dessverre ikke kommet i mål med, men det er noe jeg personlig finner svært interessant og ønsker å lære mer om. På den administrative siden har jeg tatt på meg det meste av dialogen med oppdragsgiver, som inkluderer planlegging av møtevirksomhet.

Frode Leonhardsen

Under prosjektet har jeg hatt flere allsidige roller, og fått prøvd meg på mange lærerike, forskjellige og utfordrende arbeidsoppgaver. I startfasen hadde jeg hovedansvar for utvikling

av app-design, utforming og prototyping, da dette var noe jeg hadde erfaring med fra før og en generell interesse for. Dette medførte å tolke brukerhistoriene fra detaljerte beskrivelser til visuelle skjermbilder som vi kunne bruke som mål og retningslinjer under den faktiske utviklingen. Videre i utviklingsfasen tok jeg på meg hovedansvaret for utviklingen av Android applikasjonen. Dette var noe som var helt nytt for meg og som har vært veldig lærerikt, da jeg har lært mye om hvordan Android applikasjoner er strukturert og om hvilke funksjoner som er innebygget og kan brukes gjennom Android SDKen. Jeg hadde også ansvar for oppsett av VPS og installasjon / konfigurering av LAMP webstacken. Etter at denne var konfigurert jobbet jeg med utvikling av backend logikk med å lage Rest APIer i PHP for å kunne håndtere data til og fra database mot applikasjon. Dette var en tidkrevende utfordring da PHP og Restful var teknologier jeg ikke hadde kjennskap til fra tidligere. Forøvrig så har jeg jobbet tett med de andre gruppemedlemmene rundt de fleste andre oppgavene i prosjektet, som f.eks planlegging, analyse, brukerhistorier, use-cases, utforming av database, rapportskriving etc.

Erlend Kasin Østern

Mitt arbeid i dette prosjektet har vært rettet mot databasen, men jeg har også jobbet med de «generelle» oppgavene som resten av gruppen har jobbet med, som blant annet brukerhistorier, planlegging og analyse og arbeid med rapporten. Jeg var også ansvarlig for risikoanalysen. Jeg har vært innom litt forskjellige aspekter i prosjektet og lært mye nytt om blant annet javakommunikasjon og MQTT. Jeg har ingen erfaring med bruk av sensor fra før av, så det var noe jeg syntes var spennende.