UNIVERSITETET I AGDER

Forside
IS-304: 2017

Tittel:

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

Studenter:

Etternavn	Fornavn
Flovik	Sondre
Moudnib	Robin Amir Rondestvedt
Omland	Ricky Løtoft
Thorne	Christian Fredrik
Zetterquist	Erik Oskar

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 bruker til undervisningsformål?	JA X	NEI___
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	JA X	NEI___

Bachelor-report

OSecurity

Av “Salt Mine Workers”

GitHub Android: https://github.com/Robinron/OSecurityApp
GitHub Raspberry: https://github.com/sflovik/OSecurity
Om OSecurity: http://sondrf14.wixsite.com/osecurity

Sondre Flovik

Robin Amir Rondestvedt Moudnib

Ricky Løtoft Omland

Christian Fredrik Thorne

Erik Oskar Zetterquist

Forord

Formålet med denne rapporten er å belyse prosjektgjennomføringen vår i forbindelse med faget IS-304 (Bacheloroppgave i Informasjonssystemer) og de sentrale elementene i prosjektet. Bakgrunnen for akkurat dette prosjektet er at vi har vært gjennom et innovasjonsløp i samarbeid med både Institutt for Informasjonssystemer og UiA Nyskapning samt Ungt Entreprenørskap.

Vi ønsker gjerne å takke følgende støttepersoner og organisasjoner for deres bidrag til gjennomførelsen av vår Bacheloroppgave:

Ungt Entreprenørskap v/ Vemund Ruud: for å tilrettelegge og veilede oss gjennom oppstarten av vår studentbedrift; OSecurity SB.

UiA Nyskaping v/ Kristina Maria Walker-Nordlöf for veiledning og invitasjoner til arrangementer tilknyttet UiA Nyskaping.

CoWorx v/ Terje Klungland - For å stille seg til disposisjon ved lokaler og arrangementer vi har fått deltatt på.

Janis Gailis, Universitetslektor, Institutt for Informasjonssystemer UiA for god veiledning med verdifull input.

Hallgeir Nilsen - Førstelektor, institutt for informasjonssystemer UiA - for godkjenning av et utradisjonelt prosjekt og støtte underveis.

EGDE Consulting v/ Gisle Stavland - For å ha hjulpet oss direkte med ressurser som vi kunne bruke i vårt prosjekt, i regi av konkurransen gruppen vår vant på Hack4Universitetsbyen - Hackathon. Derav ble vi også invitert med på arrangementet EGDE-treff, som har bidratt til å styrke samholdet i gruppen og gjort at vi har fått kontakt med næringslivet.

Bachelorgruppen “Familekalender” v/ Jørgen Lybeck Hansen som har vært en sparringspartner som har gitt oss tips og tilbakemeldinger samt gjennomført kodekveld sammen hvor det ble gjennomført brukertesting.

Sammendrag

Dette er vår rapport for bacheloroppgaven vår i Informasjonssystemer.
Gruppen hadde sin opprinnelse i 2016 i forbindelse med et annet emne. Vi består av følgende personer: Sondre, Robin, Ricky, Christian Fredrik og Erik Oskar.
Noe av det som gjør vår bacheloroppgave unik er at vi har vært vår egen oppdragsgiver, og dermed har gjennomført oppgaven for bedriften vi selv etablerte.
Vår gruppe og bedrift har produsert et modulært sikkerhetssystem basert på Raspberry Pi som hovedsakelig skal brukes til å overvåke ens eget hjem og eiendom ved hjelp av sensorer og kamera gjennom en Android-applikasjon, men systemet har også et stort videreutviklingspotensiale.

I tillegg til selve utviklingen av programvare på sikkerhetssystemet og Android plattform, har vi også vært aktive i forhold til gründervirksomhet, business-aspektet og produktutvikling. Backloggen vår har vært bestående av brukerhistorier som er prioritert ved bruk av MoSCoW teknikken, tatt ut i sprint backlogg med mindre deloppgaver og så estimert i timer og kompleksitet ved bruk av Planning Poker.

Vi har produsert prototyper for planlagt grensesnitt og kravspesifikasjon som forteller hvilke teknologier vi skal bruke og hvilke krav som stilles til produktet. Deretter ble det laget en forenklet arkitektur over systemet, og videre går vi inn på teknologi og stack før det går mer i dybden omkring skytjenester, IoT og sikkerhet rundt disse teknologiene.

Implementasjonen og den tekniske gjennomførelsen utført ved hjelp av flere skytjenester i form av tjenester fra både Amazon Web Services og Google Firebase. Mobilapplikasjonen ble utviklet i Java i Android Studio, mens Raspberry Pi-implementasjon ble gjort i Python.
Prosjektstyring og arbeidsplanlegging ble gjort etter Scrum-rammeverket for de forskjellige sprintene, hvor planleggings-, ”review”- og ”retrospekt” møter ble gjennomført ved de respektive begynnelsene og avslutningene av sprintene. Dette ble kontinuerlig fulgt opp av Scrum Master som tilså at timelogging ble gjennomført og at prosjektet var i rute i henhold til prosjektplanen vi fulgte.
Ved endt bacheloroppgave er resultatet en fungerende prototype som tilfredsstiller arbeidsgivers krav iht. et minsteverdiprodukt (MVP).

Innholdsfortegnelse

Figurliste

1 Introduksjon

1.1 Gruppen - SaltMineWorkers

SaltMineWorkers opprinnelse startet i august 2016 i forbindelse kurset IS308 - Internetteknologier. Intensjonen med denne gruppedannelsen var å prøve ut om et potensielt gruppesamarbeid ville fungere. Vi konkluderte dermed etter et vellykket samarbeid i dette emnet at vi ønsket å samarbeide på Bacheloroppgaven. Dette spesielt på bakgrunn av hvor god kjemi vi har i gruppen, samt at våre ulike kvaliteter og interesser utfyller hverandre og gjør oss til det vi selv anser som en sterk gruppe.

1.2 Gruppemedlemmer

Sondre Flovik

Har interesse for hele systemutviklingsprosessen, med hovedfokus på teknisk utførelse og implementasjon. Håndterer godt Java, Python og skytjenester, og har erfaring med Android-utvikling. I prosjektet har jeg hovedsakelig hatt et overordnet fokus på design og analyse, og hovedfokuset har vært rettet mot implementasjon og den tekniske løsningen. I og for bedriften, OSecurity, har jeg hatt rollen og ansvaret som styreleder.

Robin Amir Rondestvedt Moudnib

Jeg anser meg selv som en strukturert person, så jeg er derfor veldig glad i prosjektstyringen innen systemutvikling. Samtidig ønsker jeg å danne et sterkt teknisk grunnlag, og har spesialisert meg innen databaser, Java, Python og distribuerte systemer. I prosjektet har jeg vært Scrum Master og hatt ansvar for prosjektstyringen, samtidig som jeg har utviklet både på Android-applikasjonen og Raspberry Pi'en. I og for bedriften, OSecurity, har jeg hatt rollen og ansvaret som daglig leder.

Ricky Løtoft Omland

Interesserer meg for det tekniske aspektet rundt systemutviklingen, syntes arkitektur, sikkerhet og nettverk er interessant og liker å se hvordan systemer bygges opp. Brukeropplevelse (UX) og brukergrensesnitt er også deler av prosessen jeg har opparbeidet meg en del kompetanse og interesse for.

Christian Fredrik Thorne

Bakgrunn fra IKT-servicefag, jobbet mye med drift. Interesserer meg i hovedsak for design, hardware og arkitektur. Utover disse har jeg fått prøvd meg på tekniske

aspekter rundt prosjektet vårt, noe som har medført at jeg har fått økt forståelse og kompetanse innenfor de områdene jeg føler meg svakere på. Videre har jeg hatt ansvar for produktutvikling i OSecurity.

Interesserer meg i hovedsak for design, hardware og arkitektur. Utover disse har jeg fått prøvd meg på tekniske aspekter rundt prosjektet vårt, noe som har medført at jeg har fått økt forståelse og kompetanse innenfor de områdene jeg føler meg svakere på. Videre har jeg hatt ansvar for produktutvikling i OSecurity.

Erik Oskar Zetterquist

Interesserer meg for systemutvikling og spesielt for de forretningsmessige aspektene ved systemutvikling og det er nok nettopp her jeg får vist mine sterke sider. Selv om jeg ikke er den mest tekniske, har jeg en grunnleggende forståelse av Java, Python og database utvikling. Jeg ser på meg selv som en strukturert og pliktoppfyllende student. Jeg har rollen som regnskapsfører i student bedriften OSecurity.

1.3 Klienten - OSecurity SB

I dette Bachelorprosjektet har vi ikke hatt en ekstern arbeidsgiver, snarere har vi vært vår egen arbeidsgiver ved at vi dannet en studentbedrift i regi av Ungt Entreprenørskap. Vi ønsket selv å erfare hvordan det ville være å starte en små-skala teknologi-bedrift i Norge, vi mente at dette ville ha et stort læringspotensial og ettersom det er mye fokus på gründervirksomhet og entreprenørskap på Universitetet og Sørlandet generelt så vi på dette som en ypperlig mulighet. Vi ser også på dette som et potensial for videreføring av bedrift til et etablert selskap i fremtiden.

1.4 Hvorfor dette prosjektet

IoT er et voksende konsept som vi har hatt svært mye teoretisk tilnærming til i tidligere emner på studiet. Ettersom vi hadde en del kunnskap fra før av om potensialet til Internet of Things, visste vi at dette er en trend som har et enormt potensial både i forhold til nytteverdi men også et stort markedspotensial om vi ønsker å ferdigstille produktet for salg. Selve sikkerhetssystemet OSecurity er noe vi har jobbet med siden 2. året på studiet, dette har derfor gitt oss et fundament å bygge videre på, selv om vi i prinsippet bygger om hele kodebasen. Prosjektet er også viktig i og med at vi skaper et system som har til hensikt å være nyttig for kunde/forbruker i forhold til å skape en rimelig sikkerhetsløsning.

Hvor omfattende vi kan gjøre prosjektet er egentlig bare begrenset av ressurser, kapasitet og fantasi. Det har et tilnærmet ubegrenset videreutviklingspotensial

dersom vi skulle ønske å utvide til andre formål, eller bare å videreutvikle sikkerhetssystemet. Eksempler kan være smarthusmoduler, system for å måle verdier (gassmåler, røykvarsler, pH-verdier). Det er også et spennende prosjekt ettersom det er en lærerik prosess med tanke på alle aspektene som involveres. Vi har eksempelvis måtte sette oss inn i applikasjonsutvikling for mobil plattform i en skyløsning (Amazon Web Services) og å ta stilling til problemstillinger som er nye for oss, som: Hvordan kan vi garantere at informasjon ikke kommer på avveie med tanke på at vi håndterer overvåkningsdata. Andre sikkerhetsaspekter man må ta hensyn til involverer potensielt misbruk av dataen vi prosesserer. Her kommer sikker overføring, kryptering og nettverksprotokoller som MQTT inn i bildet.

1.5 Hva vi skal produsere

OSecurity er et modulært, rimelig sikkerhetssystem som skal kunne skreddersys til brukerens behov. Systemet baserer seg på Raspberry Pi, en mini-datamaskin som fungerer som en alarmsentral. Sentralen styres av ulike Python-scripts som behandler signaler fra PIR (bevegelse) sensorer og et kamera for overvåkning av brukerens eiendom. Systemet skal kunne styres fra hvor som helst ved hjelp av en Android applikasjon som vil være tilgjengelig for mobile enheter, så lenge den er tilknyttet internett.

Kompleksiteten og funksjonaliteten til sikkerhetssystemet vil variere etter pakkeløsninger som vi har satt opp, alt fra en grunnleggende bevegelsessensor, til en mer kompleks løsning som inkluderer magnetsensorer for dører og vinduer, kameraovervåkning og ansiktsgjenkjenning.

Systemet vil inneha grunnleggende funksjonalitet som sørger for at sensorene oppdager innbrudd, som igjen vil formidles til mobilapplikasjonen. Brukeren skal kunne velge å se en kamerastrøm i sanntid for å ha kontroll over sin eiendom. Kameraet vil kunne innstilles til å ta opp en gitt sekvens av bilder og video over tid, slik at brukeren hele tiden kan kunne gå tilbake i tid og se hva som har skjedd, dersom noe har hendt. Systemet vil primært kobles opp til internett via WiFi eller Ethernet for å kunne kommunisere med ens enheter, i tilfelle det skulle foreligge nettverksproblemer for eksempel hvis man mister tilkoblingen så vil systemet bytte over til Sim-kort som en backupløsning.

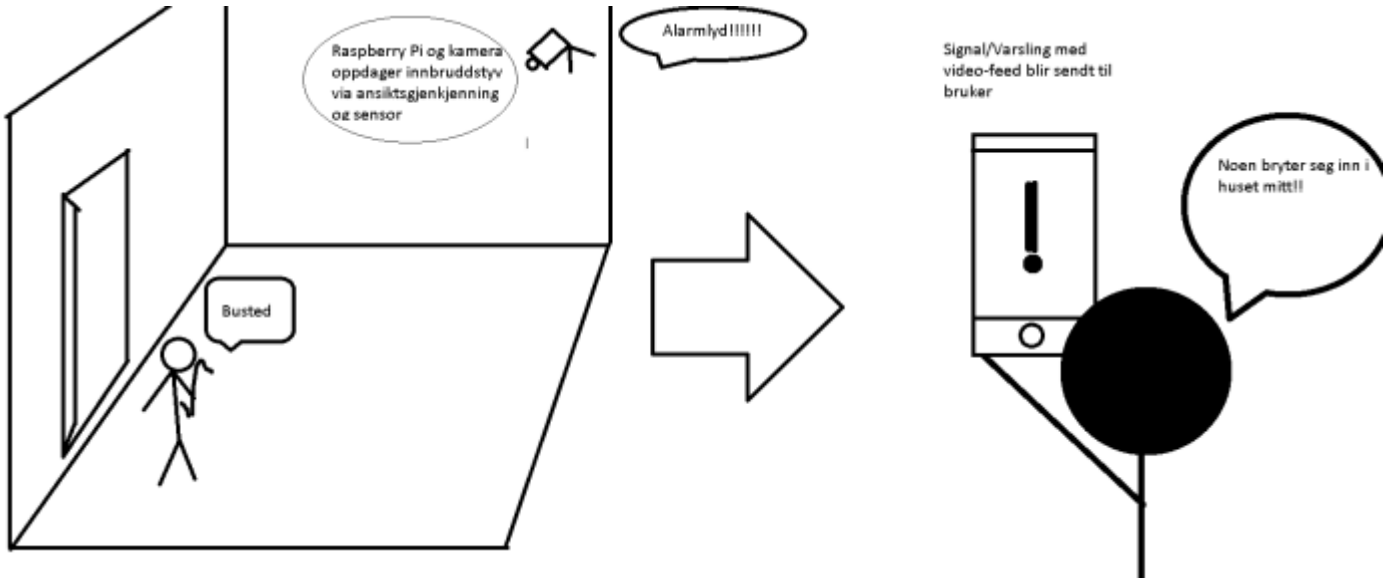
1.6 Konsept og viktig informasjon

Vi har arbeidet med dette prosjektet gjennom to semestre på universitetet, og på grunnlag av dette har vi innsett at kompleksiteten av å produsere et reelt produkt av OSecurity vil presse oss til nye lengder, øke vår kompetanse og styrke oss som gruppe, på grunn av følelsen av eierskap og ettersom dette er noe vi potensielt ønsker å jobbe videre med. En av de største utfordringene vil være sikkerhetsaspektet, både med tanke på maskinvare og programvarenivå. Dette er

avgjørende ettersom det vi produserer er et sikkerhetssystem. Vi må muligens lære oss noe om kryptografi og kryptering for å kunne tilfredsstille så vel som å håndtere og implementere ulike kryptografiske funksjoner. Dette for å sørge for et høyt sikkerhetsnivå for brukerne.
Størrelsen på prosjektet er omfattende, selv om vi har en minimalistisk fungerende prototype, så ønsker vi å lage et levedyktig prosjekt som vi potensielt kan få ut på markedet i fremtiden. Dette betyr at det må lages på en profesjonell måte i tråd med eksisterende standarder. Det må i tillegg etterhvert bygges en nettbutikk som tar hånd om salg av produktet. Vi må dermed refaktorere og forbedre kildekoden så vel som å utvikle en fullstendig mobil applikasjon.

1.7 Rikt bilde

Det rike bilde har som formål å illustrere for kunden og potensielle kunder hvordan systemet vil fungere i praksis.



Figur 1: Rikt bilde:

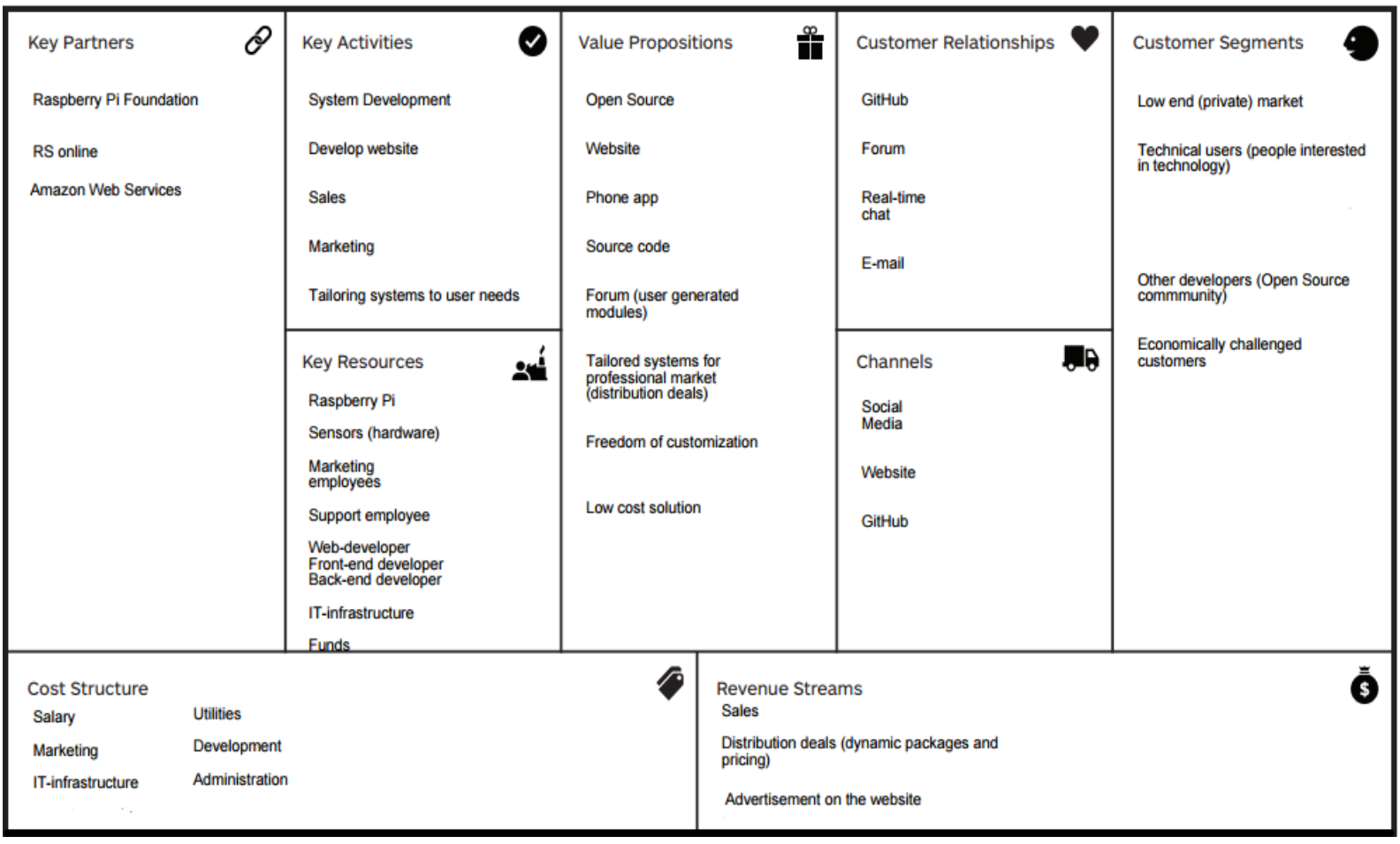
1.8 Forretningsrelatert arbeid

Ettersom vi har valgt å etablere en studentbedrift i forbindelse med bachelorprosjektet har vi også valgt å ta for oss den forretningsmessige biten av prosjektet. Vi vil i dette kapittelet ta for oss noen av de forretningsmodellene som vi føler er mest hensiktsmessig for prosjektet.

1.8.1 Business model canvas

Business model canvas er et visuelt kart som beskriver bedriftens/produktets verdi, infrastruktur og kostnader. Dette hjelper bedriften med å samkjøre sine aktiviteter ved å illustrere mulige avveininger som må gjøres. (Wikipedia, 2017)

For å vise hvordan OSecurity skaper og leverer verdi har vi tatt i bruk et Business model canvas. Salg av systemet er den primære verdien som OSecurity genererer økonomisk, til inntekt for studentbedriften "OSecurity". Systemet skaper også en verdi for kunden som får dekket et trygghetsbehov. Kunden får gjennom denne tjenesten mulighet til å overvåke hjemmet og vil bli varslet om det inntreffer eventuelle hendelser og får derfor dekket dette behovet.



Figur 2: Business Model Canvas

1.8.2 Produktpakker

For et lanseringsklart produkt ønsker vi å tilby kundene ulike pakkeløsninger slik at vi kan tilfredsstille kundens behov på best mulig måte. Det er da opp til hver enkelt kunde å velge hvor mye de ønsker å legge inn i sitt sikkerhetssystem. Det vil også være muligheter for å kjøpe ekstrautstyr for å bygge på et eksisterende sikkerhetssystem. OSecurity tilbyr også en streaming pakke som kan brukes til ulike formål. Et eksempel på bruksområde er at den muliggjør at foreleser kan streame forelesninger og publisere det som fagstoff. Pakken kan også fungere som en babymonitor eller lignende enkel videoovervåkning og videooverføring.

Tilleggsutstyr	Foreleserpakke / Babymonitor	Sikkerhetspakke	Utvidet sikkerhetspakke	MVP
Pir-sensor	Pi Model 2B	Pi Model 2B	Pi Model 2B	Pi Model 2B
Indoor IP-cam	1 Indoor Camera	1 indoor camera	2 pir-sensor	PIR-sensor
Outdoor IP-cam	1 Mikrofon	2 outdoor camera	Indoor IP-cam	Kabinett
Lys-kontroll	Kabinett	16 GB SD card	Outdoor IP-cam	Wifi-adapter
Night-vision	16 GB SD	Kabinett	Kabinett	App
Varmestyring	Wi-fi adapter	Wifi-adapter	Wifi-adapter	AWS
Vindusensor	App	App	2 Magnet-sensor	1 Kamera
Dørsensor	Livestream	Livestream	App	
	UPS	UPS	UPS	
	GSM	GSM	GSM	
	AWS	AWS	AWS	

Figur 3: Pakkeløsninger

1.8.2.1 Pakkeløsninger:

MVP: Denne pakken inneholder grunnelementene for OSecurity og er passende for brukere som ønsker å ha relativt god kontroll på sikkerheten sin til en billig pris.
Foreleserpakke / Babymonitor: Denne pakken er tilpasset for enten å streame forelesninger eller for å gi foreldre en mulighet til å monitorere deres baby når de skal legge babyen. Pakken inneholder da ikke moduler for hjemmesikkerhet og vil derfor være en rimeligere utgave.
Sikkerhetspakke: Denne pakken inneholder moduler og sensorer for hjemmeovervåking.
Utvidet sikkerhetspakke: Denne pakken er lik sikkerhetspakken men det følger med flere kameraer og sensorer for å få et bedre og utvidet sikkerhetssystem

Tilleggsutstyr: Her kan kunden selv plukke ut de produktene han/hun måtte ønske for å utvide sitt system med.

2 Prosjektstyring og sentrale beslutninger

2.1 Endring av Bachelorprosjekt

Den innledende fasen av bacheloroppgaven involverte et helt annet prosjekt vi i utgangspunktet hadde planlagt å produsere. Grunnet uenigheter med daværende product-owner så vi oss nødt til å avslutte samarbeidet og å starte på nytt. Da var det nåværende prosjektet aktuelt for oss, ettersom vi allerede hadde lagt ned mye tid i utviklingen av dette prosjektet og var inneforstått med kompleksiteten av å gjennomføre dette som et reelt produkt som kan ferdigstilles.

2.2 Metodikk

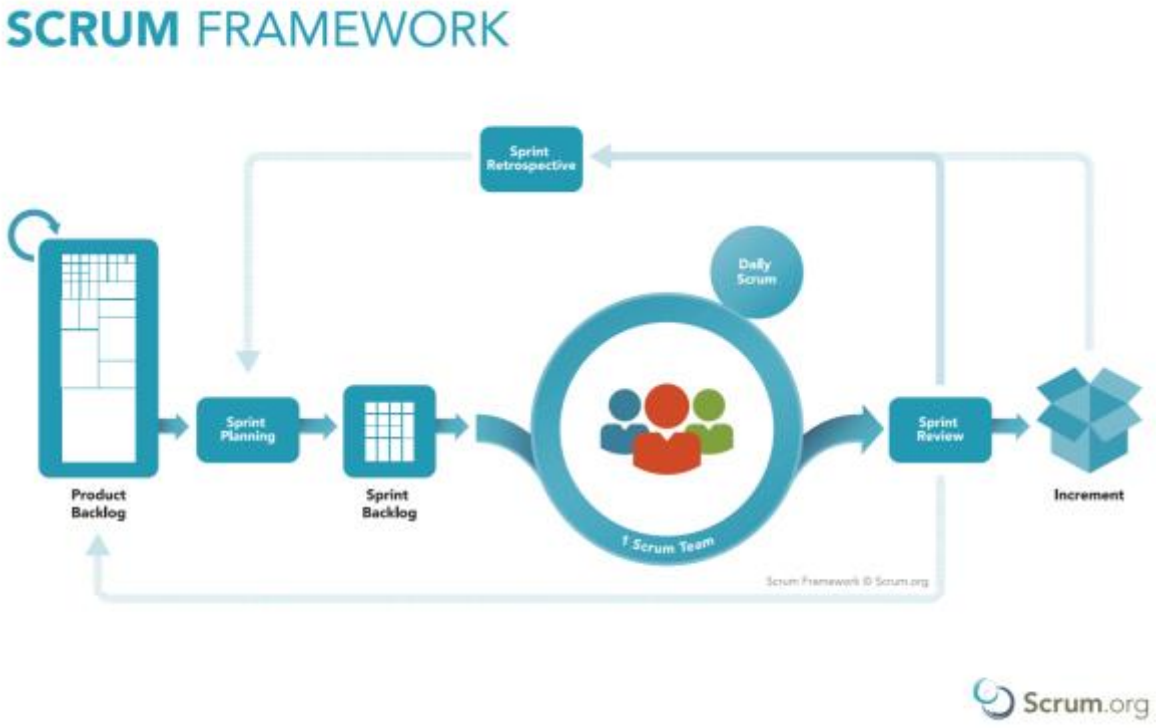
Som systemutviklingsmetodikk har vi valgt å bruke Scrum. Bakgrunnen for valget vårt av metodikk, er hovedsakelig at det virker mest hensiktsmessig for vårt prosjekt med tanke på struktureringen og kontrollen vi dermed kan ha over et vidt spekter av oppgaver innenfor ulike kategorier. Innledningsvis med tanke på det opprinnelige prosjektet, var tanken å ha en DevSecOps[1] tilnærming for å kunne ha hyppige iterasjoner med tilbakemelding på grensesnitt før videre utvikling. Selv om vi i hovedsak vil ta i bruk Scrum, er det likevel noen Kanban elementer vi ønsker å ha med oss. Spesielt dette med å visualisere oppgaver og kategorisere dem, samt prinsippet med å begrense elementer som er under arbeid. (Wikipedia, 08.05.17)

2.2.1 Backlogg, prioritering og estimering

Backloggen vår har stort sett vært overordnet, med brukerhistorier i fokus. Brukerhistoriene har således vært utgangspunktet for funksjonaliteter vi ønsker å få implementert i en gitt sprint. Dermed har vi blitt enige på forhånd i løpet av Sprint Planning møter hvilke brukerhistorier som det skal fokuseres på. Brukerhistoriene har dermed blitt delt opp i forskjellige mindre oppgaver man kan delegere til Scrum Teamet. Siden det er vi selv, OSecurity SB som er produkteier så har vi selv satt opp product backlogen. Backloggen er utformet basert på hva vi mener er nødvendig av funksjonalitet for å produsere et “minimum viable product”. Dette har vi kommet frem til ved å bruke MoSCoW (Wikipedia, 2017). Dermed må vi implementere alle Must-have backlog-oppgavene for å ha et MVP, som potensielt kan være gjenstand for videresalg til kunder gjennom vår studentbedrift.
De ulike deloppgavene estimeres i timeantall og backlogg-oppgaver estimeres i

[1] En systemutviklingsmetodikk med hyppige iterasjoner med samspill mellom de ulike avdelingene (utvikling og drift), men her også med fokus på sikkerhet.

både timer og kompleksitet, i henhold til Planning Poker (Wikipedia, 2017) prinsippet. Sprintene har hatt en varighet fra 2-4 uker, og vi gjennomfører Sprint Reviews og Retrospect etter hver endte sprint.



Figur 4: Scrum framework

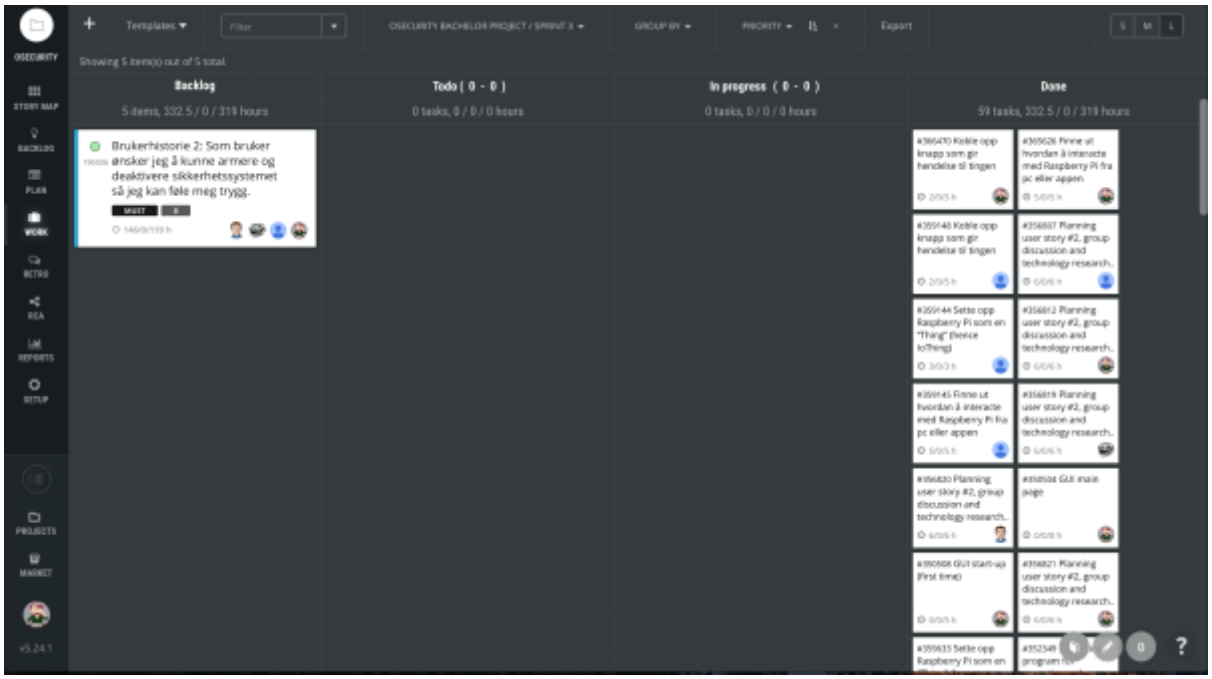
(Scrum Framework, n.d)

2.2.1 Planlegging og prosjektstyring

Ved hver påbegynnelse av en sprint vil det bli avholdt et sprint-planning møte. (Ref. vedlegg 4: Utfyllende Scrum logg) På dette møtet gjennomføres estimering av oppgaver (deloppgaver ut fra brukerhistorier) som trekkes inn i den gitte sprinten basert på tilgjengelige ressurser (timer) vi har tilgjengelig. Vår fullstendige backlogg er dermed alle brukerhistorier samt øvrige forfallende oppgaver vi selv definerer basert på behov.

For å gjøre dette i praksis tar vi i bruk planleggingsverktøyet Scrumdesk (Scrumdesk, 2017), som noen av gruppemedlemmene hadde erfaring med fra tidligere. Innledningsvis ønsket vi å ta i bruk prosjektstyringsverktøyet Trello, men det viste seg tidlig å være en lite ønskelig løsning da det ikke var tilrettelagt for loggføring av timer, samt at man ikke fikk like bra dokumentasjon (rapporter, diagrammer, kontroll over timer) som det man gjør på Scrumdesk.

Eksempel på Scrumdesk taskboard:

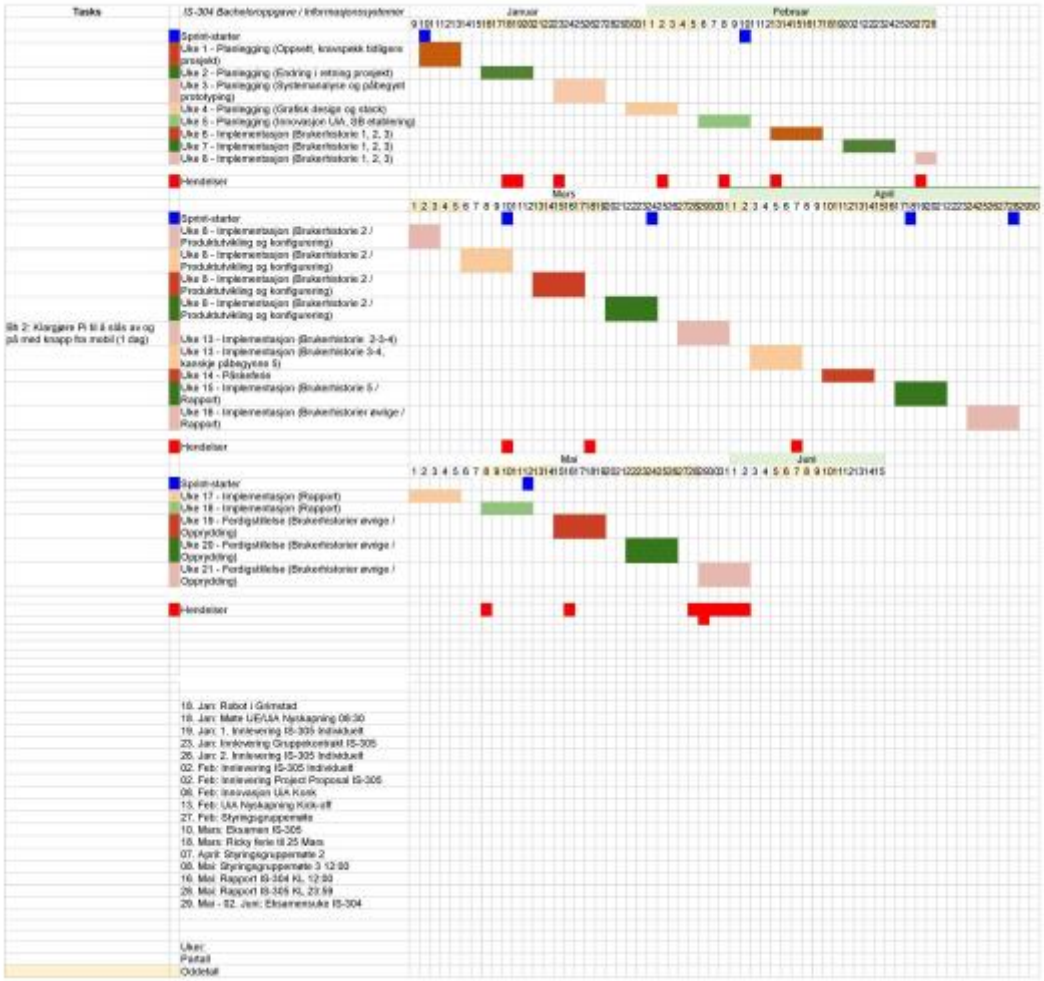


Figur 5: Scrumdesk

Her ser man at brukerhistorien ligger til venstre i “Backlog”, den kan splittes opp i “Tasks” som må gjøres for at brukerhistorien er fullført. Når alle tasks er delt opp og estimert, ligger disse på “Todo”. Deretter fordeles tasks på brukerne enten ved å bli tildelt oppgaven av Scrum Master Robin, eller dersom det er avtalt hvem som skal gjøre hva, så kan man påta seg en oppgave selv ved å dra den over til “In progress”. Når den er ferdig dras den over til “Done”. Disse oppgavene loggføres ettersom gruppemedlemmene logger timer på dem, og kan følges opp gjennom prosjektet ved å gå inn på ”Reports” på Scrumdesk og deretter velge hva slags fremvisninger som ønskes. Eksempel på logg av ført arbeid:

2.2.2 Prosjektplan

I begynnelsen av prosjektet ønsket gruppen å fastsette en overordnet prosjektplan for semesteret, dette for å kartlegge ulike milepæler og starter/slutter på sprinter samt hvilke oppgaver som bør være ferdigstilt til når. Denne er utarbeidet i forskjellige versjoner, da den måtte oppdateres ettersom det ble endringer i planene underveis i prosjektet. Dermed ble det en dynamisk plan, som vi alltid var optimistiske på, men samtidig gjorde så godt vi kunne for å overholde. Her er det endelige resultatet, mens tidligere versjoner er dokumentert i (Ref. vedlegg 5: Prosjektplan).



Figur 6: Gantt chart

2.2.3 Daily scrum

Daily scrum ble gjennomført i form av daglige statusoppdateringer ved oppmøte og forespørsel om utfordringer. Likevel, var gruppen i stor grad oppdatert på hvilke utfordringer som var til stede da man ofte opplevde dem sammen. Det var likevel nyttig å ha en liten oppdatering hver dag spesielt om noen hadde arbeidet hjemmefra.

2.3 - Innledende timeestimat

I forbindelse med prosjektgodkjenningen for Bachelorprosjektet sendte vi inn et innledende estimat på forventet tidsbruk. Det er som følger:

Et grovt estimat av timene: Vi anslår at den komplette MVP'en (Minimal Viable Product) vil komme til å ta totalt 2340, basert på disse utregningene:

26 timer * 18 uker * 5 medlemmer = 2340 timer
Planlegging - Analyse og design 20% 468 timer
Implementasjon - utvikling (front- og back-end), Infrastruktur (cloud tilbydere, AWS / kontainere, devops), brukertesting & testrutiner: 45% 1053 timer
Business del - Potensielt utvikle en businessplan, undersøke juridiske forhold med tanke på produktets funksjonalitet (Overvåkningslover, vårt ansvarsområde) 15% 351 timer
Dokumentasjon - prosjektrapport, etc. 20% 468 timer.
Vi er forberedt på å overstige det totale estimerte timeantallet, men vi håper at vi vil være i stand til å fullføre et MVP innenfor disse rammene.

2.4 QA - Kvalitetssikring

Til tross for at prosjektet blir utviklet av et lite team og i relativt liten skala, bestemte vi oss for å ha noen faste QA (Eng. Quality Assurance)-rutiner som vi skulle forsøke å opprettholde.

2.4.1 Regresjonstesting

Under planlegging og utvikling av prosjektet valgte vi å ta i bruk disse faste rutinene rundt QA, for å hjelpe oss med å identifisere og rette feil på raskest mulig måte, og for å optimalisere kvaliteten på det endelige produktet. Under QA inkluderte vi hovedsakelig funksjonelle feil og grafiske feil, men vi rapporterte også lokaliseringsfeil, som er feil relatert til skrivefeil, overlapping eller dårlige ordvalg.

Ved hver stabile versjon som ble grenet ut i vår Github-repository, gikk gruppen gjennom applikasjonen og testet den med sin egen bruker. Når en av oss fant en feil så var det faste rutiner på hvordan dette skulle håndteres, og vi fulgte et rapporteringsformat som så slik ut:

- Hva er feilen
- Hvor alvorlige er feilen (1. Grafisk / liten feil, 2. Funksjonell feil, 3. Alvorlig feil / krasjfeil)
- Hvordan oppsto feilen
- Oppstår feilen hver gang?
- Hvordan fremtvinge feilen / trigge feilen
- Forslag til løsning

Hver feil som rapporteres inn ble satt i en regresjonstabell med et nummer som identifikator og en kort forklaring av problemet. De med ansvar for utvikling i denne perioden vil da, i tillegg til å utvikle nytt innhold, prøve å rette feilene fortløpende og få opp en ny versjon av applikasjonen.

Når en ny stabil versjon av applikasjonen var klar, ville alle gruppemedlemmene bli tilsendt regresjonstabellen og måtte laste ned og kjøre gjennom applikasjonen, hvor målet var å kontrollere om feilene var rettet eller om de fremdeles oppsto. Hver enkelt ville da merke feilene med enten "FV" for fix verified, eller "FF" for fix failed, og hadde mulighet til å legge ved en kommentar for å begrunne feilens status.

2.4.2 Beste praksiser

Test-Driven Development (TDD) er en tilnærming til utvikling av kode hvor man skriver automatiserte tester først, for deretter å skrive kode som oppfyller testens krav til å gå gjennom, og deretter gjennomfører refactoring.

TDD-mantraen Rød/Grønn/Refactor er en rekkefølge for programmeringsoppgaver:
1. Rød: Å skrive en automatisert test som ikke godkjennes
2. Grønn: Å skrive den nødvendige koden for at testen i steg 1 skal godkjennes
3. Refactor: Forbedringer av eksisterende kode (Budhabhatti, 2008)

Noen av fordelene med TDD er blant annet at det forbedrer systemdesign ved å øke cohesion, på grunn av at tilnærmingen krever at man har en frakoblet (eng. decoupled) kodetilnærming. I tillegg får man en kontinuerlig tilbakemelding på om endringer i systemet gjorde at tidligere tester feilet, noe som gjør at man enkelt kan gjennomføre hyppige og trygge forandringer av kode. (Budhabhatti, 2008)
Noen av motargumentene for å innføre TDD er blant annet at det kan skape et for stort fokus på microdesign over macrodesign. Et eksempel kan være at fokus på små funksjoner og endringer kan gjøre at man ikke designer programvare som er bærekraftig for den videre utviklingen og påbyggingen.

> *""The first floor is done. It looks gorgeous; all the apartments are in perfect, livable condition. The bathrooms have marble floors and beautiful mirrors, the hallways are carpeted and decorated with the best art."*
>
> *"However," the builder adds, "I just realized that the walls I built won't be able to support a second floor, so I need to take everything down and rebuild with stronger walls. Once I'm done, I guarantee that the first two floors will look great.""*
>
> (Beust, 2014)

I tillegg nevnes det at TDD kan være vanskelig å anvende i alle situasjoner. Mye av litteratur som snakker om TDD rettes ofte mot eksempler som en poengliste for bowling, eller en container (en stack eller liste). Problemet med dette er at det ikke nødvendigvis hjelper utviklere med å forstå hvordan dette kan anvendes for mer komplekse kodebaser, og eventuelt kodebaser som ikke ble skrevet for å testes. Selv om TDD har sine utfordringer så er hensikten å få utviklere til å produsere bedre tester og ha bedre testrutiner, og det er fortsatt langt bedre enn kode uten tester. (Beust, 2014)

Ved TDD er det ikke uvanlig å implementere Continuous Integration (CI) som en utviklingspraksis for å hyppig integrasjon av nye versjoner. Med CI kan nye versjoner legges ut opptil flere ganger daglig. Prosessen er som følger.
Utvikler:

1. Skriver en ny unit test case
2. Kjører testen og kontrollerer at den feiler
3. Skriver kode slik at testen godkjennes
4. Publiserer koden til repository

CI server:

1. Laster ned ny kildekode når repository endres
2. Kompilerer og inspiserer koden
3. Setter opp påkrevde ressurser, som f.eks. konfigurasjonsfiler
4. Kjører testene på en emulator
5. Gir tilbakemelding basert på versjonens resultat

Noen av fordelene med implementasjon av CI er at det favoriserer tydelig kommunikasjon og gir høyere programvarekvalitet. Det gir mulighet for hyppig tilbakemelding og rask feilretting. For hver lille endring som gjøres i programvaren får man automatisk kontrollert om all eksisterende logikk fortsatt fungerer som ønsket. (Budhabhatti, 2008)

2.4.3 Android Robo Test

Google Firebase har et godt verktøy for kvalitetssikring og testing som heter Android Robo Test. Dette er et testverktøy som analyserer strukturen av applikasjonens grensesnitt og utforsker det metodisk, og automatisk simulerer brukeraktiviteter, noe som gjør at man kan bruke Robo Test til å regresjonsteste applikasjonen. Når testen kjøres lagrer den loggfiler, tar skjermbilder med kommentarer, og lager en video med skjermbildene for å vise hvilke simulerte oppgaver testen gjennomførte. Dette kan brukes til å finne kilden til krasjer og feil, og kan også avdekke svakheter med applikasjonens grensesnitt. (Google Firebase, n.d)

2.4.4 Brukertesting

Etter vi hadde fullført en komplett designprototype valgte vi å gjennomføre en brukertest for å få tilbakemelding på grafikk- og navigasjonsdesign. Vi tok en gjennomgang individuelt med flere individer, og noterte hvordan de løste oppgaver og hvilke usikkerheter og spørsmål de stilte oss under testingen.

Brukertesting av endelige papirprototyper

I forbindelse med et samarbeid med en annen bachelorgruppe valgte vi å arrangere et “hackathon”. Hensikten med dette var å dele kunnskap og se på hverandres ideer og designforslag, med tanke på brukertesting, brukervennlighet, grensesnitt og design. Dette er også en ny måte for oss å arbeide på, ettersom vi har jobbet veldig ensformig på prosjektet så langt.

Brukertest 1 (Jørgen Lybeck Hansen)

Login: Vurdere husk meg knapp for brukerinnlogging. (Så bruker kan velge om han vil ha) Ellers ingenting å påpeke.

Førstegangsoppsett side v1: Vil bruke appen som sikkerhetssystem, trykker på det og ser jeg skal få mer informasjon? (informasjonen kommer ikke).
Første gang jeg logger inn vil jeg få beskjed: Du kan endre hvor ofte vi skal ta bilder og filme for deg på innstillinger bilde/video.

Hovedside: Osec er armert, da vet jeg på en måte at den filmer? Eller, hva skjer i denne? Hva er dette bildet (siste bilde), er det siste bildet? (Svar: Ja)
Det jeg trodde var at play knappen her var for å starte denne videoen her for å starte filmen. Det jeg så for meg var en lengre knapp som sier “gå til live streaming” mens bildet kan stå. Kanskje siden dette er det tredje bildet jeg kommer inn på vil jeg få noen pop-ups om at det er mulig for meg å endre bildehyppighet. Eller en beskjed om: Gå til innstillinger så kan du endre på dette og dette.

Meny: Trykker på menyen, får opp valgene og er bestemt ganske tidlig på at han vil til innstillinger. Trykker på innstillinger. Det første jeg tenker når jeg ser denne skjermen med mine bilder og sånn er at ok her tar den bilder og sånn, men jeg vil stille på hvor ofte den tar bilder og sånn.

Support: Dritkult UI, digget det. Ok, så her vil jeg rett og slett komme inn på web-siden på chrome hvis jeg trykker på getting started. Her kan dere jo velge å bruke Github wiki for å vise frem informasjonen. Men ok hvis dere skal sende til nettsiden så er dette bra. Vil forum sende til nettsiden? Svar: Ja vi tenkte et Open Source community. (Nok en god grunn til å bruke GitHub wiki, proffe brukere kan da oppdatere wikien hvis de er lærd i Open Source) Suggest something blir til en text-field du kan sende inn til support? Contact support kunne gått til valg mellom å sende e-post og lage en ticket.

Generelle innstillinger: Blir forklart. Hva skjer hvis jeg trykker på de andre tabsene? For det jeg leter etter er hvor ofte bildene skal bli tatt. Blir forklart at går han inn i bilde så vil han kunne endre på det han vil endre. Her kan jeg endre, dritfett. Jeg ville skjønt at det var bilder, noe av skriften i mockupen var bare i dårlig font/litt for smått.

Innstillinger varsler: Flere valg: Ta bilde når det er motion. Hvis stillbilde er slått på vil jeg ha valg for hvor ofte bildene skal tas. Mulighet for å slå på motion-baserte stillbilder. Valg: Ta bilde hver gang motion registreres. Film 20sec hver gang motion registreres
Varsler: Åja, her er alle varslene. Hva er navnet der? (Bakdør) Svar: Det er navnet på sensoren. Hva er forskjellen på de to visningene? Svar: Den til venstre lister det opp, trykker du får du opp utvidet visning på høyresiden.

Arkiv: Står det reminder her? (Ser ut til å være dårlig font problem igjen). Det var Bakdør det stod. Spørsmål fra Thorne: Hvordan er navigeringen? Logg er alt, så hvis det er video får jeg opp det og hvis det er bilde får jeg opp det? Ja, det popper opp når du trykker på det.
Siden loggen er tidsbasert så kan kanskje video være kategorisert, en header for en type kamera (bakdør) og alle videoene under det. Sånn drop-down.
Burde bruke accordion.

Totalt:
Mye bra.

Brukertest 2 (Guro)

Login:
evt. Husk meg knapp. ikke noe mer å påpeke på login siden.
Førstegangsoppsett side v1:
Under utvikling legg til at disse modusene kommer. (fjerne v1)
Hovedside:
Hovedsiden inneholder de nødvendige fukjsonene. endre farge på låse ikonet når den er armert/desarmert for å tydeliggjøre om alarmen er aktivert eller ikke.
Meny:
ikke noe å påpeke. lett forståelig.
Support:
ikke noe å påpeke. lett forståelig. legge til mail/tlf på kontakt knappen. kanskje legge til en seksjon med “about”
Generelle innstillinger:
Er det mulighet for å aktivere bare vibrering?
Innstillinger varsler:
Blir litt dobbelt opp med justering av innstillinger på generelle innstillinger og på varsler.
vurdere å legge til SMS-varsling
Varsler:
Litt forvirrende med stjerne som marker uleste varsler. kanskje bytte til å bruke fet skrift i stedet for stjerne.
Arkiv:
mulig legge til en søkefunksjon for arkivet.
Ut fra tilbakemeldingene vi fikk fra testsubjektene konkluderte vi med at den generelle grafiske designen og navigeringen var lett forståelig, men det var noen hull i informasjonen som gjorde brukeren usikker i flere scenarioer.

Basert på tilbakemeldingene fra testingen gjennomførte vi en ny iterasjon med UX-design (User Empiriene Design), hvor vi forbedret på forklaringer og la til nye informasjonsvisninger i applikasjonen.

2.4.5 Unit-testing

Unit-tester tester funksjoner/metoder i programvaren basert på eksempelvis JUnit rammeverket i Java. JUnit bruker blant annet @Test annotasjoner for å presisere overfor kompilatoren at påfølgende kode skal teste en gitt klasse. JUnit bruker

Assert Statements, hvor et objekt eller en variabel eksempelvis skal tilsvare et annet objekt eller variabel (assertEquals). Eller (assertThat) var1 = true;
Selv om vi ikke har satt opp et rammeverk for testing i vår prototype er vi inneforstått med viktigheten av testing og hvordan tester fungerer. Grunnen til at vi har valgt å ikke bruke ressurser på testing er at vi innså at det var en del utenfor vårt scope i dette prosjektet, kompetanse og tid har ikke vært tilstrekkelig. Testing har blitt prøvd ut på App-siden av systemet uten å lykkes. Vi har brukt AWS sitt eksempelprosjekt, som ikke var satt opp med tester. Det ble da svært vanskelig for oss å teste noe som ikke er generert med funksjoner som enkelt kan testes, og det ville også innebære en omfattende jobb i å koble fra hverandre et ferdigbygd prosjekt.
Angående Android Robo Test er dette noe vi jobber med å inkorporere i vår utviklingsprosess. Vi ønsker at dette verktøyet skal ta over for vår manuelle testing og regresjonstesting av applikasjonen. I tillegg vil vi luke bort eventuelt menneskelige feil som ikke klarer å avdekke alle feil og svakheter, i tillegg til å automatisere prosessen.

3 Analyse

Den innledende sprinten (Sprint 1) ble i stor grad tatt i bruk til å gjennomføre analysefasen.

3.1 Brukerhistorier

Innledende i prosjektet hadde vi en analyse og planlegging hvor vi konstruerte brukerhistorier. Nedenfor ligger brukerhistoriene i prioritert rekkefølge, mer om hvordan vi kom frem til og prioriterte kommer vi tilbake til senere i kapittel 3.

Brukerhistorier

Must have

Brukerhistorie 1

1. Som bruker ønsker jeg å kunne logge inn på applikasjonen for å få tilgang til mine bilder/opptak og kunne aktivere/deaktivere systemet.

Vilkår for aksept: Logge inn/autentisere på applikasjonen
MUST HAVE
Planning poker: 21

Robin	Ricky	Sondre	Erik	CF
M	m	M	m	m
21	34	3	13	21

Brukerhistorie 2

2. Som bruker ønsker jeg å kunne armere og deaktivere sikkerhetssystemet så jeg kan føle meg trygg.

Vilkår for aksept: Armere/desarmere system
Testoppskrift:

1. Bruker logger inn på appen.
2. Bruker aktiverer system ved å trykke på aktivering/deaktiveringsknapp på hjemmeside.
3. Bruker deaktiverer system ved å trykke på aktivering/deaktiveringsknapp på hjemmeside.

MUST HAVE
Planning poker: 8

Robin	Ricky	Sondre	Erik	CF

M	m	M	m	m
3	0	1	13	3

Brukerhistorie 3
Som en bruker ønsker jeg å kunne se siste stillbilde fra hjemmet, slik at jeg kan få en indikasjon på situasjonen hjemme når det passer meg.

Vilkår for aksept: Se siste stillbilde fra mobilapplikasjon
Testoppskrift:

1. Bruker logger inn på appen.
2. Bruker ser siste stillbilde på appen.

MUST HAVE
Planning poker: 21

Robin	Ricky	Sondre	Erik	CF
m	m	m	m	m
21	34	13	21	21

Brukerhistorie 4
Som bruker ønsker jeg å få push-notifikasjoner når systemet oppdager anomaliteter, for å være oppdatert på hva som skjer når jeg ikke er hjemme.

Vilkår for aksept: Få notifikasjoner (push på bevegelsessensor registrering (aws er klar)
Testoppskrift:

1. Brukeren logger inn i appen.
2. Bruker aktiverer systemet.
3. Bruker skal få en push-notifikasjon på mobilen når det oppdages en hendelse.

MUST HAVE

Planning poker: 21

Robin	Ricky	Sondre	Erik	CF
S	m	m	m	m
5	13	55	3	8

Brukerhistorie 5
Som en bruker ønsker jeg å kunne se video live fra hjemmet, slik at jeg kan følge med på hva som skjer til en hver tid.

Vilkår for aksept: Se video live

Testoppskrift:

1. Brukeren må logge inn i appen.
2. Brukeren må kunne navigere seg frem til live feed visningen.
3. Brukeren skal kunne få opp live video fra hjemmet.

MUST HAVE
Planning poker: 55

Robin	Ricky	Sondre	Erik	CF
M	m	m	m	m
34	21	89	55	34

Should have

Brukerhistorie 6
Som bruker ønsker jeg å få push-notifikasjoner når systemet aktiveres og deaktiveres, for å vite om noen tukler med systemet.

Vilkår for aksept: Push på automatisk armering/desarmering (toggle for kunde), push på armering/desarmering)
Testoppskrift:

1. Brukeren må logge inn på appen.
2. Brukeren må kunne trykke på en armering/desarmere knapp for å aktivere/deaktivere systemet.

SHOULD HAVE
Planning poker: 3

Robin	Ricky	Sondre	Erik	CF
S	S	S	s	s
2	2	5	3	3

Brukerhistorie 7
Som bruker ønsker jeg å få tilgang på en logg over relevante aktiviteter så jeg kan ha kontroll over sikkerhetssystemet mitt.
Vilkår for aksept: Bruker skal kunne se og hente ut logg som blir oppdatert i sanntid fra terminalen.
Testoppskrift:
Brukeren må logge inn på appen.
Brukeren må navigere seg frem til aktivitets-logg.
Brukeren skal kunne se og hente ut logg i sanntid fra terminalen.
SHOULD HAVE
Planning poker: 13

Robin	Ricky	Sondre	Erik	CF
S	s	S	s	s
8	5	21	8	8

Brukerhistorie 8
For å skreddersy systemet til mitt behov til enhver tid, ønsker jeg som en bruker å kunne endre systemets innstillinger fra mobil-applikasjonen

Beskrivelse:
- Endre innstillinger for hyppighet stillbilder (f eks bar man kan slide?)
- Tidsstyring av system, automatisk armering / desarmering ved faste tidspunkt (innstillinger igjen)
- Innstillinger for e-post varsel (+ andre relevante innstillinger)

Vilkår for aksept: En bruker skal kunne endre innstilling for hyppighet av stillbilder, tidsstyring av system og e-post varsel.
Testoppskrift:
1. Brukeren må logge inn.
2. Brukeren må navigere seg frem til innstillingvisningen.
3. Brukeren må kunne endre innstillingene etter eget ønske.

SHOULD HAVE
Planning poker: 55

Robin	Ricky	Sondre	Erik	CF
S	s	S	s	s
55	?	13	34	55

Brukerhistorie 9
Som en bruker ønsker jeg å kunne starte og stoppe opptak av video, for å kunne ta opp video fra en kommende hendelse.

Vilkår for aksept:
Sette på opptak/stoppe opptak
Testoppskrift:
1. Brukeren må logge inn på appen.
2. Brukeren må navigere seg frem til videovisningen.
3. Brukeren må kunne starte og stoppe opptak.

SHOULD HAVE

Planning poker: 8

Robin	Ricky	Sondre	Erik	CF
S	S	S	s	s

5	13	8	5	8

Could have

Brukerhistorie 10

Som foreleser ønsker jeg å kunne livestreame forelesninger slik at studenter som ikke kan møte opp til forelesning kan ta igjen tapt undervisning

Vilkår for aksept: En bruker skal kunne sette på opptak og stoppe opptak, og laste opp opptaket (med bilde og lyd) til en skytjeneste.

Testoppskrift:

1. Brukeren må logge inn i appen.
2. Brukeren må navigere seg frem til opptaksvisning.
3. Brukeren må kunne starte og stoppe opptaket med lyd og bilde.
4. Brukeren må laste opp opptaket til en valgfri skytjeneste.

COULD HAVE

Planning poker: 21

Robin	Ricky	Sondre	Erik	CF
C	C	C	c	c
34	34	21	5	8

Brukerhistorie 11

Som en bruker ønsker jeg å kunne kommunisere med systemet om internett er nede, for å øke oppetiden og sikkerheten.

Vilkår for aksept: GSM-transmitter med SIM-kort, bruk av mobilnett f.eks. 4G, og mulighet for SMS- og anropsvarsling.

Testoppskrift:

1. Systemet må være på.
2. Systemet må kunne gå fra wifi og over til GSM-nett automatisk om wifi detter ut.

COULD HAVE

Planning poker: 13

Robin	Ricky	Sondre	Erik	CF
C	C	C	c	c
21	13	5	5	5

Brukerhistorie 12

Som bruker, ønsker jeg å fortsatt kunne overvåke hjemmet mitt om strømmen forsvinner, slik at jeg kan føle meg trygg uansett omstendigheter.

Vilkår for aksept: Systemet deaktiveres ikke dersom hoved-strømtilførsel forsvinner.

Testoppskrift:

1. Systemet må være på.
2. Systemet må kunne bytte til en intern strømkilde automatisk om det blir strøm brudd.

COULD HAVE

Planning poker: 21

Robin	Ricky	Sondre	Erik	CF
W	C	C	c	c
34	5	13	?	5

Brukerhistorie 13
Som bruker, ønsker jeg å kunne legge til en ekstra modul til systemet for å føle meg enda mer sikker.

Vilkår for aksept: Mulighet for å legge til et nytt kamera, en mikrofon eller en sensor.
Testoppskrift:

1. Brukeren må logge inn i appen.
2. Brukeren må navigere seg frem til innstilling-visningen legg til modul.
3. Brukeren må kunne trykke på knappen legge til modul.
4. Terminalen må kunne finne den nye modulen for så og aktivere den.

COULD HAVE
Planning poker: 89

Robin	Ricky	Sondre	Erik	CF
C	s	W	c	c
89	34	89	89	89

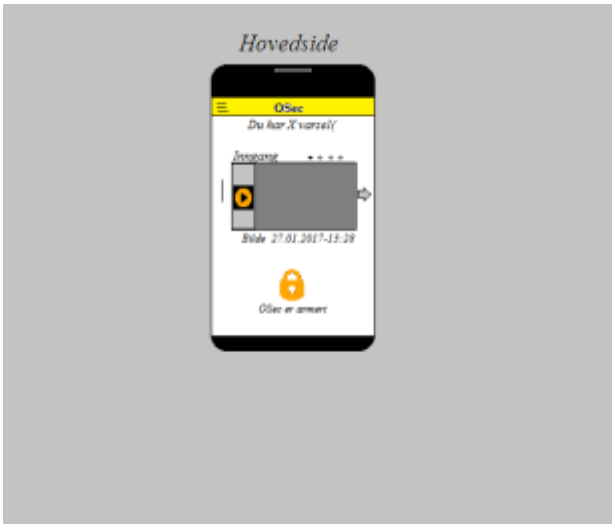
3.2 Grafisk design og prototyper

Under planleggingsfasen av prosjektgjennomføringen la vi vekt på å produsere et brukergrensesnitt som var intuitivt og enkelt for våre brukere. For å få et solid utgangspunkt hadde vi en skisseringsprosess som skulle gi oss det mest solide fundamentet til videre utvikling. Prosessen vår var at gruppemedlemmene hver for seg skisserte en egen versjon av en gitt visning uten å samarbeide eller dele den med noen andre. Dette ble gjentatt for alle applikasjonens visninger.
For eksempel tegnet alle en skisse av hjemmesiden i applikasjonen, før vi tok en runde der vi delte og forklarte våre idéer og begrunnet våre løsninger. Etter alle løsningene var lagt frem, tok vi en åpen diskusjon og skisserte en ny versjon der vi samlet det vi anså som de beste elementene fra gruppemedlemmenes individuelle løsninger.

Når vi produserte en endelig skisse og videre prototype, la vi vekt på at dette skulle være kjent for alle Android-brukere, og la derfor vekt på å følge Androids egne guide, Material Design, for å skape en så *native* applikasjon som mulig. Material Design er

en omfattende guide for visuelt-, bevegelses- og interaksjonsdesign på tvers av enheter. Formålet med dette rammeverket er å skape et visuelt språk som syntetiserer tradisjonelle prinsipper innen god design. (Android Developers, n.d)

Noe av det viktigste vi la vekt på under design for Android var det visuelle oppsettet og navigeringen, spesielt med tanke på den tradisjonelle topp-baren og menynavigeringen som er en gjenganger i Android-applikasjoner. Nedenfor er et eksempel på hvordan våre endelige skisser så ut:



Figur 7: Hovedside V1

Figur 8: Meny V1

Flere skisser finnes under (Ref. vedlegg 8: Brukerhistorier).

3.3 Kravspesifikasjon

Maskinvarespesifikasjon for Raspberry Pi 3B:

A 1.2GHz 64-bit quad-core ARMv8 CPU
802.11n Wireless LAN
Bluetooth 4.1
Bluetooth Low Energy (BLE)
1GB RAM
4 USB ports
40 GPIO pins
Full HDMI port
Ethernet port
Combined 3.5mm audio jack and composite video
Camera interface (CSI)
Display interface (DSI)

Micro SD card slot (now push-pull rather than push-push)
VideoCore IV 3D graphics core

Raspberry Pi 3 Model B. (n.d.).

3.3.1 Rammekrav:

Hvis systemet skal fungere hensiktsmessig bør driftsstans være minimal og en tilnærmet konstant oppetid, det avhenger dog av brukeren. Hvis f.eks. tilgang på strøm er fraværende så vil systemet naturligvis ikke fungere for brukeren.
Sikring mot tap eller ødeleggelse av data:
Ved å bruke Cloud Computing Services (CCS) som i vårt tilfelle er AWS, så har man i utgangspunktet sikret seg mot tap og ødeleggelse av data ved at data prinsipielt lagres i skyen og kontinuerlig backes opp der uten at vi som systemutviklere trenger å ta hensyn til dette.

3.3.2 Kjøretid og nøyaktighet:

Ettersom dette er et sikkerhetssystem som skal varsle brukeren om eventuelle anomalier i hjemmet er det viktig med kjøretid og nøyaktighet. Systemet skal kunne kjøres 24 timer i døgnet og 7 dager i uken om nødvendig, og kjøretid vil være avhengig av brukerens strøm/internett stabilitet. For nøyaktighetens skyld blir aktivitet på sensorene loggført slik at brukeren vil kunne registrere dersom sensorene skulle være defekt eller oppføre seg unormalt. Oppetid på serversiden blir tilnærmet konstant ettersom vi bruker AWS.

3.3.3 Applikasjon

Applikasjonsdelen av systemet er utviklet for Android-plattformen og styres fra telefon eller nettbrett som kjører på Androids operativsystem.
Applikasjonen skal styre systemet i form av å armere / desarmere alarmsystemet, og den skal ha mulighet for å gi varsler.
I tillegg skal man ha mulighet til å se en logg over alarmer / utslag på sensorer i hjemmet, for eksempel via en listevisning som viser tidspunkt og lokasjon for utslaget (dvs. hvor i boligen/området anomaliteten oppdages)
Klienten OSecurity ønsker også på vegne av sluttbrukeren å inkorporere følgende funksjonalitet: Livestreaming/Live overvåkning gjennom et lite tilkoblet kamera “Raspicam” av bolig eller annen privat grunn. Livestreamen skal kunne styres av bruker for å se til at alt er som normalt. Dette kan for eksempel være noe så enkelt som å se til at et kjæledyr har det bra om man er bortreist.

Inndatavolumet på applikasjonen vil være små datamengder. Applikasjonen mottar kun data ved forespurt logg eller ved at en alarm utløses.

Utdatavolumet vil ha en høyere frekvens, ettersom applikasjonen skal brukes til å styre systemet. Det er fremdeles en minimal datamengde siden det kun er styringssignal som blir sendt, selve behandlingen av disse signalene blir gjort av terminalen (RaspberryPi).
Interaksjoner vil øke bruksfrekvensen, som for eksempel ved alarm eller ved interesse for å hente ut logg. Applikasjonens oppetid er avhengig av brukerens nettverkstilgang og strømforsyning til terminal og sensorer i hjemmet, ettersom både inn- og utdata går over nettverk, og at resten av systemet krever strøm for å kjøre.

3.3.4 Teknologier som benyttes:

- Java for utvikling av mobil plattform til Android OS for brukernes enheter.
- Python for utvikling av programvare på Raspberry Pi. Programvaren vi utvikler her fungerer på lavnivå, og kommuniserer gjennom Operativsystemet til maskinvare (Raspbian, en Debian variant) og styrer systemets sensor og kamera.
- Amazon Web Services (AWS). Serverløsningen for systemet som fungerer som styringsenheten mellom Client (App) og Terminal (Raspberry Pi). Simple Storage Service (S3) Elastic Compute Cloud (EC2) Cognito. De ulike teknologiene blir beskrevet under kapittel 5 Teknologi og Stack.
  AWS innehar mye ressurser, rammeverk og verktøy man kan ta i bruk, og er å anse som bransjeleder innen Cloud Computing og CC-infrastruktur.
- For ytterligere informasjon om hvilke teknologier som ble brukt henvises det til kapittel 5 om teknologi og stack.

3.4 Planning poker

Vi tok i bruk metoden “Planning poker” for å estimere kompleksiteten til de ulike brukerhistoriene. Da gjorde vi vårt beste for å estimere basert på kompleksitet og ikke nødvendigvis tidsbruk, men i noen tilfeller fløt det litt over i hverandre. Vi tok i bruk et online verktøy for å gjennomføre dette da vi anså det som mer effektivt enn å manuelt skrive ned / fremvise kort. Vi brukte siden kalt:
“https://play.planningpoker.com/”, og hadde en god erfaring med dette.

Vi brukte følgende skala: 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ?(unsure), coffee break.

3.5 MoSCoW

For å prioritere brukerhistoriene gikk vi først gjennom alle brukerhistoriene og prioriterte dem i henhold til MoSCoW. Dette gjorde vi ved å legge opp tabeller hvor vi alle på samme tid la inn vår personlige prioritering, og kom frem til en felles prioritering basert på gjennomsnittet.

Da vi fikk delt inn brukerhistoriene i disse kategoriene, gikk vi videre og estimerte ved bruk av planning poker før vi til slutt prioriterte brukerhistoriene i de ulike kategoriene for å få den optimale rekkefølgen vi burde implementere dem i.

3.6 Klassediagram

Klassediagrammet ble i fellesskap skissert tidlig i analysedelen av prosjektet. Ut fra klassediagrammet produserte vi videre modeller som bidro under implementasjon av systemet. Selve klassediagrammet ble ikke brukt til særlig grad senere i prosjektet, og er derfor noe utdatert iht. den faktiske tekniske løsningen som senere ble planlagt og implementert.

Figur 9: Klassediagram

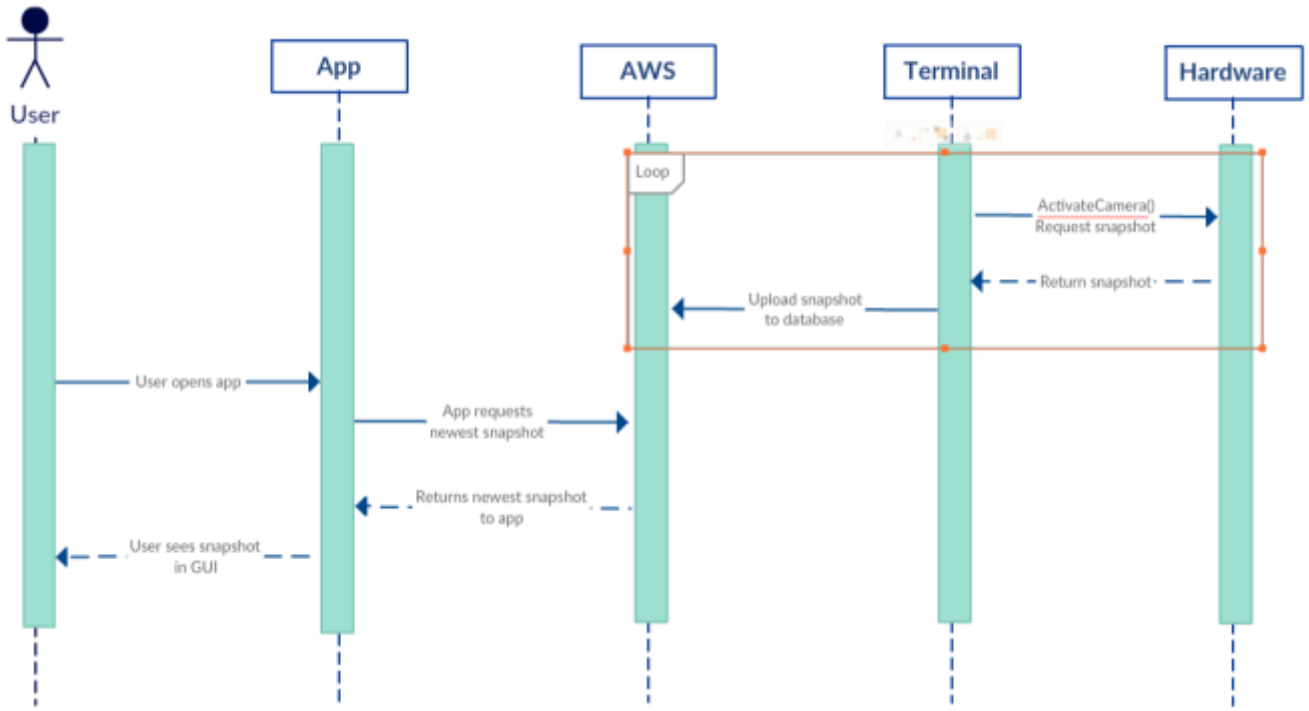
3.7 Funksjonsliste

For å kartlegge funksjonelle krav til prosjektet valgte vi å produsere en høy-nivå overordnet funksjonsliste ut fra de mest kritiske brukerhistoriene. Dette for å assistere oss med å produsere en fullstendig produkt-backlog til bruk senere i prosjektet.

Hovedfunksjoner i systemet	**Kompleksitet**		**Type**	
Log in	Kompleks		Avlesning	
Log out	Enkel		Oppdatering	
Send notifications	Kompleks		Signalisering	
Recieve notifications	Kompleks		Avlesning	
Upload resource	Medium		Beregning	
Download resource	Medium		Avlesning	
Save resource (local)	Enkel		Oppdatering	Oppdatering
Start camera	Enkel		Signalisering	Signalisering
Stop camera	Enkel		Oppdatering	Avlesning
start livestream	Særdeles Kompleks		Beregning	Beregning
stop livestream	Særdeles Kompleks		Oppdatering	
Activate sensor	Enkel		Avlesning	
Deactivate sensor	Enkel		Avlesning	
Send sensor signal	Medium			
Manage sensor signal	Medium			
Activate system	Medium		Signalisering	
Deactivate system	Medium		Signalisering	
Start buzzer	Enkel		Signalisering	
Stop buzzer	Enkel		Oppdatering	

Figur 10: Funksjonsliste

3.8 Sekvensdiagram



Figur 11: Sekvensdiagram

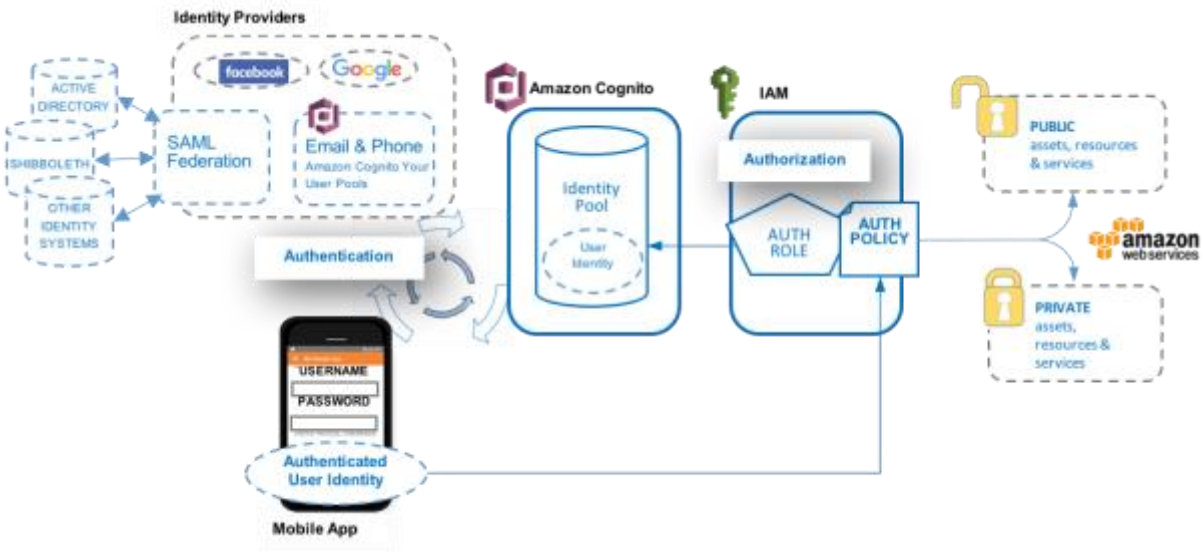
Denne illustrerte funksjonen ble implementert på samme måte som visualisert tidlig i planleggingsfasen, og produksjon av diagrammet var behjelpelig for å få en oversikt over hvordan dette skulle løses på best mulig måte.

4 Design

4.1 Arkitektur

Figur 12: Arkitektur V1

Denne arkitekturen er vårt første utkast til hvordan systemet vil fungere i praksis. Raspberry Pi'en vil inneha all input/output logikk som omhandler behandling av sensorer og kamera, dvs. Det som ligger på maskinvarenivå. Raspberry Pi'en vil da videresende signal dersom en sensor oppdager bevegelser videre til server i AWS som deretter blir behandlet der. Det samme gjelder for opptak via kamera og andre funksjoner. Sensorsignal, videoopptak og snapshots vil så bli lagret i databasen med logg for hendelsen, hva som skjedde, hvor og når.
Når denne informasjonen har blitt behandlet i Cloud-tjenesten skal det bli aksesserbart via Android appen på enheten man innehar. Kommunikasjonen vil foregå via AWS VPC, en VPN tjeneste som forsikrer en trygg og kryptert kommunikasjonstunnel.
Man vil da få push notifikasjon(er) og diverse valg som å starte live-feed og evt. Slå av alarmen dersom det viser seg å være en falsk alarm. Klientsiden (mobilappen) styrer systemet, og Raspberry'en fungerer imidlertid som en "dum" terminal som i stor grad behandler av/på forespørsler.
I tillegg vil brukerinformasjon bli lagret i databasen, en mer detaljert modell for denne AWS arkitekturen ligger nedenunder:



Figur 13: Autentiseringsprosess

5 Teknologi og Stack

I dette kapittelet ser vi litt på hvilke teknologier og hvordan vår stack var satt opp under utviklingen.

Siden dette var et prosjekt vi har jobbet med tidligere var vi allerede klare på noen av teknologiene og verktøyene vi ønsket å benytte for utviklingen. Siden vi skulle skrive en Android-applikasjon falt valget på å benytte Java med IDE-en Android Studio. Grunnen til at vi gjorde dette valget var at gruppen allerede har god kjennskap til Java, og at noen av gruppemedlemmene tidligere har jobbet i Android Studio for å bygge applikasjoner.

På Raspberry Pi valgte vi å fortsette med Python som vi har brukt tidligere for dette produktet. Grunnen til dette er at Python egner seg godt til prototyping og har gode biblioteker og støtte for bruk av sensorer og IoT-protokoller, i tillegg til at gruppen har god kjennskap til Python fra før.

For versjonskontrollsystem (VCS) benyttet vi oss av GitHub, siden dette er kjent for gruppen, i tillegg til at man som student får gratis tilgang til private repositories, noe som var nødvendig for at sårbar AWS-informasjon ville ligge i prosjektet.

Det første store valget var valg av skytjeneste for å håndtere brukerdata, lagring og kommunikasjon mellom enheter. Valget falt til slutt på å bruke Amazon Web Services (AWS) til vår infrastruktur. Grunnen til at vi benyttet AWS er mangfoldig, men noen av de viktigste punktene for vår situasjon var prissetting av tjenester og støttedokumentasjon. AWS blir brukt både av oppstartsbedrifter og store bedrifter, grunnet dens skalerbarhet, betalingsløsninger og sikkerhet, blant annet bruker store bedrifter som Netflix, Expedia og Adobe alle AWS. (QuickBooks, n.d)

For oss ga dette oss muligheten til å få eksepsjonell infrastruktur til ingen kostnad grunnet AWS sin “free tier” løsning, uten at vi trengte å drifte eller betale for infrastruktur selv. I tillegg til infrastrukturen, kunne vi også skreddersy et eksempel-prosjekt fra AWS, som ga oss en prototype applikasjon for Android, som vi kunne hente inspirasjon fra og jobbe ut fra. Alt innhold generert av AWS ble vårt eierskap, og vi sto fri til å ta, endre og distribuere denne kodebasen.

For å kunne sende push-varslinger til enheter, var vi avhengige av å bruke en notifikasjonstjeneste, og vi fant ut at AWS sin SNS (Simple Notification Service) ikke hadde tilstrekkelig tilgang til å sende systemvarsler på Android enheter, og at dette eventuelt måtte kobles sammen med Google Cloud Messaging (GCM), som i nyere tid går inn under Google Firebase. Vi valgte derfor å ikke benytte oss av SNS, og heller benytte Firebase eksklusivt for push-varslinger. I tillegg er Android Studio skreddersydd for bruk av Firebase, og har en innebygd plugin for å sette dette opp i ditt eget prosjekt. I tillegg ga Firebase oss analyseverktøy og testverktøy som AWS ikke hadde tilgjengelig på sitt free tier, som gjorde at vi til slutt valgte å skreddersy

vår “stack” med flere skytjenester.

5.1 Skytjenestene

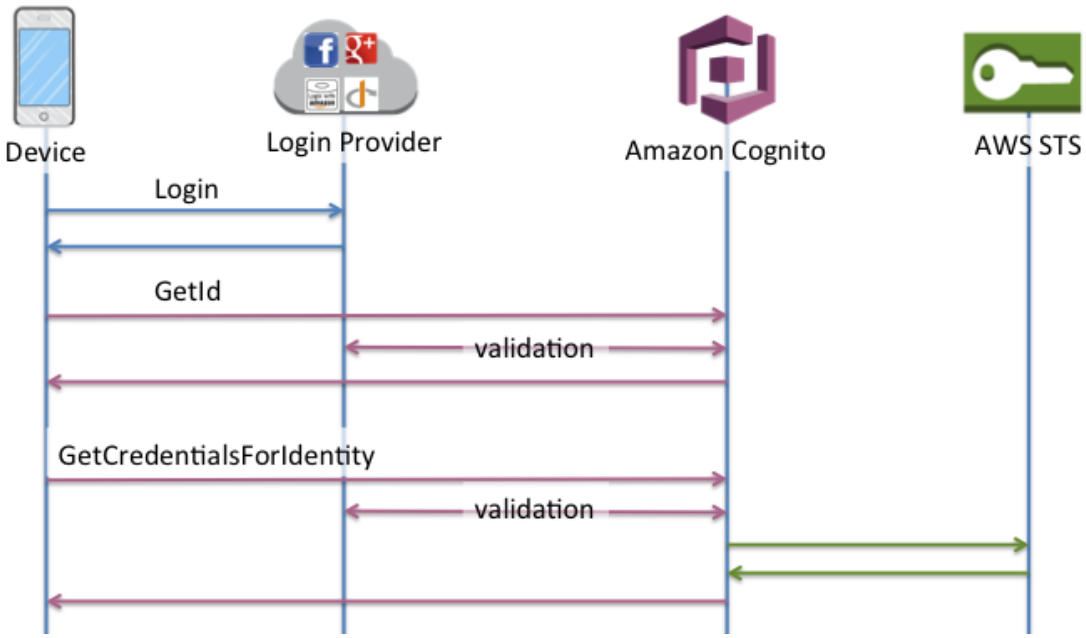
5.1.1 Amazon Mobile Hub

Mobile Hub er en tjeneste som forenkler bygging, testing og monitorering av mobile applikasjoner som benytter seg av AWS-tjenester. Mobile Hub lar deg hoppe over de tunge løftene å integrere og konfigurere tjenester manuelt, og lar deg legge til tjenester ved noen få tastetrykk, for eksempel brukerautentisering, datalagring, backend-logikk, push varslinger og innholdsleveranse. (Barr, 2015)

Ved å sette opp prosjektet i Mobile Hub kunne vi skreddersy infrastrukturen etter våre unike krav, og det ble generert en eksempel-applikasjon som viste det grunnleggende i hvordan man interagerte med og brukte disse tjenestene.

5.1.1.1 Amazon Cognito

For brukerregistrering, innlogging og autentisering brukte vi Amazon Cognito. I Cognito blir det opprettet forskjellige “pools” for brukere og identiteter, som fortløpende utvides når brukere registrerer seg, og tar seg også av innlogging av eksisterende brukere. Autentiseringsstrømmen er illustrert nedenfor ved AWS sin dokumentasjon.



Figur 14: AWS enhanced authentication flow

(Enhanced flow, 2015)

Som login provider benyttet vi oss av Amazons egne tjeneste, og Cognito tok seg av valideringen av innlogging. AWS STS står for AWS Security Token Service, en

tjeneste som genererer tokens som gir midlertidige en midlertidig legitimasjon for tilgang til AWS ressurser og tjenester.

5.1.1.2 Amazon IAM

AWS Identity and Access Management (IAM) brukes for å kontrollere tilgang til tjenester og ressurser for brukere. I Cognito kan man spesifisere at autentiserte brukere, for eksempel, får en gitt rolle tildelt når innloggingen er gjennomført. Da kan man enkelt gi denne rollen tilgang til de nødvendige ressursene, samtidig som man enkelt kan begrense tilgangen til bruker-spesifikke ressurser som private lagrede data. Dette administreres ved å opprette retningslinje-dokumenter i IAM-konsollen på AWS sin nettside, og deretter legge disse retningslinjene til roller og grupper av brukere. For eksempel kan et av disse dokumentene settes slik:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "*"
    }
  ]
}
```

Med denne retningslinjen, vil den gitte bruker, gruppe eller rolle, ha full tilgang til ressursen AWS S3 (Simple Storage Service) og alle tjenestene som går under S3. Videre kan slike dokumenter opprettes for de individuelle tjenestene og ressursene, for eksempel for å passe på at kun bruker A får tilgang til bruker A sin data, og at ikke bruker B.

5.1.1.3 Amazon S3

AWS Simple Storage Service (S3) er en enkel lagringstjeneste som vi bruker til lagring av brukerdata. Fra Raspberry Pi laster vi opp bilder og video, f.eks. ved alarm, slik at brukeren kan få tilgang til dette på mobilapplikasjonen.

5.1.1.4 Amazon IoT

For kommunikasjon mellom mobilapplikasjon og Raspberry Pi ønsket vi en lettvektet kommunikasjonsprotokoll, og kom frem til at MQTT var svært godt egnet til dette formålet. MQTT er en maskin-til-maskin (M2M), “Internet of Things”, protokoll som var laget for ekstrem lettvekt publisering/abonnering meldingstjeneste transport. (Mqtt, n.d)
Skyplattformen vi bruker er AWS Internet of Things (IOT), som lar koblede enheter kommunisere enkelt og sikkert med andre skytjenester eller andre enheter. Med AWS IoT kommuniserer mobilapplikasjonen direkte med Raspberry Pi over MQTT, og denne kommunikasjonen brukes blant annet til armering/desarmering av alarmsystemet og sending av data for push-varsler.

5.1.1.5 Amazon CloudWatch

Både for feilsøking og optimalisering benyttet vi oss av Amazon CloudWatch som er en overvåkingstjeneste for AWS-ressurser og applikasjoner som kjøres på AWS. Vårt primære formål denne tjenesten var å samle og overvåke loggfiler som ble generert når brukere etterspurte tilgang til AWS ressurser. Dette hjelper oss i stor grad ved løsing av feil. Blant annet oppdaget vi en, opp til det punktet, ukjent autoriseringsfeil mellom mobilapplikasjonen og AWS IoT.

5.1.2 Google Firebase

For å sende push-varsler benytter vi oss av Google Firebase sine "Notifications"-tjeneste, som kan sende systemvarsler til spesifikke brukere og/eller enheter, en gruppe av brukere, eller alle brukere samtidig. Firebase er konfigurert lokalt på applikasjonen og på Raspberry Pi. Enheten som har mobilapplikasjonen installert genererer en unik identifikator (Firebase registration id), som vi sender over MQTT til Raspberry Pi, for å sikre at systemvarslet alltid blir send til korrekt enhet. (Google Firebase, n.d) I tillegg til systemvarsel har også Firebase verktøyet Android Robo Test som nevnes under kapittelet om QA.

5.2 Tingenes internett (IoT) og IoT-sikkerhet

Samtidig som tingenes internett vokser, vokser utfordringene relatert til informasjonssikkerhet og personvern. Dette er noe vi måtte ta hensyn til under planlegging og utvikling. Det er primært fire kategoriske krav til sikkerhet og personvern som blir beskrevet:

- Motstandsdyktighet mot angrep: Systemet må kunne unngå en feil på ett punkt og tilpasse seg om en node feiler
- Dataautentisering: Adresse og objektinformasjon må autentiseres
- Tilgangskontroll: Informasjonstilbyder må kunne implementere adgangskontroll på de aktuelle data
- Klientens personvern: Det må tas steg som gjør at informasjon kun er tilgjengelig for en spesifikk bruker. Interferens skal vanskeliggjøres.

For å i best mulig grad møte kravene om informasjonssikkerhet og personvern i tingenes internett blir det brukt flere Privacy enhancing technologies (PET). For eksempel blir VPN-teknologier brukt for lukkede grupper eller avdelinger. Problemet med dette er at det blir vanskelig med en dynamisk global informasjonsutveksling, og er upraktisk for eventuelle tredjeparter utenfor nettet, noe som ikke gjorde dette til et aktuelt alternativ for OSecurity. Transport Layer Security (TLS) kan også bidra til å øke konfidensialiteten og integriteten[2] til IoT. (Weber, 2010).

[2] I forhold til CIA (Confidentiality, Integrity, Availability) triangelet innen informasjonssikkerhet

For kommunikasjon, tilgangskontroll og personvern benyttet vi oss av TLS og AWS. All kommunikasjon mellom enheter er kryptert, og adgangskontrollen styres av Amazon Cognito og AWS IAM. På denne måten har vi vist hensyn til og tatt steg for å møte kriterier for tilgangskontroll og dataautentisering. På hver Raspberry Pi ("thing") ligger det lokale sertifikater og nøkler som benyttes for å kommunisere med AWS-ressurser og andre enheter, denne kommunikasjonen er også beskyttet av TLS. På mobilapplikasjonen er alt sikret ved sikkerhet-tokens som genereres ved innlogging, tokens som må brukes for å få tilgang til informasjon eller for å kommunisere med andre AWS-ressurser og enheter.

5.3 Sikkerhet for skytjenestene

Cloud computing (CC) sikkerhet innehar alle problemstillinger som vanligvis assosieres med informasjonssikkerhet. Dette inkluderer design av sikkerhetsarkitektur, minimering av angrep, beskyttelse fra malware og forsterkning av adgangskontroll.
Det finnes dog noen aspekter som er spesifikt til dette domenet.

1. Skyen er normalt sett en delt ressurs, og andre "leietakere" kan være angripere.
2. Skybasert data er vanligvis intensjonelt tilgjengelig gjennom potensielt usikre protokoller og APIer over offentlige nettverk
3. Data i skyen har en risiko for å gå tapt (eksempelvis slettet ved uhell) eller feilaktig endret av skyleverandøren.
4. Data i skyen er tilgjengelig for skytilbyderen, dets underleverandør(er) og ansatte.

Mark D. Ryan påpeker at det eneste av disse punktene som er spesielt unikt for skyløsninger er nr. 4. da de andre generelt har vært tekniske utfordringer før Skyløsninger fantes, som for eksempel risiko for at data blir tuklet med eller går tapt. Vi ønsker likevel å se litt nærmere på lagring i sky, da det svært ofte er dette CC blir brukt til, også i vårt tilfelle, da sikkerheten og konfidensialiteten til våre potensielle brukere sine data er viktig. Ryan nevner forøvrig at de ulike problemstillingene (1-3) er vitenskapelig løsbare:
1. Delt ressurs problematikken kan løses ved å utplassere sterke VM løsninger og operativssystemer som sikrer at prosesser separeres.
2. Usikker transport kan løses ved autentisersingsprotokoller, autorisasosjonsrammeverk og kryptering.
3. Gode retningslinjer for backup av data. (Ryan, 2013, s.1.)

Slik som det er med nåværende løsninger så må brukere/kunder av CC tjenester stole på at data som har blitt lagret i en skyløsning ikke forsvinner eller blir tuklet med. De må også stole på at tilbyderen ikke snoker i data, ettersom tilbyder har full tilgang til informasjon som brukere måtte ha i systemet. Google Docs krypterer for

eksempel både datatrafikk og lagret data, men de betjener også krypteringsnøkler og dermed har ikke brukere direkte kontroll over hvem som har tilgang på deres data. Når data-konfidensialitet må garanteres kan løsningen være å kryptere data før det når skyen.
(Samarati, Vimercati, n.d., s.4)

I følge M.D Ryan og Samarati/Vimercati kan problemet med snoking på lagret data i skyen løses med kryptering. Ryan nevner blant annet at kryptering av data hvor brukeren beholder nøklene er løsningen dersom skyplattformen utelukkende skal brukes til lagring. Problemet med konfidensialitet er dog ikke enkelt å løse dersom skyløsningen skal behandle og beregne data; eksempelvis gjennom beregning av finans, dokumentbehandling, modellering av nettverkstrafikk. I slike tilfeller må tredjeparter ha en viss adgang til data. (Ryan, 2013, s.2.)

Det finnes også andre måter å sikre data i skyen på.
Man kan garantere data-integritet ved å benytte “Signature Schemes”. Dette gjør at endringer og modifiseringer av data blir oppdaget. (Samarati, Vimercati, n.d., s.5).
Vi ønsker å forklare “Digital Signature schemes” litt nærmere: I klartekst vil "En setning" se lik ut hos alle mottakere. Så akkurat som en signatur på et papirdokument, trengs det en mekanisme som skiller mellom beskjeder som er sendt fra forskjellige mottakere.
Derfor bruker man "digital signature scheme". Som består av en algoritme som genererer nøkler, en signeringsalgoritme og en verifiseringsalgoritme.
Ingen motpart uten en privat nøkkel kan generere en beskjed og signatur som passerer verifiseringsalgoritmen. Det finnes forøvrig to typer "signature schemes". En som må vite hva beskjeden er i forkant for å kunne verifisere signaturen, og en annen hvor målmeldingen er utdata fra verifikasjonsalgoritmen, slik at meldingen ikke trenger å bli sendt med signaturen.
(Tilborg, Jajodia, 2011, s. 344)

5.4 Ulike typer skytjenester

Det er vanlig å dele skytjenester opp i tjenestemodeller. De tre vanligste er:

- Programvare som tjeneste (software as a service - SaaS), som er en modell for leveranse over et nettverk hvor kunden benytter leverandørens applikasjon(er) på en nettsky-infrastruktur. Kunden har i utgangspunktet ikke kontroll over verken applikasjoner, nettverk, servere, operativsystemer eller lagringsmuligheter.
- Plattform som tjeneste (platform as a service - PaaS), hvor kunden innfører applikasjoner utviklet/kjøpt av kunden i leverandørens nettsky-

infrastruktur gjennom å benytte programmeringsspråk og verktøy støttet av leverandøren. Kunden har kontroll over egne applikasjoner, men har ikke kontroll over nettverk, servere, operativsystemer eller lagringsmuligheter.

- Infrastruktur som tjeneste (infrastructure as a service - IaaS), som gjelder levering av datainfrastruktur som en tjeneste over et nettverk. Kunden har kontroll over relevante applikasjoner, servere, operativsystemer og lagringsmuligheter, samt i noen tilfeller visse elementer i nettverket (for eksempel på brannmursiden).

Skytjenester kan i tillegg deles inn i leveransemodeller som:

- Allmenn tilgjengelig sky (public cloud), hvor skytjenestene gjøres tilgjengelige av leverandøren for alle kunder.
- Privat tilgjengelig sky (private cloud), hvor skytjenestene gjøres tilgjengelige kun for de virksomheter som skytjenestene skal gjelde for. Her vil miljøet/miljøene som skytjenesten leveres fra, typisk dedikeres til den enkelte kunde eller en definert kundegruppe. Dette opplegget åpner for større grad av spesifikke kundetilpasninger enn tilfellet er med modellen for offentlig tilgjengelig sky.
- Hybridsky (hybrid cloud), som kan være en blanding av modellene over. (Datatilsynet, n.d)

Private Cloud er en sky-modell som fokuserer på et tydelig og sikkert skybasert miljø. «Resource pool» er kun tilgjengelig av en enkelt organisasjon eller en enkelt klient. I motsetning til «public clouds» samler «private clouds» ressurser fra et distinkt «pool» av fysiske datamaskiner, som kan «hostes» internt eller eksternt. Det kan nås på tvers av privat-leide linjer eller sikre krypterte forbindelser via offentlige nettverk. Denne modellen er ideell for lagring og behandling av private data, da man kan dra nytte av fordelene ved CC, som for eksempel ressursfordeling ved forespørsel («on-demand»). Tilgang kan begrenses til tilkoblinger fra bak organisasjonens brannmur, dette betyr at det vil fungere på samme måte som et lokalt nettverk eller et intranett. Noe som bare gjør det tilgjengelig for ansatte med riktig legitimasjon. Private skyer er imidlertid mer effektive enn tradisjonelle lokalnettverk (LAN), da de minimerer investeringen til ubrukt kapasitet. (Interoute, 2015, 04.12)

6 Implementasjon

I dette kapittelet ønsker vi å dekke det tekniske aspektet under gjennomføringen av prosjektet, hvor vi går litt mer i dybden på vår tekniske løsning.

I møte den 07.02.17 før implementasjonsfasen påbegynte, satt gruppen seg ned for å formalisere og dokumentere ønsket arbeidsflyt gjennom prosjektet. Dette for å unngå misforståelser, og for å ha et referansepunkt ved utfordringer. Dette er dokumentert i vedlegg 7: Arbeidsflyt utvikling OSecurity.

6.1 Generell teknisk utførelse

6.1.1 Innlogging og registrering

Under kapittelet om stack og teknologi ble det lagt frem hvordan Amazon Cognito fungerer, og implementasjonen av dette i vårt prosjekt gikk smertefritt. Fra eksempel-applikasjonen som AWS genererte, var det begrenset med endringer som vi trengte å gjøre for å inkorporere dette i vårt prosjekt, hovedsakelig var det snakk om å tilpasse grensesnittet og tilpasse språk og format til Norsk.

6.1.2 Aktivering og deaktivering av systemet

Under implementasjon av kommunikasjon mellom mobilapplikasjon, AWS og Raspberry Pi tok vi i bruk AWS IoT og kommuniserte over MQTT. Dette krevde at vi skreddersydde tilgangskontroll og ressursstyringen fra AWS-konsollen, samt en god del utbedringer på eksempelet som AWS viste til.
I dokumentasjonen til Amazon var det kun vist til hvordan man kunne bruke MQTT og AWS IoT med uautoriserte brukere, noe som ikke møtte våre krav til tjenesten. AWSCredentialsProvider[3] ble brukt for å koble til MQTT-megleren (AWS IoT) i utgangspunktet, noe som vi trodde skulle være tilstrekkelig for å få tilgang til ressurser, siden det fungerte smertefritt for andre ressurser som AWS S3. Etter feilsøking med CloudWatch viste det seg at den ble nektet tilkobling grunnet en “auth error”. Det viste seg da at CredentialsProvider kan knyttes opp mot såkalte tokens[4] som blir generert etter brukere er autentisert, men ikke nødvendigvis lagrer denne sikkerhetsinformasjonen til senere bruk. Så ved å hente ut tokens til et HashMap (HM), og så sette HM til CredentialsProvider, så gikk kommunikasjonen gjennom. På dette stadiet var kommunikasjonen fungerende, som gjorde at fokuset kunne gå over på funksjonalitet og forbedring av løsningen.
Så hvordan fungerer det, jo, forutsatt at terminalen er koblet til, vil appen først sende en “check” over MQTT som sjekker om terminalen er armert eller desarmert. Måten dette gjøres på er at terminalen hele tiden abonnerer på kanalen /osecurity/fromapp, som mobilapplikasjonen publiserer meldinger til. Når terminalen mottar “check”, vil

[3] AWSCredentialsProvider er et interface for å hente ut AWS credentials
[4] En ID token, Access token og Refresh token. ID token er den som benyttes til å bekrefte identitet

den kontrollere om tråden for alarmsystemet er aktiv. Basert på om tråden kjører eller ikke så sender terminalen en beskjed tilbake på en kanal vi kaller /osecurity/fromterminal, som mobilapplikasjonen er abonnert til. Basert på responsen vil aktiveringsknappen på telefonen enten vise “Aktiver” eller “Deaktiver”, og basert på knappens status publiserer den enten en aktiverings- eller deaktiveringssignal til /osecurity/fromapp når knappen trykkes. For å hindre at hyppig trykking av knappen skaper feil i denne prosessen vil alltid appen lytte til en bekreftelse fra terminalen før den bytter state på knappen, og mens den venter på bekreftelse kan man ikke trykke på knappen på nytt. For å redusere problematikken med nedetid satt vi opp vår MQTT-klient med clean_session = false, slik at selv om klienten er frakoblet så vil beskjedene som sendes over kanalen bli lagret av MQTT-megleren og være tilgjengelig for klienten når den kobler til på nytt.

6.1.3 Lagring og brukerdata

For lagring og brukerdata benytter vi oss av AWS S3. Et av våre viktigste formål med bruk av S3 er lagring av stillbilder fra Raspberry Pi, som hentes ut i mobilapplikasjonen. På terminalsiden blir kamera aktivert og stillbilde tatt hvert 15. sekund så lenge systemet er aktivert, og for hvert bilde som blir tatt blir det lastet opp en ny versjon i S3, noe som gjør at vi slipper overflødig data. Det blir gjort ved at bildet først blir lagret lokalt, og deretter lastet opp til en såkalt “bucket” i S3, med et unikt navn, som blir kalt en key. Key defineres selv når man laster opp bildet.
På mobilapplikasjonen bruker man AWSCredentialsProvider til å få tilgang til S3, og deretter spesifiserer man hvilken bucket og hvilken key man ønsker å hente ut. Filen man henter blir deretter satt til et ImageView[5] på forsiden i applikasjonen, i tillegg settes det informasjon under bildet som viser når det siste bildet ble tatt og/eller hentet ut. Dette skjer automatisk hver gang forsiden lastes inn, og kan også kjøres på en tidsinnstilt “scheduled task” i Java, hvor intervallet styres av brukeren selv via. innstillinger.

Bruken av S3 er også tiltenkt for å lagre bilder og videoopptak som blir tatt når bevegelse oppdages. Disse filene skal være unike og lagres over tid, slik at brukeren får tilgang til vedlagt media på alarmvarsler i ettertid via. det tiltenkte arkivet i mobilapplikasjonen.

6.1.4 Push- og systemvarsling

Ved bruk av Google Firebase er det implementert enkle systemvarsel som kommer frem på enheten hver gang bevegelse oppdages av sensor på terminalen.

[5] Et grafisk XML-element for bildevisning i Android

For å gjøre dette er terminal autentisert til Firebase ved hjelp av en unik server API-key som google genererer når Firebase settes opp. Appen autoriseres også ved en Android API-key på samme måte, og når appen kobler seg til firebase på oppstart, vil en Firebase registrerings-id genereres. Denne registrerings-IDen er en unik identifikator som ikke forandres før enten appen avinstalleres, logges inn med på en annen enhet, eller om applikasjonsdata blir slettet / resatt på enheten.

For å sende push-varsler bruker terminalen denne unike Firebase identifikatoren når den sender varsler. For å få tilgang på denne så sendes denne over MQTT når appen startes, og settes inn til en variabel på terminalen. På denne måten vil alltid terminalen ha tilgang til den nyeste Firebase identifikatoren, i tilfelle den endres. Ved en senere iterasjoner tenker vi å ha en kontinuerlig sjekk på at terminalen har token tilgjengelig slik at push-varsler alltid vil fungere som tiltenkt. I tillegg er dette foreløpig kun push-varsler som går til systemet, og vi ønsker senere å passe på at disse varslene også blir fremvist inne i appen. Vi skal gjøre dette ved å sende en vanlig push-melding i tillegg til systemnotifikasjonenen, som vises som “uleste varsler” eller lignende, som en pop up i appen, som videre tar deg til informasjon om dette varselet, og for eksempel viser bilde og video som terminalen lastet opp til S3.

6.1.5 Streaming og video

For brukerhistorie 5 ble det tatt en del viktige beslutninger i gruppa. Først og fremst fant vi ut av at å kjøre streaming gjennom AWS vil føre til løpende kostnader grunnet behovet for å holde oppe en EC2 instance som kan hoste streamen. Dermed ble det gjort research for å finne alternative løsninger for et kostnadsfritt alternativ, som ikke er ideelt med tanke på at vi ønsker å holde oss fortrinnsvis til AWS sine moduler med tanke på sikkerhet og administrative årsaker, deriblant muligheten for portere ut løsningen til en annen plattform. Denne alternative løsningen blir derfor midlertidig, og kun for å demonstrere funksjonaliteten.

Løsningen for denne brukerhistorien ble dermed som følger: på Raspberry Pi ble det installert UV4L, som muliggjør “per camera” streaming server som samtidig ble tilgjengelig gjennom hvilken som helst nettleser over HTTP eller HTTPS protokoller.

I tillegg tilbyr UV4L et web brukergrensesnitt som gjør det mulig å se streamen på ulike måter, og et kontrollpanel som gjør det lar deg styre kamera innstillinger med Video4Linux applikasjoner samtidig som det støtter grunnleggende autentisering for normale brukere og admin brukere. (Streaming Server – (advanced) Projects, n.d).

Nå som streamen fungerer fra Raspberry Pi siden gjennom localhost kunne port-forwarding vært et alternativ for å raskt få opp streamen og demonstrere videostrømming. Dette ville også fjernet et ekstra ledd, men vi har ikke tilgang til Universitetets ruter. Etter videre research løste vi dette med bruk av tjenesten Weaved som funger som en privat brannmur/VPN tilkobling som er et sikkert

alternativ til port-forwarding. Til slutt, for å fremvise streamen tok vi i bruk klassen WebView på android applikasjonen som da er en klasse for fremvisning av web innhold.

"*Weaved acts as your private firewall/VPN connection service to all your devices and desktops. Weaved creates secure connections on-demand, over the Internet, to services running in your private network space like Telnet, SSH, HTTP, SFTP, SCP, RDP and VNC. Weaved is a secure alternative to port-forwarding*" (Weaved, n.d).

6.2 Raspberry Pi

I tillegg til programmeringen under implementasjon, var det nødvendig med konfigurasjon og oppsett av maskinvare for vårt prosjekt. I dette delkapittelet ønsker vi å legge frem noe av det mest relevante.

6.2.1 Biblioteker og konfigurasjon

De viktigste verktøyene vi tok i bruk for konfigurasjon var paho for MQTT, boto for S3 og pyfcm for Firebase Cloud Messaging.

6.2.1.1 Paho

Paho MQTT er et python biblioteksprosjekt av Eclipse Foundation for å koble til en MQTT-megler, publisering av meldinger, og abonnering på kanaler og mottak av publiserte meldinger over MQTT som er skreddersydd for Internet of Things. (Python Software Foundation, n.d)

Et eksempel på hvordan vi satt opp paho-mqtt er at man først definerer en klient, f.eks:

```
import paho.mqtt.client as mqtt
client = mqtt.Client(client_id="0securityPi", clean_session=False)
```

Og deretter kobler til vårt unike AWS IoT endpoint:

```
client.connect("endpointID.iot.eu-west-1.amazonaws.com", port=8883)
```

Videre definerte vi tilbakekall for klienten for kobling, publisering, abonnering og mottak, eksempel på kobling:

```
def on_connect(client, userdata, flags, rc):
    // - Definer det du vil skal skje når den er koblet til her
    eks.
    if rc==0:
        print("Success!")
```

6.2.1.2 Boto

Boto er en Python-pakke som tilbyr forskjellige interfaces for AWS. Den har et vidt spekter av bruksområder, for eksempel nettverking, lagring, applikasjonstjenester, databaser, datakraft, med mer. (Python Software Foundation, n.d)
Vår bruk av boto var rettet mot lagring og brukerdata, Amazon S3. I likhet med paho defineres en klient som kobles opp mot det aktuelle sluttpunktet.
Vårt formål på Raspberry Pi er å laste opp innhold til S3, og det løses ved at klienten finner henholdsvis bucket og nøkkel som skal oppdateres, for deretter å skrive over den eksisterende filen i S3 med den lokale filen.

Et lite utdrag av koden:

```
key = s3client.get_bucket('latest-
snapshot').get_key('updatedPicture.JPG')
key.set_contents_from_filename('/home/pi/OSecuritySnapshots/latestSn
apshot.JPG')
```

6.2.1.3 pyfcm

pyfcm er en Python-klient for Firebase Cloud Messaging for Android og iOS. FCM er den nye versjonen av Google Cloud Messaging (GCM), den har arvet den solide og skalerbare infrastrukturen men har blitt utvidet med nye funksjoner. Med FCM kan man sende varsler til klient-applikasjoner, f.eks. systemvarsler og oppdateringsvarsler, opp til en størrelse på 4 KB.(Python Software Foundation, n.d)

Som nevnt tidligere bruker vi en unik Firebase identifikator (Firebase RegistrationId) til å sende varsler til en spesifikk bruker, ved at applikasjonen sender denne over MQTT til Raspberry Pi. Et eksempel på vår bruk er systemvarsel når bevegelse oppdages av PIR-sensor.

Et utdrag av kode som kjøres når bevegelse oppdages:

```
push_service = FCMNotification(api_key="AIzaSyBxUGqEvrIxL0-5-
wzfhr2EjmHXdQe3vcA")
message_title = "Varsel"
message_body = "Bevegelse oppdaget av bevegelsessensor"
result =
push_service.notify_single_device(registration_id=registration,
message_title=message_title, message_body=message_body)
```

Dette vises som et systemvarsel med applikasjonens logo på toppmenyen i Android, med tittelen fra message_title og beskjeden message_body.

6.2.2 Sensorer og maskinvare

Under utviklingen har vi både benyttet en Raspberry Pi 2 Model B, og den nyere Raspberry Pi 3 som vi anskaffet ved en senere anledning. Systemet er kompatibelt til å kjøre på begge versjoner uten komplikasjoner. PIR-sensoren har 3 koblinger på Raspberry Pi, hvorav en er strømforsyning, en er jording, og en er General purpose input/output (GPIO). Strøm og jording kobles på sine respektive pinner, mens GPIO må kobles på en nummerert GPIO-pinne som vi også definerer i Python på Raspberry Pi. Kameramodulen kobles direkte til en egen CSI-port tilrettelagt for kameraet. Til slutt koblet vi alt sammen i et kabinett som er produsert for å kunne montere PIR-sensor og Raspberry Pi sin kameramodul.

6.3 Github og branching-modell

Under implementasjon og bruka av Github fulgte vi en fastsatt rutine for branching og fordeling. Hver stabile versjon av systemet ble lagt i en Release-branch. For videre arbeid ble denne branchet ut til Development hvor vi jobbet på neste versjon. Ut fra Development hadde vi individuelle brancher for hver enkelt «feature». Når flere features var klare ble de merget sammen i Development, og videre til Release når versjonen var fullkommen. Etter endt implementasjonsfase vil seneste Release-versjon legges ut i Master-branchen på bedriftens Github-repository.f

7 Scrum

Sprintene

7.1 Sprint 1: 10.01.17-10.02.17

Den første sprinten ble i stor grad tatt i bruk som en planleggingssprint. Innledningsvis i prosjektperioden hadde vi et samarbeid med en ekstern oppdragsgiver for å gjennomføre et annet prosjekt. Da dette samarbeidet brøt sammen, gikk vi over til å arbeide med OSecurity som prosjekt, og måtte derfor ta igjen tapt tid, samt planlegge utførelsen av OSecurity prosjektet. Vi bestemte oss for å ta i bruk flere elementer fra systemutviklingsprosessen vi er lært opp i her ved Universitetet i Agder, herunder: brukerhistorier, klassediagram, sekvensdiagram, funksjonsliste og rikt bilde. Her fikk vi også bestemt oss for den ønskelige arkitekturen til prosjektet. Etter at analyse og design elementene var utarbeidet, gikk vi videre til å arbeide med utarbeidelse av skisser for brukergrensesnittet. Vi ble enige om en standard og et utseende på papir, og gikk så videre til å få det samme designet klart digitalt.

Vi fikk også skrevet gruppekontrakt (Vedlegg 8: Gruppekontrakt) som vi utarbeidet i fellesskap for å ha felles retningslinjer og forventninger til prosjektløpet. I tillegg gikk

det en del tid til både selve produktutviklingen (sammensetting av komponenter og innkjøp) samt business delen rundt prosjektet vårt. Her inngikk skriving av forretningside for påmelding til konkurranse, møter med UiA Nyskapning og opplæring innen nyetablering i samarbeid med studenter og foreleser på Music Management master-studiet.

Til slutt laget vi førsteutkast til prosjektplan (Vedlegg 5: Prosjektplaner) basert på informasjonen vi hadde tilgjengelig om prosjektløpet (endelig frist, rapport-frist, inndeling i analyse/design/implementasjon). Helt i slutten av sprinten oppdaterte vi prosjektplanen og kom med andre utkast, her hadde vi lagt inn alt av relevante datoer for semesteret som vi visste om på det tidspunktet.

7.2 Sprint 2: 10.02.17-13.03.17

Den andre sprinten var start-skuddet for implementasjonsfasen ettersom analyse og design var unnagjort. Vi gikk rett på brukerhistorie 1 (innlogging) som da var viktig, i og med at autentisering og autorisasjon er avgjørende faktorer for vår mulighet til ressurs-allokering og tilgangskontroll hos våre brukere. Spesielt med tanke på tilgang til stillbilder, videoopptak og direktestrømming av video fra brukernes hjem. Da vi anså sikkerhet som et viktig element, valgte vi å sette vår lit til Amazon Web Services som har modulene IAM (Identity and Access Management) og Amazon Cognito (Registrering og innloggingstjeneste).

Det gikk også en del tid til etableringen av Studentbedrift, i samarbeid med Ungt Entreprenørskap. Det ble avholdt stiftelsesmøte og papirene ble sendt inn for godkjenning av Brønnøysundregisteret.

Som en teambuilding-øvelse, deltok gruppen også på Hackathon 2017 i regi av linjeforeningene Systematicus og Beta, Kristiansand Kommune og selskapet Egde Consulting. Under Hackathonet utviklet vi en app som skulle knytte sammen folk basert på lokasjonsdata og preferanser. Dette ble også vinnerbidraget, og vi følte at vi som bachelor-gruppe fikk mye ut av erfaringen. Premien var blant annet at hele gruppen fikk en Raspberry Pi 3 hver, som da kom godt med i og med at prosjektet vårt tar i bruk Raspberry Pi. Pengesummen vi vant (5000kr) gikk til etablering av studentbedriften og innkjøp av kamera til Pi så vi hadde to klare prototyper tilgjengelig

Dette var også sprinten vi tok for oss mye av researchen når det kom til lovligheten ved hjemmeovervåkning og hvilke forholdsregler vi som selskap burde ta. Vi utarbeidet også et Business Model Canvas for å kartlegge ideen vår og foretok en innledende kostnadsanalyse for å forsikre oss om at ideen om et modulbasert sikkerhetssystem var lønnsom.

I denne sprinten hadde vi vårt første styringsgruppemøte med veileder Janis Gailis. Her fikk vi tilbakemelding om at det var forbedringspotensiale i estimering og

planleggingsmetodikken vår, spesielt med tanke på at vi estimerte ut ifra hva vi følte vi måtte gjøre, og ikke etter hvor mye tid/ressurser vi hadde tilgjengelig og hva som måtte gjøres. Vi gjennomførte et ekstraordinært sprint-planleggingsmøte og fikk til estimater vi var mer fornøyd med. I tillegg bestemte vi oss for å korte ned sprintene til å være to ukers i stedet for fire ukers, så vi kunne ha bedre kontroll på iterasjonene og kunne planlegge bedre.

Til slutt i sprinten oppdaterte vi prosjektplanen (Vedlegg 5: Prosjektplan) vår basert på den nye informasjonen vi hadde fått, og den nye planen som var lagt i det ekstraordinære sprint-planleggingsmøtet.

Det viste seg at vi hadde overvurdert kapasiteten vår, og innledningsvis forventet å ha fullført brukerhistorie 1-3 i løpet av denne sprinten. Da kun brukerhistorie 1 ble implementert forstod vi at det kunne være noe galt med estimeringen vår, samtidig som vi hadde et håp om at det viktigste grunnlaget var lagt (selve Android applikasjonen med innlogging med AWS(Cognito) brukere) og at vi dermed kom til å klare å ta igjen det tapte.

7.3 Sprint 3: 13.03.17-27.03.17

Den tredje sprinten ble altså den første to-ukers sprinten med den nye estimeringsmetoden. Det var også her vi kom på vår første (og største) utfordring i prosjektet; MQTT. Dette i forbindelse med brukerhistorie 2 (aktivering og deaktivering fra app). Her hadde vi diverse utfordringer. Da dette tok mye tid fra gruppen, følte vi at det stoppet oss alt for mye opp og vi begynte å få en følelse av å henge etter.

Ettersom brukerhistorie 1-3 skulle vært ferdigstilt (i henhold til den første prosjektplanen) i løpet av den andre sprinten, var det en ganske anspent stemning hos gruppen denne sprinten. I tillegg var gruppemedlem Ricky Omland bortreist på ferie en uke, så vi var færre enn normalt sett.

Mot slutten av sprinten løste en av de største utfordringene seg, og vi fikk implementert brukerhistorie 2 til slutt. Den ble ikke helt ferdigstilt, da vi måtte løse utfordringer med multiprosessering på Raspberry siden, men vi ble med nytt håp mot neste sprint.

I tillegg ble vi klare på at vi hadde hatt et for dårlig fokus på testing, så noen av gruppemedlemmene ble satt til arbeid med testing av brukerhistorie 1 for å ta igjen arbeidet og passe på at vi hadde testet de ulike elementene tilstrekkelig. I tillegg hadde vi et fokus på bedriftssiden og hadde møter med andre grundere for å kartlegge hvordan man estimerer forventet tidsbruk på et prosjekt. Dette i forbindelse med at vi fikk en henvendelse om et potensielt prosjekt vi kunne ta på oss til etter bachelor-løpet. Et slikt prosjekt ville vært i tråd med våre ønsker for Studentbedriften, og vi inngikk forhandling om budsjett for prosjektet. Det resulterte i at den potensielle

kunden fikk mye å gjøre, og ønsket å sette prosjektet på hold. Gruppen er usikker på om dette kunne vært unngått, men vår manglende erfaring med slik forhandling og gjennomføring av prosjekter var muligens årsaken til at avtalen falt gjennom.

7.4 Sprint 4: 27.03.17-07.04.17

Sprint 4 var uten tvil den mest hektiske og produktive sprinten vi hadde. I denne sprinten hadde vi også en samling med gruppen ”Familiekalender”) hvor vi gjennomførte brukertesting og arbeidet langt på overtid med prosjektet. Dette fordi gruppen var motiverte for å ta igjen etter tapt tid, med ny motivasjon fra gjennombruddet på brukerhistorie 2. I løpet av sprint 4 så fikk vi fullført funksjonaliteten til både brukerhistorie 2, 3 og 4. Dette i tillegg til mye tid lagt inn i rapportskriving.

Det andre styringsgruppemøtet ble avholdt siste dag i sprint 4, altså 07.04.17. Planen opp mot dette møtet (fra forrige styringsgruppemøte) var å fullføre brukerhistorie 1-4 samt å ha påbegynt brukerhistorie 5. I og med at vi den 06.04.17 fikk påbegynt research til implementasjon av brukerhistorie 5, anså vi målet som nådd.

7.5 Sprint 5: 19.04.17-02.05.17

I sprint 5 var det fokus på å få fullført brukerhistorie 5(live stream), for deretter å kunne arbeide med opprydding av Android-applikasjonen og ferdigskrive rapporten. Vi opplevde noen utfordringer denne sprinten, blant annet med at vi hadde sykdomsfravær grunnet operasjon, men også med selve den tekniske implementasjonen av brukerhistorie 5. Hovedutfordringen var kostnadsaspektet, da vi innledningsvis hadde planlagt å bruke AWS sin CloudFront streaming tjeneste til å sende direkte-video fra Raspberry Pi’en. Det viste seg at det var knyttet kostnader opp mot det å holde oppe serveren, og vi så deretter mot alternative løsninger for å ha en prototype å forholde oss til frem til vi hadde en kostnadsanalyse og midlene til å kunne betjene driftskostnadene.

Dette ble løst til slutt, og vi klarte å få til streaming med flere alternativer, deriblant; YouTube, UStream og en hostingtjeneste kalt Weaved. De ulike løsningene hadde forskjellig responstid og kvalitet, og det ble dermed et spørsmål om hvilken løsning som var mest fordelaktig, selv om det i bunn og grunn kun skulle representere en prototype av den endelige løsningen.

7.6 Sprint 6: 02.05.17-16.05.17

Sprint 6 var opprinnelig tiltenkt den avsluttende sprinten, men i og med at tidspunktet for rapport-innlevering i faget IS-304 var den 16.05.17, avgjorde vi i fellesskap at det

var en bedre løsning å dele det opp og bruke sprint 6 til ferdigstillelse av rapporten mens en post-sprint vil bli tatt i bruk til opprydding av design-elementer og refactoring av kode.

I sprint 6 fikk gruppen strukturert dokumentasjonen som var opparbeidet seg gjennom semesteret, og samlet det i rapporten, samt prioritert hvilke elementer som var viktigst å dokumentere og hvordan fremvise disse elementene best mulig.

7.7 Sprint 7: 18.05.17-01.06.17

Denne sprinten vil bli tatt i bruk til en post-sprint. Vi har et arbeidsdokument hvor vi har listet opp oppgavene som gjenstår før vi er fornøyde med prosjektgjennomføringen. I sprint 7/post-sprinten vil dette være fokus, og resultat vil bli fremlagt på eksamen 29.05.17. I tillegg må vi ha en fungerende versjon klar til "IT-Expo"[6] 23.05.17, så her er det viktig å holde kontroll på versjonene og unngå feil under demonstrasjon. Av denne grunn er det også viktig å ha et fokus på testing i post-sprinten.

[6] I henhold til tradisjonen stiller avgangsstudentene på Bachelorstudiet opp på stand i Vrimlehallen på UiA for å fremvise produktene/resultatene sine.

8 Refleksjon

I dette kapittelet har vi valgt å ta for oss gruppens egne refleksjoner tatt i prosjektets nåværende tilstand og de erfaringene vi har tatt med oss gjennom arbeidet med bachelorprosjektet. For at vi skal kunne utvikle oss videre er det viktig at vi reflekterer over prosjektgjennomføring, utfordringer og hvordan de ble håndtert.

8.1 Delegering av tid

Estimering av arbeidsoppgaver var i starten av prosjektfasen svært vanskelig å beregne tidsbruk og estimater på deloppgaver i sprint backloggen. Dette gjorde at vi raskt ble liggende etter skjema og etterhvert måtte dra brukerhistorier over i videre sprinter, dermed ble det en flaskehals i og med at det ble samlet opp flere brukerhistorier i en sprint enn det var mulig å gjennomføre i henhold til estimatene. Vi klarte likevel å hente oss inn i noen tilfeller ved at diverse brukerhistorier tok langt mindre tid enn først antatt å gjennomføre.

Vi har noen ganger definert overskridelse av estimater som “tapt tid” i visse tilfeller, men vi innser at “tapt tid” ikke nødvendigvis representerer noe kritikkverdig. Tidligere hadde vi en holdning der vi følte at “tapt tid” var tid som vi måtte ta igjen. Vi har også innsett at overskridelse av estimater henholdsvis er naturlig til en viss grad og målet gjennom systemutviklingsprosessen er blant annet å forbedre disse tingene, noe som vi har erfart er at det er demotiverende å overskride estimater, for så å tenke at vi eksempelvis må jobbe “overtid” den neste sprinten.

8.2 Håndtering av utfordringer

Prosjektet har vært omfattende med både store og små utfordringer. Løsningen på utfordringene har blitt håndtert ved at de som har ansvar for oppgavene der det oppstår utfordringer får tildelt ekstra ressurser. Det vil si at man enten får tildelt et annet/ekstra gruppemedlem som er tilgjengelig til å assistere, eller at et gruppemedlem tar over oppgaven. Dermed kan man bli satt på en ny oppgave eller velge en man føler man mestrer. Dette har resultert i at vi har gjennomført prosjektet med dynamiske roller.

8.3 Håndtering av å være vår egen arbeidsgiver?

“Frihet under ansvar”. Vi har lært mye om hvordan det er å styre en bedrift, både i forhold til å avholde styringsmøter, fordele formelle titler og dermed ansvarsområder, i tillegg til hvor mye arbeid det faktisk ligger i å oppnå suksess med en IT/Software bedrift. Vi har valgt å bruke mye tid på saker relatert til studentbedriften, som eksempelvis har vært møter med Ungt Entreprenørskap og UiA Nyskapning så vel som styremøter og møter med potensielle kunder.

Problemet med å være egen arbeidsgiver er at det i ettertid er vanskelig å reflektere rundt om vi har vært selvkritiske i stor nok grad. Det vi vet vi er fornøyd med i etterkant er at vi føler vi har vært i stand til å stille omfattende og høye krav til oss selv, samtidig som vi føler at vi har en eierskapsfølelse til prosjektet at vi har lagt ekstra innsats fordi det er noe vi brenner for.

8.4 samarbeid med veileder, medstudenter og Ungt Entreprenørskap

Selv om vi spesifiserte i (Vedlegg 4: Utfyllende scrum logg, sprint 3, retrospekt) at gruppens terskel for å kontakte veileder burde senkes, føler vi i ettertid at dette er absolutt en ressurs vi ikke brukte nok. Veileder sitter på bred kompetanse og i mange av situasjonene der vi har stått fast, valgte vi å løse dem ved å bruke mer tid og ressurser, fremfor å ta kontakt med veileder for råd. Dette har medført at vi som regel løste utfordringer på egenhånd, men på kostnad av mye tid som vi gjerne skulle hatt tilgjengelig til øvrige oppgaver nå som det nærmer seg slutten. I ettertid er det lett å se at vi kunne gjort ting annerledes her, men i de tilfellene vi har brukt veileder med tanke på møter og løpende forespørsler har vi fått gode bidrag til prosjektgjennomførelsen samtidig som veileder alltid har disponert tid til oss ved behov.

Samtidig som veileder har vært en god ressurs for prosjektgjennomførelsen har innspill fra medstudenter i IS-304 forelesninger vært til stor hjelp for gruppa. Det at medstudenter har kunnet bidra så positivt for prosjektets del har medført at vi ved flere anledninger har kommunisert med andre grupper og hatt kodeveld sammen.

I forbindelse med at vi har fått en unik mulighet til å gjennomføre et “utradisjonelt” bachelorløp, ved å starte egen studentbedrift og samarbeide med Ungt Entreprenørskap. De har hjulpet med gode råd og tips til hvordan vi kunne styre prosjektet og bedriften på en god måte gjennom å være bevisstgjøre oss på de ulike rollene og kvalitetene hver enkelt av oss besitter.

9 Oppsummering og resultat

I henhold til egne tidsestimater og normert arbeidsbelastning for 20 studiepoeng på 26 timer per gruppemedlem per uke, har vi lagt ned tilstrekkelig med ressurser for å møte kravene fra instituttet og arbeidsgiver, med faste arbeidstider fra 10-16:00 hver virkedag, med et snitt på ca. 30 timer per uke per gruppemedlem. Med veileder har vi hatt de obligatoriske styringsgruppemøtene (ref. Vedlegg 9: Referat møte med veileder Janis Gailis 02.02.17), i tillegg til å ha noen ekstra møter for generell veiledning med rapport og den tekniske løsningen, noe vi mener har fungert godt, og har bidratt til å øke den sammenlagte kvaliteten på endelig oppgave og produkt.
I henhold til kravspesifikasjon og definert minsteverdiprodukt (MVP) har den endelige løsningen innfridd kriteriene satt av “must have” brukerhistorier. OSecurity er ved endt prosjekt en velfungerende prototype som er klar til å videreutvikles og gjøres klar til produksjon.
På mobilapplikasjonen kan man registrere en Cognito-bruker. Man får deretter tilsendt en verifikasjonskode på e-post som må fylles inn. Etter dette kan brukeren logge inn på applikasjonen.
Videre kan man på hjemmesiden i applikasjonen aktivere og deaktivere systemet ved å trykke på en knapp.
Når man laster inn applikasjonen vil automatisk siste opplastede stillbilde fra Raspberry Pi hentes inn på mobilapplikasjonen og fremvises på hjemmesiden. Under bildet vises det når siste bilde ble hentet ut.
Når en bevegelse oppdages av alarmsystemet vil man få et push-varsel på enheten der applikasjonen er installert. Det er ikke nødvendig at applikasjonen kjører for at meldingen vises. Når man trykker på visningen kommer man rett inn til hjemmesiden i applikasjonen. Ved å trykke på “Start stream” knappen i applikasjonen vil man starte en direkteoverføring fra alarmsystemet. Når streamen er startet kan man stoppe den ved å trykke på “Stopp stream”, og da vil applikasjonen gå tilbake til å vise frem siste stillbilde. Vi har utover dette implementert push-varslinger når systemet aktiveres og deaktiveres, i tilfelle noen tukler med systemet lokalt eller fra en annen enhet.
Veien videre med OSecurity er på kort sikt forbedring av den eksisterende løsningen i en post-sprint frem mot IT-expo 23 mai, og videre frem mot eksaminering og utspørring den 29 mai, hvor det fortløpende vil bli vurdert om vi har kapasitet til å løse flere oppgaver fra backloggen. På et lengre perspektiv ser vi potensielt på dette som en løsning som kan videreutvikles for smart-hjem teknologi og det stadig voksende IoT, for eksempel ved å utvikle en komplett løsning for hjemmestyring og sikkerhet.
På lengre sikt ser vi for oss at bedriftsaspektet, OSecurity SB, vil bli tatt videre, i håp om å kunne etablere et eget AS, og videre drive en gründervirksomhet.

Referanseliste

Android Developers (n.d). Material Design for Android. Hentet fra: https://developer.android.com/design/material/index.html

Barr, J. (2015, 08.10), AWS Mobile Hub – Build, Test, and Monitor Mobile Applications, hentet fra: https://aws.amazon.com/blogs/aws/aws-mobile-hub-build-test-and-monitor-mobile-applications/

Beust, C. (2014, 11.05). The pitfalls of Test-Driven Development. hentet fra: http://beust.com/weblog/2014/05/11/the-pitfalls-of-test-driven-development/

Budhabhatti, M. (2008, January), Test-Driven Development and Continuous Integration for Mobile Applications, hentet fra: https://msdn.microsoft.com/en-us/library/bb985498.aspx

Business Model Canvas. (2017, May 12). In Wikipedia, The Free Encyclopedia. Retrieved 18:22, May15,2017, from https://en.wikipedia.org/w/index.php?title=Business_Model_Canvas&oldid=780086295

Enhanced flow. [Bilde] (2015) Hentet fra: https://aws.amazon.com/blogs/mobile/understanding-amazon-cognito-authentication-part-4-enhanced-flow/

FAQ. (n.d.). Retrieved May 12, 2017, from https://www.weaved.com/faq/

Google Firebase (n.d). Firebase Test Lab for Android Robo Test. Hentet fra: https://firebase.google.com/docs/test-lab/robo-ux-test

Kanban (development). (2017, May 08). Retrieved May 12, 2017, from

https://en.wikipedia.org/wiki/Kanban_(development)

Laudon, K. & Traver, C. (2016),*E-commerce 2016: Business, Technology, Society, 12'th*

edition, Pearson

Mather, T., Kumaraswamy, S., & Latif, S. (2010). Cloud security and privacy: An enterprise

MoSCoW method. (2017, May 10). In Wikipedia, The Free Encyclopedia. Retrieved 18:16, May 15, 2017, from https://en.wikipedia.org/w/index.php?title=MoSCoW_method&oldid=779721559

Pierangela Samarati, Sabrina De Capitani di Vimercati. (n.d) Cloud Security: Issues and Concerns. chapter. 3.1 Protection of data at rest, p. 4. hentet [10.05.2017] fra http://spdp.di.unimi.it/papers/sd-cloud_security.pdf

Planning poker. (2017, March 12). In Wikipedia, The Free Encyclopedia. Retrieved 20:09, May 15, 2017, from https://en.wikipedia.org/w/index.php?title=Planning_poker&oldid=769955265

Python Software Foundation (n.d). paho-mqtt 1.1. Hentet fra: https://pypi.python.org/pypi/paho-mqtt/1.1

Python Software Foundation (n.d). boto 2.46.1. Hentet fra:
https://pypi.python.org/pypi/boto
Python Software Foundation (n.d). pyfcm 1.2.9. Hentet fra:
https://pypi.python.org/pypi/pyfcm/
Ryan, M. D. (2013, 09). Cloud computing security: The scientific challenge, and a survey of solutions. *Journal of Systems and Software, 86*(9), 2263-2268. doi:10.1016/j.jss.2012.12.025

QuickBooks, n.d., The Most Popular Cloud Platforms, hentet 02.05.2017 fra:

http://quickbooks.intuit.com/r/product-services/popular-cloud-platforms/

Scrumdesk (n.d) Hentet fra:
https://www.scrumdesk.com/

Streaming Server – (advanced) Projects. (n.d). Linux-projects.org. Hentet 15.05.2017 fra:
https://www.linux-projects.org/uv4l/tutorials/streaming-server
The Economist. (2015, 31.10), The great chain of being sure about things, retrieved 2016, 24.11 from: http://www.economist.com/news/briefing/21677228-technology-behind-bitcoin-lets-people-who-do-not-know-or-trust-each-other-build-dependable
Tilborg, H. C., & Jajodia, S. (2011). *Encyclopedia of cryptography and security*. New York: Springer. doi:10.1007/978-1-4419-5906-5 hentet [10.05.2017] fra

http://link.springer.com/referenceworkentry/10.1007/978-1-4419-5906-5_17

What is a Private Cloud? (2015, December 04). Retrieved May 10, 2017, from http://www.interoute.com/cloud-article/what-private-cloud

Vedlegg 1 - Uttalelse fra oppdragsgiver

I og med at vår gruppe har etablert studentbedriften ”Osecurity SB”, ORG nr. 918 590 188, er vi dermed vår egen oppdragsgiver. Likevel har vi utarbeidet en uttalelse på et så objektivt grunnlag som mulig, med hvordan vi opplevde student-gruppen ”SaltMineWorkers” og produktet de produserte.

Uttalelse fra OSecurity SB

Oppdragsbeskrivelse
Oppdraget som skulle utføres var å arbeide videre med OSecurity sikkerhetsprosjektet i sin helhet, altså å videreutvikle selve sikkerhetssystemet på Raspberry Pi i Python, samt utvikle en Android-applikasjon som sluttbruker skal ta i bruk.

Påkrevd funksjonalitet:

- Innlogging på mobilapplikasjon
- Aktivering og deaktivering av systemet fra mobilapplikasjon
- Opplasting av bilder fra Raspberry Pi til skydatabase som fremvises på mobilapplikasjon
- Push-notifikasjoner ved oppdagelse av bevegelse
- Direktevisning av video fra Raspberry Pi til mobilapplikasjon

Inntrykk av arbeidet til gruppen
Vårt inntrykk som daglig leder og styreleder av bedriften er at gruppen arbeider mye, hardt og strukturert. Det er en del de fremdeles ikke kan, men til å være en student-gruppe som kun har fått innføring og noe praksis med systemutvikling, mener vi at de overgår våre forventninger for hva som kunne bli produsert av gruppen.

De kunne gjerne ha fokusert mer på testing og det å ha ryddig kode, men gjennom de tre styringsgruppemøtene vi har hatt gjennom prosjektet har de forklart godt for seg når det kommer til hvorfor dette ble nedprioritert. Dette forklarer, men unnskylder ikke, noe mangelfull teknisk testing av produktet.

Likevel er vi i OSecurity SB overbevist om at høyere fokus på testing ville medført mye tid på kunnskapsheving, undersøkelser og innføring av test-elementer. Dette ville dermed sannsynligvis gått på bekostning av funksjonaliteten til produktet, og med tanke på at produktet enda er i produktutviklingsfasen så er funksjonalitet et viktigere element enn feilfri teknisk implementasjon i dette stadiet. Dermed opplever vi at gruppen har tatt hensyn til våre ønsker og gjort så godt de kan.

Generell tilfredshet med gjennomført prosjekt/produkt
Det totale inntrykket vi sitter igjen med etter gjennomført prosjekt er at UiA står med mange ressurssterke studenter, og at det har vært en stor ressursbesparelse for oss å ta i bruk studentene samtidig som vi har fått gitt dem verdifull erfaring av å gjennomføre et prosjekt i en slik skala.

Vi ønsker gjerne å benytte oss av muligheten for å ha en Bachelor-gruppe igjen i fremtiden, og er takknemlige for arbeidet gruppen "SaltMineWorkers" har gjort for oss.

Dato	Daglig leder	Dato	Styreleder
15/05-17	Robin A. R. Myrhol	15/05-17	Sondre Fløtten

Vedlegg 2 - Selvevaluering

Gruppeevaluering

I løpet av semesteret har gruppens samarbeid fungert godt, og alle medlemmene har respektert rammene og kravene som ble satt. Vi har i utgangspunktet operert med faste oppmøtetider fra 10-16:00 hver virkedag, med noen unntak i form av selvstendig arbeid med statusrapportering påfølgende oppmøte.
Ved å sette faste rammer for oppmøte og arbeidstider har vi unngått å ta de tradisjonelle skippertakene som ofte blir tilfelle i slike prosjekter.

Spesielt stolte er vi som gruppe over hvordan vi har klart å utnytte de enkelte gruppemedlemmers styrker og støttet opp mot hverandres svakheter. Dette både i form av kompetansedeling og opplæring fra noen av de mer teknisk sterke i gruppen, men også ved at de mer business-orienterte har fått blomstre i arbeidet med studentbedriften.

Sondre Flovik

Mine bidrag har vært varierte, spesielt innledningsvis under analyse og design hadde jeg et relativt overordnet fokus på det helhetlige. I senere sprinter har jeg hatt hovedansvaret for utviklingen av den tekniske løsningen, blant annet ved å kartlegge oppgavene som var nødvendige for å få implementert den nødvendige funksjonaliteten for å godkjenne de forskjellige brukerhistoriene. Under utviklingen har jeg primært hatt ansvar for implementasjon av brukerhistorie 1-4 for Android og Raspberry Pi, samt administrasjon av skyressurser og oppsett av AWS. I bedriften har jeg hatt rollen som Styreleder, og har sammen med daglig leder utført administrative oppgaver i sammenheng med etablering og oppstart av bedriften, samt arrangering av styremøte.

Robin Amir Rondestvedt Moudnib

Mitt bidrag til prosjektet har vært svært allsidig, men hovedsaklig har det vært i form av Scrum Master, utvikler og daglig leder for bedriften. Som Scrum Master har jeg tatt hovedansvaret for gjennomføring av planleggingsmøter samt review og retrospekt i start og slutt av sprintene, samt et overordnet dokumentasjonsansvar. Som utvikler har jeg vært delaktig i sentrale deler av utviklingen som: Raspberry Pi Pub/Sub, Android applikasjon med streaming, administrasjon av Amazon Web Services, m.m. Som daglig leder har jeg vært fokusert på etableringen av bedriften, samt kartlegging av videre planer for bedriften og vært bedriftens ansikt utad i kontakt med potensielle kunder og prosjekter vi vurderer å ta på oss.

Ricky Løtoft Omland

Mitt bidrag har vært dynamisk og stort sett gått litt innpå de fleste aspektene. Noen av hovedfokusene har blant annet vært design av brukergrensesnitt fra tegnebrett til implementasjon, og design av systemarkitektur som diverse modeller som er ment til å være en slags blåtegning for systemet. I tillegg til deler av analysebiten som f.eks.

kravspesifikasjon. Jeg har også vært en av de ansvarlige for implementering av brukerhistorie 5, live-streaming som for det meste baserte seg på konfigurering og utvikling på Raspberry Pi siden, men også utforsking av ulike muligheter på Android siden.

Christian Fredrik Thorne

Har i hovedsak hatt en allsidig rolle i prosjektet der jeg har fått prøvd meg på flere områder i systemutviklingsprossesen ved siden av det å være produktansvarlig i OSecurity. Under prosjektet har jeg jobbet mye med løsningen av brukerhistorie 5, utvikling av bedriftens nettside, design, produktbestilling og montering. Har fått utfordret meg på flere områder og er veldig fornøyd med den generelle forståelsen og kompetanse økningen omfanget av prosjektet har medbrakt.

Erik Oskar Zetterquist

Har hatt hoved ansvaret for forretningsdelen av prosjektet, har også vært aktiv design og utviklingsfasen. Har deltatt på samtlige møter studentbedriften har hatt med ulike aktører som f.eks. UIA nyskapning og Ungtentrepenørskap. Ettersom vi har rullert på arbeidsoppgaver slik at alle får vært med på ulike deler av prosjektet har jeg bidratt teknisk på brukerhistorie 5. Har ellers bidratt med å finne ut av hvordan vi kan løse ulike problemer som har dukket opp under utviklingen. Jeg føler at jeg igjennom dette prosjektet har fått en bedre teknisk forståelse.

Vedlegg 3 – Refleksjon

Ettersom vi allerede har et refleksjonskapittel i rapporten dekker dette vedlegget bare de obligatoriske refleksjonspunktene gitt av kursansvarlig.

Utviklingsavgjørelser

Under utviklingen av prosjektet er vi stort sett fornøyde med hvordan det gikk og valgene som ble tatt. Til tross for dette ser vi at vi kunne revurdert noen av valgene våre om vi skulle gjennomført prosjektet på nytt.
Under utforming av papirprototype og førstegangsutkast av digital designprototype kunne vi i større grad involvert testsubjekter. Noe av grunnen til at vi valgte å ikke gjøre dette var at Android-applikasjoner gjerne følger «Material Design» (nevnt under design i rapporten), og har en fastsatt ramme for navigasjon- og designvalg.
Under teknisk implementasjon kunne vi også vært noe flinkere til å involvere veileder med våre utfordringer. Vi hadde i stor grad fastsatte roller under denne delen av utviklingen, og har i ettertid vurdert om vi burde rullert mer på våre ansvarsområder for å få flere perspektiver på prosjektets helhet.

Prosjektstyring

Gjennom prosjektet brukte vi flere forskjellige metoder for estimering basert på erfaringene vi gjorde oss underveis. I all hovedsak gikk prosjektstyringen over all forventning, i og med at vi fikk gjort en del mer enn vi hadde forventet og håpet på. Likevel er det rom for forbedring, spesielt på området med å budsjettere hvor timene vil gå, og det å faktisk overholde dette. Ved alt for mange tilfeller satt vi igjen og hadde arbeidet med arbeid som ikke passet inn i de ulike oppgavene vi hadde gjort. Dette resulterte i noe feilaktige grafer når vi ønsket å se tilbake på arbeidet vi la inn.

Kvalitetssikring, QA

Vi hadde tidlig i prosjektet en ambisjon om å legge vekt på bruk av testrammeverk og følge en fast rutine på automatisering av testprosedyrer. Dette ble etter hvert nedprioritert grunnet at det var svært ressurskrevende i forhold til hva vi fikk ut av det. For produksjon av denne prototypen prioriterte vi å sette våre ressurser i bruk for produktutvikling og prototyping. Ettersom applikasjonens bruksområde for en MVP innehar relativt få funksjoner, la vi heller om til en manuell testrutine som beskrevet under kapittelet for QA. Vi er fult klar over at dette er en utdatert måte å gjennomføre QA, men følte der og da at dette var en avgjørelse vi måtte ta.

Prosesser

Med den kunnskapen vi har i dag ville vi først og fremst ha brukt mer tid på utvikling og mindre tid på dokumentasjon dersom vi skulle gjennomført et liknende prosjekt i fremtiden. Selv om det er flott å ha mye dokumentasjon, så er det en grense for hva som er nødvendig for prosjektet og hva som er greit å ha for en rapport. I arbeidslivet vil førstnevnte sannsynligvis være viktigere enn sistnevnte, ifølge inntrykket vi sitter igjen med.

Vedlegg 4 - Utfyllende Scrum-logg

Sprint 1

Planning meeting

I sprint 1 var ting fremdeles uavklart når det kom til prosjektgjennomføringen og hvorvidt vi i det hele tatt skulle fortsette med daværende prosjekt eller gå i ny retning med OSecurity. Dermed ble det planlagt innledningsvis for det opprinnelige prosjektet og så ble det gjennomført ny planlegging for gjennomføring av OSecurity prosjektet. I stedet for tradisjonell sprint-planning meeting forsøkte vi å utarbeide en prosjektplan (Vedlegg 5: Prosjektplaner) basert på informasjonen vi hadde tilgjengelig, og fulgte denne etter beste evne. Med tanke på at systemanalyse og design var først, anså gruppen det som for vanskelig å estimere disse elementene nøyaktig, så et overordnet bilde av hvilke oppgaver som skulle gjøres var nok. I og med at gruppen har hatt fast møtetid mellom 10-16 var det viktigste at vi visste hva vi skulle gjøre, og at vi kom i gang.

Sprint backlog

Følgende elementer var i backloggen vi utarbeidet for sprint 1:

Write group contract

System architecture

Product development

Android app-planning

- ***Papirprototype***
- ***Systemanalyse***
- ***Oppsett av stack***

Det kan her bemerkes at under oppgaven “Android app-planning” og “Systemanalyse” tok gruppen i bruk systemutviklingsmetodikk lært i kurset IS-200 (Systemanalyse og systemutvikling). For å prioritere og velge ut hvilke elementer vi anså som nødvendig for prosjektet, hadde vi en samling hvor vi først diskuterte hvilke potensielle elementer som kunne være nødvendige, deretter prioriterte vi dem

i tråd med PlanningPoker og prioriterte nødvendigheten på de ulike på en skala fra 1-6. Der det var uenighet, ble det drøftet ulike fordeler og ulemper ved den gitte modellen/metodikken det var snakk om, og gruppen kom i fellesskap til slutt frem til hva vi ønsket å ta i bruk. Denne prioriteringsprosessen dokumenterte vi og er loggført under ”Vedlegg 10: Systemanalyse-elementer og prioriteringer.”

Sprint retrospekt

I henhold til planen vår (Gant chart first draft) var første sprint tilegnet planlegging og analysefasen. Her skulle vi få satt opp en kravspesifikasjon og få denne godkjent av product owner. I tillegg bestod den første sprinten av å sette opp utviklermiljøene våre, lage brukerhistorier til utviklingen og gjennomføre øvrige systemanalyse og designmetoder for å kartlegge utviklingsprosessen før igangsettelse. Til slutt ønsket vi å få på plass designskisser.

Etter den første uken var det allerede uenighet med product owner, da han begynte å gå tilbake på ordet sitt og ønsket å gjøre fundamentale endringer i produktet vi opprinnelig skulle lage. Gruppen forsøkte etter beste evne å forklare for product owner at vi ikke har kapasitet til å lage alt han ønsket, og at vi burde rette blikket mot en minimalistisk utgave som scope for IS-304, men dette var heller ikke godt nok.

Det beste for gruppen og product owner på det tidspunktet var å bryte og gå hver sin vei. På denne måten kunne vi ihvertfall unngå ytterligere konflikt.

Da det var avklart tok vi samtaler både med veileder Janis Gailis og fagansvarlig Hallgeir Nilsen om våre tanker om det videre løpet for IS-304. Heldigvis hadde vi et tidligere prosjekt vi hadde jobbet med, OSecurity, og vi ønsket dermed å ta det videre for å lage en Android applikasjon til systemet, samt refactore koden for å tillate autentisering og kommunikasjon med cloud (Amazon Web Services). Da vi fikk tillatelse til dette var det tilbake til tegnebordet i planleggingsfasen igjen.

Dette ledet oss til en andre plan (Gant chart second draft) som da tok det nye prosjektet i utgangspunkt og ble lagt opp for å best mulig ta igjen den tapte tiden. Det ble en nokså lik plan i fundamentet ihvertfall, bestående av planlegging og analyse samt design av applikasjonen. En stor forskjell som gruppen har god tiltro til, er involveringen av Ungt Entreprenørskap og opprettelsen av en StudentBedrift gjennom dem. I samarbeid med rådgiver Vemund Ruud (hos Ungt Entreprenørskap) har vi deltatt i forelesning/opplæring og møter når det kommer til oppstart av egen bedrift og hvilke fordeler Ungt Entreprenørskap bringer med seg.

Gruppen ser på dette som en stor fordel og viktig læringsprosess dette avsluttende semesteret av bachelor-graden. Det er ønskelig å holde Vemund oppdatert på

fremgang slik at han i forbindelse med veileder Janis Gailis kan opptre som en slags oppdragsgiver i og med at de stiller spørsmål til fremgang og vi vet av erfaring at Janis har forventninger til resultatet.

Vi var for øvrig også påmeldt på Innovasjon UiA, som er en idekonkurranse hvor man kan bidra med sin ide og komme foran et panel og kjempe om 30 000,-. Vi utarbeidet innmeldingsskjema og forklarte ideen vår, men nådde dessverre ikke opp til å bli finalist (kun fem plasser tilgjengelig av totalt 22 innsendte ideer). Likevel tok vi det som en god lærdom vedrørende hvordan utarbeide en kort business plan og fremheve styrker ved produktet du utvikler som tilfredsstiller et visst behov.

Sprint review

Gjennomgang av backlogg og hva som er produsert

System architecture
Resultat:
Vi fullførte nesten alle tasks her, bortsett fra å skrive/forklare den nye reviderte systemarkitekturen for rapporten. Altså dokumentasjonen. Fullførte oppgaver inkluderer revisjon av arkitekturen, drøfting av muligheter og endelig fastsettelse av den nye modellen.

Product development
Resultat:
Vi fullførte nesten alle tasks her, bortsett fra å undersøke muligheter for backup-strøm og backup-nett (GSM). Da dette ble nedprioritert helt til “Could Have” av brukerhistoriene våre, så valgte vi å ikke fokusere på dette. Fullførte oppgaver inkluderer opprinnelig hardware research og definering av endelige komponenter for det ferdige systemet. Det ble utarbeidet en oversikt over forskjellige typer produkter med forskjellige komponenter (MVP, Nanny-cam versjon, Forelesningsversjon).

Android app planning
Resultat:
Her fullførte vi alle tasks. Dette var kategorien vi hadde gjort best arbeid innen, da vi klarte å produsere brukerhistorier, klassediagram, funksjonsliste og sekvensdiagram. Vi har et rikt bilde fra et tidligere prosjekt vi har valgt å bruke så langt, selv om den muligens må endres på. I tillegg fikk vi laget papirprototyper av grensesnittet og overført noen av dem til digitale skisser. Dette bør jobbes videre på for å ferdigstilles i digital form med fargevalg parallelt mens utviklingen pågår.

Business part
Resultat:

Her var det tre tasks innen kostnadsplanlegging og markedspotensiale samt research som ikke ble gjort. Øvrige fullførte oppgaver inkluderer opplæring i studentbedrift, møte med UiA Nyskapning og Ungt Entreprenørskap, Scrum-Master arbeid og søknadsutarbeidelse til Innovasjon UiA konkurransen.

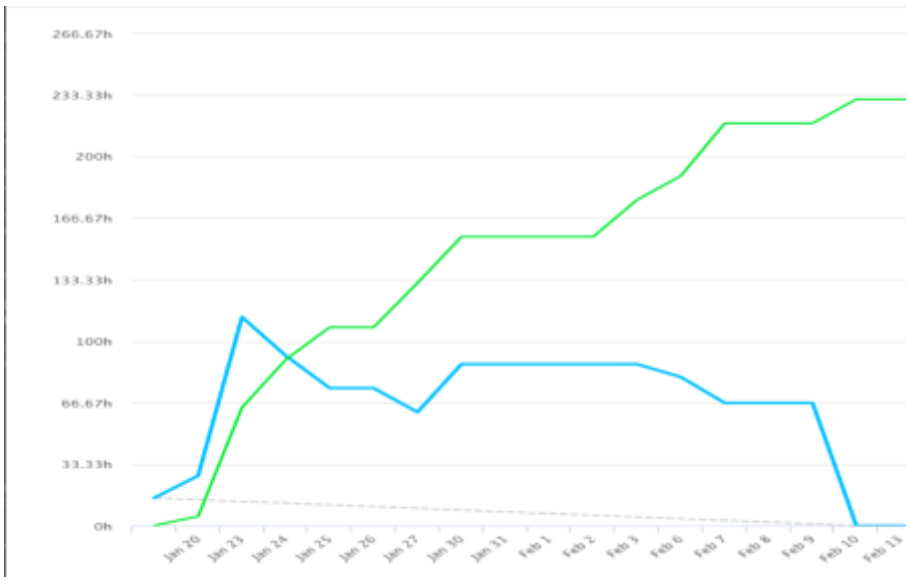
Write group contract
Resultat:
Da dette var en oppgave med tidsfrist da vi skulle bruke samme kontrakt i faget IS-305 så ble denne gjort på tiden og ble fullført med tilfredsstillende resultat.

Report writing IS-304
Resultat:
Her ble alle oppgaver unntatt en gjort, den hvor vi samler sammen dokumentasjonen og setter det inn i en innledende mal for bachelor-rapporten. Dette er vi meget fornøyde med likevel, da flere av disse oppgavene ble formulert underveis i sprinten ettersom de kom opp. Dette kan tyde på behov for mer nøyaktig planlegging tidlig i sprinten. Fullførte oppgaver inkluderer refleksjoner over brudd med product owner, planleggings og analyse forklaringer, sprint retrospect og sprint review.

Burndown chart

I denne bolken på sprint-dokumentasjonen tar vi for oss den resulterende grafen vi får etter endt sprint ved hjelp av Scrumdesk. Det er lagt opp slik at grønn linje er faktisk tidsbruk mens blå linje er gjenstående tid basert på oppgaver. Grafen viser timer, dette kan sorteres i oppgaver også, men vi har valgt å ta ut grafene etter timer da dette gir en god visuell representasjon av arbeidsinnsatsen vår.

Burndown chart for sprint 1:



Sprint 2

Planning meeting

Vi ønsker å jobbe målrettet innledningsvis mot å ferdigstille brukergrensesnitt. Altså, skal det legges opp GUI sider for hver visning og dermed ha enklere/raskere implementasjon av logikk når man har GUI klart.
Det overordnede målet for sprint 2 skal være å fullføre brukerhistorie 1 (Innlogging), brukerhistorie 2 (Armering/desarmering) samt å få ferdigklart produktutviklingen (siste deler er bestilt, det gjelder nå å sette sammen produktet og ferdigstille scripts som blir brukt av appen).
I henhold til vår tredje versjon av Gant-chart så ser vi at vi har satt som plan for andre sprint å gjøre ferdig brukerhistorie 1, 2 og 3, i tillegg til produktutvikling og konfigurering. I og med at vi har eksamen i faget IS-305, samt at noen deltakere av gruppen har øvrige eksamener i februar/mars månedsskiftet, må vi dessverre nedjustere målet for denne sprinten. Vi har likevel tro på at ved å ha gjennomført en så effektiv og omfattende analyse og designfase, vil vi kunne ta igjen tapt tid i implementasjonsfasen av prosjektet.

For å unngå overstyring av prosjektet, velger gruppen å la være å detaljspesifisere alle oppgaver innen brukerhistoriene, og gjør heller innledende estimater for oppgavene når de opprettes direkte i ScrumDesk. Dette har Scrum Master tatt initiativ til, og tar forbehold om at det kan være estimert feil. Forhåpentligvis lærer gruppen og får erfaring med estimering av å jobbe på denne måten.

Sprint backlog

Følgende elementer var i backloggen vi utarbeidet for sprint 2:
Server teknologi
Designe innstilingerside GUI
Brukerhistorie 1
Brukerhistorie 2
Produktutvikling
Business part

Rapportskriving

Øvrig

Sprint retrospekt

Gjennom de største delene av sprint 2 var gruppen positive til hvordan vi lå an i prosjektet generelt. Dette fordi vi hadde progresjon i prosjektet og et gjennombrudd med tanke på autentisering og innlogging gjennom Amazon Web Services.
Likevel, fikk gruppen et tilbakeslag under styringsgruppemøte 1 med veileder Janis Gailis (ref dokumentet), hvor det ble gjort klart at måten planleggingen og estimeringen av oppgaver var gjort på var noe ineffektiv og uoversiktlig. Gruppen fikk skryt for kontrollen vi viste med vår bruk av Gant-chart (ref Gant versjon 3), men fikk også beskjed om at denne kunne vært gjort enda bedre og mer spesifikk kontra vår “åpne” beskrivelse av ukentlige mål. Veileders oppfordring der var å først se vårt totale tilgjengelige timeantall for en sprint opp mot hvilke oppgaver vi har bestemt oss for å ha fullført i løpet av den tiden, og fordele

timene til estimater slik at alle timer av “budsjettet” blir brukt. Veileder mente at på denne måten ville vi ende opp med et positivt utbytte i form av klarhet i hva som skulle bli gjort og hva som faktisk ble gjort med de ressursene vi hadde tilgjengelige.
Gruppen tok i mot tipsene og innså fort at foreslått teknikk ville kunne være bedre for oss enn nåværende teknikk, og hadde derfor et ekstraordinært sprint-planning møte for å ta i bruk den nye teknikken, og vil dermed se an estimatet mot faktisk timebruk i sprint 2 review når denne gjennomføres.
Vi satt også et mål for hva som skal være fullført til neste styringsgruppemøte, og fastsatte at det skal være en representant for Studentbedriften som i første omgang blir Robin A. R. Moudnib, som dermed skal etterspørre status i prosjektet i henhold til løftet han har fått om at brukerhistorie 1-4 er fullført og at brukerhistorie 5 er påbegynt. Dette møtet vil gjennomføres den 7. april i form av et rollespill hvor Robin er oppdragsgiver og de øvrige gruppemedlemmene er utviklerne som har påtatt seg oppgaven.
De erfaringene og tilbakemeldingene vi har fått underveis i arbeidet med prosjektet har resultert i at gruppa har fått en mer effektiv arbeidsstruktur og et mer oppstykket arbeid.
Til slutt har gruppen diskutert de overordnede utfordringene med tanke på tidsbruk og det å ha nok oppgaver til alle, og kommet frem til at vi sannsynligvis vil måtte tenke større når vi skal utarbeide oppgaver. Flere av “OSecurity” oppgavene er såpass presise at de må gjøres sekvensielt og dermed fører til arbeidsopphold for øvrige gruppemedlemmer. En løsning som er tatt opp kan være å utbrodere prosjektet til å ha fokus på OSecurity som bedrift, og at OSecurity er et prosjekt innad i bedriften, men at vi har øvrige bedriftsoppgaver knyttet til daglig drift. Da kan man for eksempel dra inn: Nettsideutforming for bedriften, opprettelse av Facebook-side og administrering av den, design av logo, styremøter og bedriftsvirksomhet, forhandling og planlegging av andre prosjekter (F eks studiehjelpen), med mer.

Sprint review

Brukerhistorie 1 (Innlogging)
Estimat: 40
Faktisk timeantall: 31,5
Her ble vi ferdige. Vi anser oppgaven som godkjent i henhold til akseptansekriterier med forbehold om at en del testing må implementeres, da dette ble bortglemt under planleggingen.

Brukerhistorie 2 (Armering/desarmering)
Estimat: 125
Faktisk timeantall: 71
Denne brukerhistorien fikk vi ikke fullført. En stor del av årsaken var mangel på kompetanse vedrørende AWS og hvordan koble opp en “thing” mot en spesifikk bruker og dermed kunne gi beskjed om å armere/desarmere. Årsaken til at vi gjør dette med en gang er for å slippe ekstraarbeid/refactoring på de senere oppgavene som krever at en spesifikk “thing” er tilknyttet en spesifikk bruker. Dermed tas arbeidet videre i neste sprint.

Produktutvikling
Estimat: 9
Faktisk timeantall: 7.5
Produktutvikling gikk som planlagt iht. innkjøpt utstyr og oppsett. Vi har nå et klart fysisk produkt med oppsatt PIR-sensor og kamera i kabinett og WiFi-dongle for vår Raspberry Pi 2

Model B. Videre utføres samme installasjon for Raspberry Pi 3. Oppsettet baserer seg på det vi avklarte som MVP i sprint 1 (ref.navn på vedlegg)

Rapportskrivning
Estimat: 107
Faktisk timeantall: 62
Rapportskrivning gikk bra etter forholdene, da vi har fått dokumentert mange ulike viktige aspekter ved prosjektet, samt kartlagt hvordan vi skal arbeide videre med dokumentasjonen. I tillegg inngikk det her noen struktureringsoppgaver, konsultasjon med foreleser og et ekstraordinært sprint-planning møte for å omstrukturere planleggingsmetodikken vår.
Noe av målet og hensikten med rapportskrivingen var å få satt opp strukturen som vi anser som viktig i startfasen, slik at det er tilrettelagt for videre arbeid. Det vi ikke fikk fullført var sprint review og retrospekt som ble dyttet over i begynnelsen av sprint 3.

Øvrig (Server-research, business-part, team-building, enhancement of skills, app-planning)
Estimat: 252
Faktisk timeantall: 196
Server-technology (Authentication, research, etc) - 26/85
Settings-Gui 5/5
Business-part 63/59
App-planning 3/4
Enhancement of skills/Teambuilding - 96/96
System architecture - 3/3
Under øvrig fikk vi gjennomført planlagt arbeid, og estimatene våre overensstemmer i stor grad reell timebruk. Estimatene rundt autentisering og server-teknologi bommet en del, men dette grunnet en uvisshet om hvordan dette ble løst med AWS. Det viste seg at dette var en strømlinjeformet prosess som tok langt mindre tid enn antatt.

Generelt
Totalt tilgjengelig: 520
Totalt brukt: 368
Timer for lite: 152
Total tidsbruk under denne sprinten har hovedsakelig to årsaker. Primært var vi ikke flinke nok til å loggføre tidsbruk og legge opp oppgaver for alt vi jobbet med på Scrumdesk, noe som gjør at våre loggførte timer ikke reflekterer reell tidsbruk på en tilfredsstillende måte, og dette er noe vi må ta med oss videre. I tillegg hadde flere gruppemedlemmer ekstraordinære eksamener, i tillegg til at alle gruppens medlemmer hadde eksamen i IS-305, så IS-304 ble noe nedprioritert i en ukes tid på grunn av dette. Vi er forberedt på å legge et større fokus på IS-304 fremover, siden den relativt intense kursplanen i IS-305 nå til en viss grad er gjennomført, i tillegg til at ekstraordinære eksamener er gjennomført.

Brukerhistorie 1 (Innlogging)
Estimat: 40
Faktisk timeantall: 31,5
Her ble vi ferdige. Vi anser oppgaven som godkjent i henhold til akseptansekriterier med forbehold om at en del testing må implementeres, da dette ble bortglemt under planleggingen.

Brukerhistorie 2 (Armering/desarmering)
Estimat: 125
Faktisk timeantall: 71
Denne brukerhistorien fikk vi ikke fullført. En stor del av årsaken var mangel på kompetanse vedrørende AWS og hvordan koble opp en "thing" mot en spesifikk bruker og dermed kunne gi beskjed om å armere/desarmere. Årsaken til at vi gjør dette med en gang er for å slippe ekstraarbeid/refactoring på de senere oppgavene som krever at en spesifikk "thing" er tilknyttet en spesifikk bruker. Dermed tas arbeidet videre i neste sprint.

Produktutvikling
Estimat: 9
Faktisk timeantall: 7.5
Produktutvikling gikk som planlagt iht. innkjøpt utstyr og oppsett. Vi har nå et klart fysisk produkt med oppsatt PIR-sensor og kamera i kabinett og WiFi-dongle for vår Raspberry Pi 2 Model B. Videre utføres samme installasjon for Raspberry Pi 3. Oppsettet baserer seg på det vi avklarte som MVP i sprint 1 (ref.navn på vedlegg)

Rapportskrivning
Estimat: 107
Faktisk timeantall: 62
Rapportskrivning gikk bra etter forholdene, da vi har fått dokumentert mange ulike viktige aspekter ved prosjektet, samt kartlagt hvordan vi skal arbeide videre med dokumentasjonen. I tillegg inngikk det her noen struktureringsoppgaver, konsultasjon med foreleser og et ekstraordinært sprint-planning møte for å omstrukturere planleggingsmetodikken vår.
Noe av målet og hensikten med rapportskrivingen var å få satt opp strukturen som vi anser som viktig i startfasen, slik at det er tilrettelagt for videre arbeid. Det vi ikke fikk fullført var sprint review og retrospekt som ble dyttet over i begynnelsen av sprint 3.

Øvrig (Server-research, business-part, team-building, enhancement of skills, app-planning)
Estimat: 252
Faktisk timeantall: 196
Server-technology (Authentication, research, etc) - 26/85
Settings-Gui 5/5
Business-part 63/59
App-planning 3/4
Enhancement of skills/Teambuilding - 96/96
System architecture - 3/3
Under øvrig fikk vi gjennomført planlagt arbeid, og estimatene våre overensstemmer i stor grad reell timebruk. Estimatene rundt autentisering og server-teknologi bommet en del, men dette grunnet en uvisshet om hvordan dette ble løst med AWS. Det viste seg at dette var en strømlinjeformet prosess som tok langt mindre tid enn antatt.

Generelt
Totalt tilgjengelig: 520
Totalt brukt: 368
Timer for lite: 152

Total tidsbruk under denne sprinten har hovedsakelig to årsaker. Primært var vi ikke flinke nok til å loggføre tidsbruk og legge opp oppgaver for alt vi jobbet med på Scrumdesk, noe som gjør at våre loggførte timer ikke reflekterer reell tidsbruk på en tilfredsstillende måte, og dette er noe vi må ta med oss videre. I tillegg hadde flere gruppemedlemmer ekstraordinære eksamener, i tillegg til at alle gruppens medlemmer hadde eksamen i IS-305, så IS-304 ble noe nedprioritert i en ukes tid på grunn av dette. Vi er forberedt på å legge et større fokus på IS-304 fremover, siden den relativt intense kursplanen i IS-305 nå til en viss grad er gjennomført, i tillegg til at ekstraordinære eksamener er gjennomført.

Brukerhistorie 1 (Innlogging)
Estimat: 40
Faktisk timeantall: 31,5
Her ble vi ferdige. Vi anser oppgaven som godkjent i henhold til akseptansekriterier med forbehold om at en del testing må implementeres, da dette ble bortglemt under planleggingen.

Brukerhistorie 2 (Armering/desarmering)
Estimat: 125
Faktisk timeantall: 71
Denne brukerhistorien fikk vi ikke fullført. En stor del av årsaken var mangel på kompetanse vedrørende AWS og hvordan koble opp en “thing” mot en spesifikk bruker og dermed kunne gi beskjed om å armere/desarmere. Årsaken til at vi gjør dette med en gang er for å slippe ekstraarbeid/refactoring på de senere oppgavene som krever at en spesifikk “thing” er tilknyttet en spesifikk bruker. Dermed tas arbeidet videre i neste sprint.

Produktutvikling
Estimat: 9
Faktisk timeantall: 7.5
Produktutvikling gikk som planlagt iht. innkjøpt utstyr og oppsett. Vi har nå et klart fysisk produkt med oppsatt PIR-sensor og kamera i kabinett og WiFi-dongle for vår Raspberry Pi 2 Model B. Videre utføres samme installasjon for Raspberry Pi 3. Oppsettet baserer seg på det vi avklarte som MVP i sprint 1 (ref.navn på vedlegg)

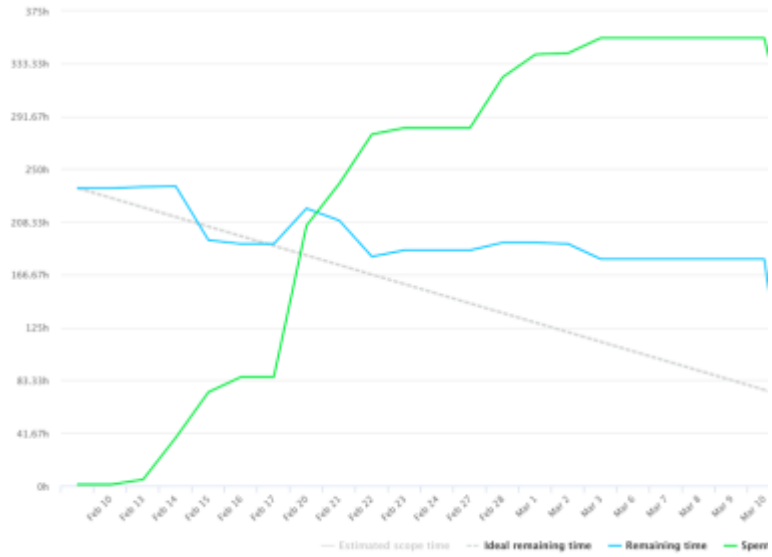
Rapportskrivning
Estimat: 107
Faktisk timeantall: 62
Rapportskrivning gikk bra etter forholdene, da vi har fått dokumentert mange ulike viktige aspekter ved prosjektet, samt kartlagt hvordan vi skal arbeide videre med dokumentasjonen. I tillegg inngikk det her noen struktureringsoppgaver, konsultasjon med foreleser og et ekstraordinært sprint-planning møte for å omstrukturere planleggingsmetodikken vår.
Noe av målet og hensikten med rapportskrivingen var å få satt opp strukturen som vi anser som viktig i startfasen, slik at det er tilrettelagt for videre arbeid. Det vi ikke fikk fullført var sprint review og retrospekt som ble dyttet over i begynnelsen av sprint 3.

Øvrig (Server-research, business-part, team-building, enhancement of skills, app-planning)
Estimat: 252
Faktisk timeantall: 196

Server-technology (Authentication, research, etc) - 26/85
Settings-Gui 5/5
Business-part 63/59
App-planning 3/4
Enhancement of skills/Teambuilding - 96/96
System architecture - 3/3
Under øvrig fikk vi gjennomført planlagt arbeid, og estimatene våre overensstemmer i stor grad reell timebruk. Estimatene rundt autentisering og server-teknologi bommet en del, men dette grunnet en uvisshet om hvordan dette ble løst med AWS. Det viste seg at dette var en strømlinjeformet prosess som tok langt mindre tid enn antatt.

Generelt
Totalt tilgjengelig: 520
Totalt brukt: 368
Timer for lite: 152
Total tidsbruk under denne sprinten har hovedsakelig to årsaker. Primært var vi ikke flinke nok til å loggføre tidsbruk og legge opp oppgaver for alt vi jobbet med på Scrumdesk, noe som gjør at våre loggførte timer ikke reflekterer reell tidsbruk på en tilfredsstillende måte, og dette er noe vi må ta med oss videre. I tillegg hadde flere gruppemedlemmer ekstraordinære eksamener, i tillegg til at alle gruppens medlemmer hadde eksamen i IS-305, så IS-304 ble noe nedprioritert i en ukes tid på grunn av dette. Vi er forberedt på å legge et større fokus på IS-304 fremover, siden den relativt intense kursplanen i IS-305 nå til en viss grad er gjennomført, i tillegg til at ekstraordinære eksamener er gjennomført.

Burndown chart



I sprint 2 gjennomførte vi mange oppgaver og fikk loggført mange timer på bra arbeid. Utfordringen her var at vi ofte slet med å føre timene direkte på oppgavene som var estimert, så vi arbeidet mye, men med begrenset reduksjon av backloggen. Dette fordi det kom opp uforutsette oppgaver som måtte gjennomføres.

Sprint 3

Planning meeting

På bakgrunn av første styringsgruppemøte med veileder Janis Gailis den 27.02.17 valgte vi å ha et ekstraordinært planleggingsmøte den 28.02.17 for å få kontroll på timeestimeringen av oppgaver og hvordan effektivisere den prosessen.

Selv om vi allerede hadde definert opp oppgaver og estimert dem til sprint 2, var det ikke alle nødvendige oppgaver som var gjort, og heller ikke alle tilgjengelige timer som var tildelt. Etter rådgivning fra veileder har vi kommet frem til følgende fremgangsmåte:

1. Sett opp alle oppgaver (oppdelte oppgaver) som kreves for å fullføre en brukerhistorie (inkludert teknisk dokumentasjon og testing)
2. Tell opp antall timer tilgjengelig totalt i den gitte tidsperioden det planlegges for
3. Fordel av total-antallet til de ulike oppgavene

Vi valgte også på bakgrunn av dette møtet å korte ned sprint 3 slik at den varer i 2 uker i stedet for 4, og dermed går fra 10.03.17-24.03.17. Deretter går sprint 4 fra 24.03.17-07.03.17 hvor påfølgende uke er påskeferie-uke.

Sprint 3 ekstraordinært sprint-planning meeting

Resultat:

Brukerhistorie 2: Som bruker ønsker jeg å kunne armere og deaktivere sikkerhetssystemet så jeg kan føle meg trygg.
*Totalt timeantall 01.03.17-03.03.17: 63 (4*5,2*3 siden siste medlem har eksamen)*

Sette opp Raspberry Pi som en “Thing” (hence IoThing) 3

Finne ut hvordan å interacte med Raspberry Pi fra pc eller appen 5

Implementere interaction med Raspberry Pi fra app (koble sammen Thing til Mobile Hub, med tanke på User Pool og brukere) Kun en gitt innlogget bruker kan interacte med thingen. En thing må være knyttet til en bruker. 34

Koble opp knapp som gir hendelse til tingen 5

Endre til at knappen aktiverer/deaktiverer systemet 5

Dokumentere IoT oppkoblingen på teknisk plan (protokoll, hvordan informasjon hentes og leveres og hva slags type requests/svar) 3

Skrive tester for funksjonene vi implementerer i brukerhistorien 8

Antall timer estimert til oppgaver: 63

Brukerhistorie 3: Som en bruker ønsker jeg å kunne se siste stillbilde fra hjemmet, slik at jeg kan få en indikasjon på situasjonen hjemme når det passer meg.

*Totalt timeantall 06.03.17-10.03.17: 130 (26*5)*

Å interacte med "thingen" → Hente ut en fil fra Thing inn i mobile hub 55
Appen interacter med mobile hub og får fremvist fil på hjemmeside 34
Dokumentere filbehandlingen på teknisk plan 3
Skrive tester for funksjonene vi implementerer i brukerhistorien 8
GUI for arkiv 8
Funksjonalitet til arkiv (få inn bildene i liste i GUI fra database i skyen) 21

Antall timer estimert til oppgaver: 129

Brukerhistorie 4: Som bruker ønsker jeg å få push-notifikasjoner når systemet oppdager anomaliteter, for å være oppdatert på hva som skjer når jeg ikke er hjemme.

*Totalt timeantall 13.03.17-24.03.17: 260 (26*5*2)*

Research på AWS SNS for push varsler 55
Aktivering/implementering av et enkelt push varsel 55
Tilrettelegge scripts på Pi til å kunne gi beskjed om anomaliteter til AWS 34
Skreddersy push varsler til å komme ved anomaliteter i sikkerhetssystemet 34
Skrive tester for funksjonene vi implementerer i brukerhistorien 21
Dokumentere push-notifikasjonene på teknisk plan 8

Rapportskrivning øvrige 50

Antall timer estimert til oppgaver: 257

Brukerhistorie 5: Som en bruker ønsker jeg å kunne se video live fra hjemmet, slik at jeg kan følge med på hva som skjer til en hver tid.

*Totalt timeantall 27.03.17-21.04.17: 390 (26*5*3)*

Research (89)
AWS Cloudfront streaming, inkorpoere med AWS IoT & mobile hub (89)
Utforme grensesnitt for streaming (10)
Motta bitstream fra AWS, fremvise i grensesnitt (89)
Koble opp at knapp triggrer av/på med bitstream (34)
Skrive tester for funksjonene vi implementerer i brukerhistorien (55)
Dokumentere, teknisk (21)

Antall timer estimert til oppgaver: 387

Sprint backlog

Retrospect

Gjennom sprint 3 opplevde vi den største stillstanden i prosjektet så langt. Det ble en teknisk krevende oppgave å komme videre, noe som gjorde at vi måtte omprioritere og fordele nye arbeidsoppgaver for å utnytte ressursene våre best mulig.
To av gruppemedlemmene fortsatte for å løse utfordringen rundt brukerhistorie 2, mens de andre ble satt til henholdsvis unit testing, web-utvikling og andre organisatoriske oppgaver tilknyttet studentbedriftens virksomhet.

Grunnet noe manglende logging og feilhåndtering av AWS sin SDK og AWS sine APIer ble det noe utfordrende å feilsøke og rette problemene vi hadde med MQTT-tilkobling for autoriserte brukere. Det var lite ressurser tilgjengelig for utfordringen vi var fast på, og måtte bare fortsette å lese dokumentasjon og benytte oss av spørsmål på stackoverflow. Etter å ha undersøkt AWS sin dokumentasjson og benyttet oss av AWS CloudWatch (loggingsverktøy for AWS-ressurser), klarte gruppen til slutt å lokalisere problemet. Ved å jobbe videre for å løse det som var det antatte problemet, fikk vi til slutt løst opp denne 'flaskehalsen'.

Når vi ser tilbake burde vi kanskje involvert veileder for en samtale som kunne satt oss på rett spor. Vi brukte mye lengre tid enn nødvendig på å lokalisere feilen, og ved å snakke med en ekstern person ville det kanskje løst opp tankegangen vår og hjulpet oss til å løse problemet raskere. Vi bør nok jobbe for å senke terskelen vi har for å kontakte veileder for hjelp og tips, og ikke alltid være fast bestemt på å løse alt uten hjelp.

I tillegg til denne utfordringen ved brukerhistorie 2 var ett av gruppemedlemmene bortreist på en planlagt ferie under halve sprinten, noe vi ikke tok høyde for under planlegging og estimering av sprint 3. Dette gjorde at vi i tillegg til å stå fast, jobbet med 80% av de estimerte ressursene under innspurten av sprinten, og utfallet av dette er jo da at vi ligger noe bak der vi hadde ønsket å være.

Når progresjonen var stillestående fordelte vi i større grad forskjellige arbeidsoppgaver til de forskjellige gruppemedlemmene for å minimere konsekvensen av flaskehalsen i utviklingen. Dette reflekterte vi over etter endt sprint 2 i den tilhørende retrospekten, og vi fikk bruk for dette under sprint 3. Ved å distribuere flere uavhengige arbeidsoppgaver rundt studentbedriften og testing, fikk vi løst noe av utfordringene ved de sekvensielle oppgavene som oppstår ved app-utvikling og programmeringsoppgavene.

Sprint 3 review (Sprint 3: 10.03.17 - 24.03.17)

Sprint 3 var en kortere sprint som resultat av styringsgruppemøte med Janis 27.02.17, dette for å ha bedre kontroll over hvilke oppgaver som skal være løst innen det tidsrommet. Det ble en del utfordringer i denne sprinten relatert til kommunikasjon mellom Raspberry Pi, AWS og Applikasjonen vi utvikler. Dette medførte forsinkelser og vi er dermed bak skjema.

Resultat:

Brukerhistorie 2: Som bruker ønsker jeg å kunne armere og deaktivere sikkerhetssystemet så jeg kan føle meg trygg.

Estimert i planning: 63
Estimert i scrumdesk: 119
Endelig timeantall: 146

Brukerhistorie 2 ble en noe mer kompleks oppgave i forhold til hva vi antok. Ved bruk av MQTT websocket skulle vi sende og motta signaler i applikasjon og på Raspberry Pi (terminal). Med et eksempel-prosjekt fikk vi dette til relativt smertefritt, ved å tillate uautorisert tilgang (uten noen innlogging) til AWS-ressursene våre. Problemet oppstod når dette skulle inkorporeres med autoriserte brukere gjennom Amazon Cognito, da det grunnet manglende feilhåndtering og logging i Amazon's SDK og API'er ble vanskelig å feilsøke at kommunikasjonen ikke gikk gjennom. Ved å sette opp Amazon CloudWatch, et loggingsverktøy for AWS, fant vi ut at det var en autoriseringsfeil. Dette selv om innlogging gikk gjennom og brukeren hadde tilgang til andre AWS ressurser, men ikke IoT over MQTT. Det kom senere frem at dette var fordi AWS Credentials Provider ikke 'lagret' den nødvendige informasjonen til å autorisere MQTT, og vi fikk til slutt løst dette ved å hente ut nødvendige tokens fra brukersession, og sette disse til credentials provider via en hashmap.

Brukerhistorien er enda ikke ansett som gjennomført, da det gjenstår å refactore kildekoden på Raspberry Pi til å kunne kjøre disse to hovedprosessene (systemActive og pub/sub) samtidig.

Brukerhistorie 3: Som en bruker ønsker jeg å kunne se siste stillbilde fra hjemmet, slik at jeg kan få en indikasjon på situasjonen hjemme når det passer meg.

Estimert i planning: 129
Estimert i scrumdesk: 146
Endelig timeantall: 2,5 (Research)
Grunnet den uforutsette kompleksiteten for å få godkjent brukerhistorie 2, kom vi kun til research-fasen til implementasjon av brukerhistorie 3.

Brukerhistorie 4: Som bruker ønsker jeg å få push-notifikasjoner når systemet oppdager anomaliteter, for å være oppdatert på hva som skjer når jeg ikke er hjemme.

Estimert i planning: 257
Estimert i scrumdesk: 207
Endelig timeantall: 0
Ikke påbegynt i denne sprinten grunnet utfordringene på brukerhistorie 2.

Testing brukerhistorie 1
Estimert i planning: 0
Estimert i scrumdesk: 40
Endelig timeantall: 35

Testing er noe vi har lite erfaring med fra før, i hovedsak ønsket vi å starte med å implementere unit tests for brukerhistorie 1. Dette viste seg å være mer utfordrende enn forventet. Grunnen til dette er at vi ikke er helt inneforstått med hvordan vi skal teste de ulike funksjonene ettersom det hentes inn mye data fra API'er og Back-end funksjonalitet iht. kommunikasjon mellom server og klient fra AWS. Dette gjør det utfordrende å bruke standardiserte tester til å teste kode som vi til dels ikke har skrevet selv. En faktor her er mangel på kunnskap om testing, og spesielt mangel på kunnskap iht. testing av nettverkskommunikasjon. Vi har derimot lært oss strukturen på tester og hvordan man utfører testing til ulike formål, men har ikke lykkes i å inkorporere dette til vårt formål. En annen tilnærming til disse testene vil være å tenke nytt, og å opprette nye klasser for prosjektet som tar for seg HTTP kommunikasjon mellom AWS og å teste disse.

Business part
Estimert i planning: 0
Estimert i scrumdesk: 66
Endelig timeantall: 62

Dette ble nedprioritert i denne sprinten grunnet utfordringene vi hadde på brukerhistorie 2 og testskriving for brukerhistorie 1. Likevel klarte vi å skaffe oss relevant kunnskap til denne biten av prosjektet gjennom intervjuer med Frontbyte og Egde Consulting i forbindelse med IS-305 rapport. Disse intervjuene ble også tildelt et visst fokus når det kommer til struktureringen av bedriften vår, og hvordan tilegne seg kontrakter/prosjekter og hvilken godtgjørelsesmodell vi ønsket. Intervjuene ble derfor betraktet som meget nyttige.

Report writing IS-304
Estimert i planning: 0
Estimert i scrumdesk: 94
Endelig timeantall: 89,5
Rapportskrivning til bachelor-oppgaven var noe som opprinnelig ikke var en del av planen for sprinten, og dette er noe vi må jobbe bedre med å estimere som en del av arbeidet. Vi har satt av en del tid til rapportskrivning på slutten av prosjektløpet. Likevel er det viktig for gruppen å dokumentere underveis mens man har stoffet friskt i minne, og det gikk en del tid her til veiledningsmøter og et ekstraordinært sprint-planleggingsmøte.

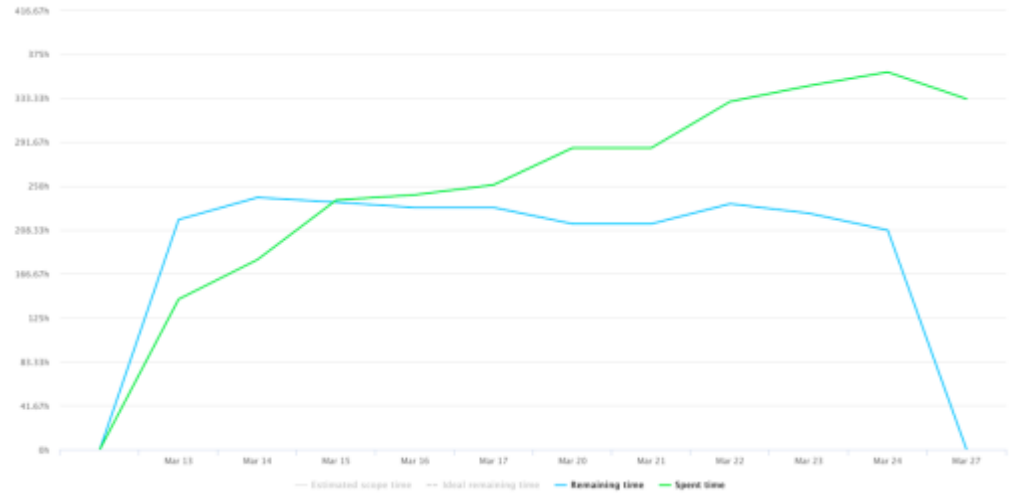
Totalt
Estimert i planning: 449

Estimert i scrumdesk: 672
Endelig timeantall: 335

Totalt tidsbruk er også under målet denne sprinten, dette er noe vi hadde håpet på å estimere bedre med tanke tanke på erfaringene og kunnskapen vi tilegnet oss etter sprint 2. Når vi ser tilbake på sprinten ligger noe av disse manglende timene i at en av gruppemedlemmene var på en planlagt ferie vi ikke tok høyde for under estimeringen.

Likevel ser vi lys i enden av tunnelen i og med at systemet nå er tilrettelagt med de ønskede sikkerhetsinnstillingene og sjekker av autentisering fungerer nå som det skal. Dermed vil vi kunne spare inn tid når vi går videre til brukerhistorie 3 spesielt, men forhåpentligvis også ha en fordel når vi kommer til brukerhistorie 4. Gruppen må nå ta igjen tapt arbeid grunnet eksamen og utfordringer på brukerhistorie 2, og planlegger deretter.

Burndown chart



Sprint 3 sin graf viser at dette var en produktiv sprint, selv om det ble noe tapt tid grunnet et gruppemedlem på ferie samt utfordringer med testing.

Sprint 4

Planning meeting

I og med at det har vært noen utfordringer så langt, og at en del tid har gått med til eksamenslesning i andre fag, velger gruppen å bruke sprint 4 på å ta igjen tapt tid ved å jobbe ekstra. Da vi anser prosjektløpet og det totale antallet timer som vårt "budsjett" så har vi mer å gå på, og vil dermed gjøre vårt beste for å ta igjen det tapte og å få utnyttet alle tilegnede ressurser.

Den overordnede planen for denne sprinten er å ferdigstille det siste på brukerhistorie 2, samt å få gjort brukerhistorie 3 og 4. Det er også ønskelig å påbegynne brukerhistorie 5, men dette kan være noe vanskelig da vi allerede har mer enn nok å gjøre.

*Totalt timeantall sprint 4, 24.03.17-07.04.17: 260 (26*5*2) (Pluss evt ekstra)*

Brukerhistorie 2: Som bruker ønsker jeg å kunne armere og deaktivere sikkerhetssystemet så jeg kan føle meg trygg.
Estimert ferdigstilt: 29.03.17

Gjenstående:
Dokumentasjon på teknisk plan 6
Fikse kodebase på Raspberry Pi, interaksjon for armering/disarmering 6

Antall timer estimert til oppgaver: 12

Brukerhistorie 3: Som en bruker ønsker jeg å kunne se siste stillbilde fra hjemmet, slik at jeg kan få en indikasjon på situasjonen hjemme når det passer meg.

Å interacte med "thingen" → Hente ut en fil fra Thing inn i mobile hub 55
Appen interacter med mobile hub og får fremvist fil på hjemmeside 34
Dokumentere filbehandlingen på teknisk plan 3
Skrive tester for funksjonene vi implementerer i brukerhistorien 8
GUI for arkiv 8
Funksjonalitet til arkiv (få inn bildene i liste i GUI fra database i skyen) 21

Antall timer estimert til oppgaver: 129

Brukerhistorie 4: Som bruker ønsker jeg å få push-notifikasjoner når systemet oppdager anomaliteter, for å være oppdatert på hva som skjer når jeg ikke er hjemme.

*Totalt timeantall 13.03.17-24.03.17: 260 (26*5*2)*

Research på AWS SNS for push varsler 55
Aktivering/implementering av et enkelt push varsel 55
Tilrettelegge scripts på Pi til å kunne gi beskjed om anomaliteter til AWS 34
Skreddersy push varsler til å komme ved anomaliteter i sikkerhetssystemet 34
Skrive tester for funksjonene vi implementerer i brukerhistorien 21
Dokumentere push-notifikasjonene på teknisk plan 8

Rapportskrivning øvrige 50

Antall timer estimert til oppgaver: 257

Forhåpentligvis får vi påbegynt:

Brukerhistorie 5: Som en bruker ønsker jeg å kunne se video live fra hjemmet, slik at jeg kan følge med på hva som skjer til enhver tid.

*Totalt timeantall 27.03.17-21.04.17: 390 (26*5*3)*

Research (89)
AWS Cloudfront streaming, inkorporere med AWS IoT & mobile hub (89)
Utforme grensesnitt for streaming (10)
Motta bitstream fra AWS, fremvise i grensesnitt (89)
Koble opp at knapp triggrer av/på med bitstream (34)
Skrive tester for funksjonene vi implementerer i brukerhistorien (55)
Dokumentere, teknisk (21)

Antall timer estimert til oppgaver: 387

Sprint 4 retrospect
Etter en utfordrende sprint 3 økte produktivitet og fremdrift i sprint 4.
Det ble gjort mye rapportarbeid som tidligere ble utsatt grunnet eksamen og avslutning av IS-305, og samtidig fikk vi implementert brukerhistorie 3 og 4, og fikk som planlagt lagt en videre plan for implementasjon av brukerhistorie 5.

Etter vi i større grad fikk forståelse for AWS og klarte og håndtere autorisering og allokering av ressurser, ble implementasjonen av de påfølgende brukerhistoriene noe mindre komplekse enn vi hadde fryktet. For brukerhistorie 4 gikk vi forøvrig bort fra å kun bruke AWS, og valgte å bruke Google Firebase for push-varslinger til applikasjonen. Grunnen til dette er at AWS sin SNS-modul krever at man i tillegg bruker Firebase eller GCM (Google Cloud Messaging, men denne er nå utdatert og erstattet av Firebase), så vi valgte derfor å ikke bruke AWS SNS i denne omgang. Det AWS kunne automatisert er håndtering av Firebase registrerings-ID'er som vi håndterte manuelt, men på dette stadiet så vi ikke verdien i å i tillegg benytte oss av SNS fremfor kun Firebase.
Gruppen delte i større grad ut ansvarsområder til de enkelte medlemmene som gjorde at vi klarte å ha fokus og være produktiv på flere arbeidsoppgaver til samme tid. I tillegg til stor fremgang både på den tekniske løsningen og det organisatoriske med rapporten, fikk vi i tillegg samarbeidet med en annen gruppe for utveksling av tanker og idéer, samt gjennomført brukertesting av digital prototype og mobilapplikasjonen.
Det var ingen store problemer på denne sprinten, da gruppen føler at det planlagte arbeidet ble gjennomført til rett tid.

Ved slutten av sprinten hadde vi et styringsgruppemøte med veileder, hvor vi foretok et rollespill grunnet at vi ikke har en ekstern arbeidsgiver. Dermed tok Scrum-master på seg rollen som produkteier for å simulere et mer reelt styringsgruppemøte og for å se situasjonen fra en annen vinkling. Møtet var vellykket og vi fikk gode tilbakemeldinger og refleksjoner fra veileder. Noen av tilbakemeldingene gjaldt blant annet testing og tips om fremtidig endring av kodebase på pi'en, veileder anbefalte oss å bruke GoLang om vi ønsket å faktisk videreselge et ferdig produkt i fremtiden. Dette grunnet noen utfordringer vi har hatt i forhold til multithreading og ytelse i Python. Det kom også frem fra veileder at om vi er flinke til å lage og utføre gode unit-tester er det ikke så nødvendig med systemtesting.

Sprint 4 review (Sprint 4: 24.03.17 - 07.04.17)
Sprint 4 har vært en hektisk sprint med veldig mye arbeid å gjøre, og en del overtid for å fullføre de planlagte oppgavene for å komme oss ajour.

Den overordnede planen for sprint 4 var å ferdigstille det siste på brukerhistorie 2, samt å få gjort brukerhistorie 3 og 4. Det var også ønskelig at vi fikk påbegynt brukerhistorie 5.

Resultat:
Brukerhistorie 2: Som bruker ønsker jeg å kunne armere og deaktivere sikkerhetssystemet så jeg kan føle meg trygg.
Estimert i planning: 12
Estimert i scrumdesk: 28
Endelig timeantall: 19,5
Ved starten av denne sprinten var nesten alt løst i forhold til brukerhistorie 2, men det gjenstod enda å konfigurere Raspberry Pi til å håndtere interaksjonen når meldinger sendes over MQTT og hvordan den skal reagere. Det var også et spørsmål hvordan vi skulle håndtere alle de ulike prosessene som skulle kjøre på en gang, og unngå konflikter. Dette ble løst ved å bruke multiprosessering hvor det blir opprettet en prosess for hver av oppgavene som skal løses og være aktive samtidig.

Brukerhistorie 3: Som en bruker ønsker jeg å kunne se siste stillbilde fra hjemmet, slik at jeg kan få en indikasjon på situasjonen hjemme når det passer meg.
Estimert i planning: 129
Estimert i scrumdesk: 97
Endelig timeantall: 41
Her var det brukt mye tid fra før av (brukerhistorie 2) på ressursallokering og innstillinger for autorisasjon, og dermed gikk denne biten raskere enn forventet. Raspberry Pi måtte kobles opp for å kunne sende inn bilde til en S3 bucket, og den måtte ha tillatelse til å gjøre det. Deretter måtte app'en kobles opp til å hente ut det nyeste bildet, og ha tillatelse til å gjøre det. Dette gikk relativt smertefritt, men mangler fremdeles litt finpuss og feilhåndtering som vil bli gjort i oppryddingsfasen.

Brukerhistorie 4: Som bruker ønsker jeg å få push-notifikasjoner når systemet oppdager anomaliteter, for å være oppdatert på hva som skjer når jeg ikke er hjemme.
Estimert i planning: 257
Estimert i scrumdesk: 181
Endelig timeantall: 52
Også her hadde våre tidligere erfaringer stor innvirkning på hvordan vi fikk løst brukerhistorien. Opprinnelig var planen å bruke Amazon sin SNS men vi gikk over til å bruke Google Firebase for å fullføre brukerhistorien. Dette ble utført ganske enkelt siden Android Studio har verktøy for å inkorporere Firebase direkte prosjektet. Ved å sette dette opp mot Firebasen vi satte opp hos Google, fikk vi enkelt sendt push-varslinger til Android-systemet med en Firebase-registrerings ID som appen genererer, fra administreringskonsollen i nettleseren. Etter dette fant vi ut hvordan Raspberry Pi kunne interagere med Firebase for å sende push-varsler når PIR-sensor oppdaget noe bevegelse, og fikk app til å sende over sin Firebase-ID hver gang den startet opp.

Brukerhistorie 5: Som en bruker ønsker jeg å kunne se video live fra hjemmet, slik at jeg kan følge med på hva som skjer til enhver tid.
Estimert i planning: 387
Estimert i scrumdesk: 392
Endelig timeantall: 10
Planen var bare å komme i gang med brukerhistorien, og dette fikk vi til. Vi researchet hvordan best mulig løse dette, og det første vi så var at AWS sin CloudFront streaming tjeneste krever at man oppretter en EC2 instans som dermed sannsynligvis ville blitt kostbart å ta i bruk, spesielt nå under produktutvikling hvor det ikke genereres inntekt. Vi har sett på et alternativ som kan være bedre og kostnadsfritt, som da involverer en privat VPN tunnel mellom app og terminal, siden streamingen sitt formål ikke krever en streamingserver som skal nå mange noder samtidig. Da kan vi bruke tokens eller Firebase registrerings ID til å autorisere start av og avspilling av stream over denne VPN-tunnelen

SPLIT Report writing IS-304
Estimert i planning: 0
Estimert i scrumdesk: 85
Endelig timeantall: 127
Her gikk det en del tid på rapportskrivning og veiledning fra faglærer. Dette var oppgaver som skulle vært gjort tidligere, men ble nedprioritert grunnet eksamen i IS-305. Denne sprinten fikk vi jobbet ekstra og dermed fått mye bra stoff til rapporten.
SPLIT Business part
Estimert i planning: 0
Estimert i scrumdesk: 36
Endelig timeantall: 27

Under disse oppgavene inngikk blant annet kontraktsforhandling med potensiell kunde Studiehjelpen, styremøte for studentbedriften og nettsideutvikling for studentbedriften. Disse oppgavene ble løst ganske rett frem, og status med potensiell kunde Studiehjelpen er enda uavklart.

Report writing IS-304

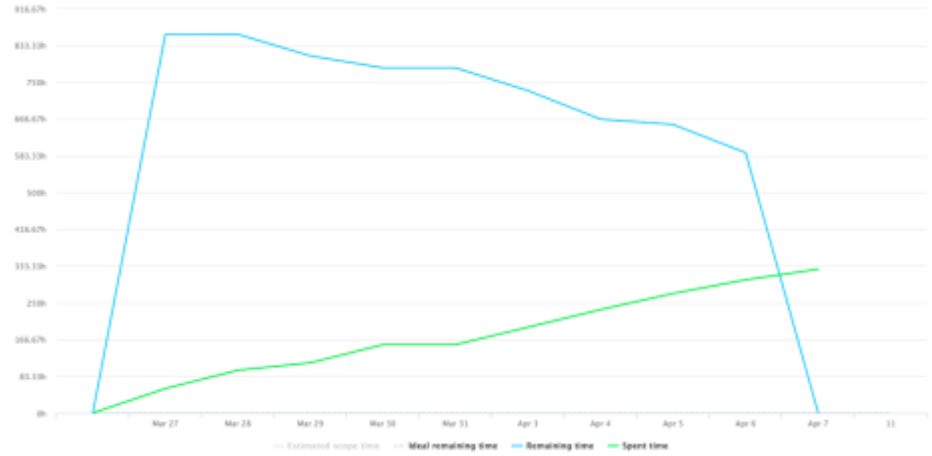
Estimert i planning: 0
Estimert i scrumdesk: 54
Endelig timeantall: 49,5

Her inngikk scrum-master arbeid, styringsgruppemøte, sprint planning og review/retrospekt

Business part

Estimert i planning: 0
Estimert i scrumdesk: 10
Endelig timeantall: 10

Burndown chart



Sprint 5

Planning meeting

Til sprint 5 var det noe utfordrende å planlegge. Dette var rett etter påsken, og grunnet logistiske utfordringer var vi ikke fulltallig gruppe før torsdagen. Før den tid fikk vi gjort ferdig review av forrige sprint, men retrospect og planning ble forsinket til torsdagen og vi henger dermed etter allerede fra starten.

I tillegg gjorde sykdom at to medlemmer uteble fra mandagen den andre uken i påsken, og et tredje medlem ble delvis borte grunnet sykdom og legetime. Dermed er det kun 2-3 medlemmer igjen til å prøve å holde fortet den siste uken av denne sprinten.

Brukerhistorie 5 V1: Som en bruker ønsker jeg å kunne se video live fra hjemmet, slik at jeg kan følge med på hva som skjer til enhver tid.
*Totalt timeantall 19.04.17-28.04.17: 204 (10*4*5,1)*

Research (89)
AWS Cloudfront streaming, inkorpoere med AWS IoT & mobile hub (89)
Utforme grensesnitt for streaming (10)
Motta bitstream fra AWS, fremvise i grensesnitt (89)
Koble opp at knapp triggrer av/på med bitstream (34)
Skrive tester for funksjonene vi implementerer i brukerhistorien (55)
Dokumentere, teknisk (21)

Antall timer estimert til oppgaver: 387 **(utdatert)**

Brukerhistorie 5 V2: Som en bruker ønsker jeg å kunne se video live fra hjemmet, slik at jeg kan følge med på hva som skjer til enhver tid.

Grunnet endring i planlagt gjennomføring måtte det gjøres ny planlegging for implementasjon av brukerhistorie 5.

Research (89)
Sette opp VPN med Raspberry Pi host (89)
Utforme grensesnitt for streaming (10)
Motta bitstream over VPN, fremvise i grensesnitt (89)
Koble opp at knapp triggrer av/på med bitstream (34)
Kontrollere VPN / stream tilgang med Firebase-ID eller annen unik identifikator(55)
Dokumentere, teknisk (21)

Antall timer estimert til oppgaver: 387

Brukerhistorie 6. Som bruker ønsker jeg å få push-notifikasjoner når systemet aktiveres og deaktiveres, for å vite om noen tukler med systemet.

Skrive ny logikk for flere push-varsler(10)

Grunnet at vi allerede har implementert push-varslinger for systemet, anser vi denne brukerhistorien som en mindre kompleks oppgave som kan utføres på en arbeidsdag. Eksisterende funksjonalitet kan brukes til å skrive ny applikasjonslogikk som håndterer en ny hendelse.
Antall timer estimert til oppgaver: 10

Brukerhistorie 8. For å skreddersy systemet til mitt behov til enhver tid, ønsker jeg som en bruker å kunne endre systemets innstillinger fra mobil-applikasjonen

Research(21)
GUI for innstillinger(10)
Logikk for å lagre innstillinger(89)
Testing(21)
Dokumentering(21)

Timer estimert til oppgaver: 162
Selv om brukerhistorie 8 er et stykke ned på listen, har vi valgt å endre prioritering for å implementere denne før brukerhistorie 7. Dette grunnet at vi føler at innstillinger er en viktigere del av en fungerende prototype og MVP. Vi er noe usikre på hvordan dette løses på en god måte, og setter derfor mye tid til research og logikk, og mindre fokus på grensesnitt og testing i denne iterasjonen.

Rapportskriving.
I tillegg til å holde hovedfokuset på implementasjon ønsker vi parallelt å jobbe med rapporten for Bacheloroppgaven.

Timer estimert til oppgaven: 12

Sprint 5 retrospect (Sprint 5: 19.04.17 - 28.04.17)

Vi fikk implementert brukerhistorie 5v2 (Video-streaming) i sprint 5.
Det ble dog ikke fokusert særlig mye på de resterende brukerhistoriene og oppgavene vi hadde med i sprint-planningen. Det viste seg å være mer utfordrende enn først antatt å implementere live-video streaming. Dette skyldtes blant annet at vi fikk opp en stream tidlig i sprinten, men at følgende rammeverk UV4L ikke tilbyr noe form for hosting på serversiden, derfor antok vi at det kanskje ville ta mer tid å sette opp egen server enn å prøve en annen løsning. Vi prøvde derfor ut UStream, IBM's plattform for videostreaming da vi hadde kjennskap til denne plattformen fra før av, og det fantes en del god dokumentasjon på dette. Noe av utfrodringen her var å få installert FFmpeg, det viste seg at det var kompatibilitetsproblemer med den versjonen av operativsystemet Raspbian Jessie (8) som vi bruker på Raspberry Pi'en. Måten vi løste problemet på var at vi clonet rett fra repositoriet til FFmpeg, som forøvrig er Open Source. Slik som dette: `git clone https://github.com/FFmpeg/FFmpeg.git`

Det ble deretter installert diverse biblioteker og H264 støtte. Vi fikk dermed opp en stream på UStream, og med en liten justering også på Youtube. Dette fungerte fint og vi fikk høyoppløst videostream på internett. Ulempen var at streamen ikke ble privat på UStream, da måtte man betale for private-sessions. Hovedproblemet var at vi ikke fikk det inn i appen med konvensjonelle WebView, VideoView eller MediaPlayer bibliotekene. Vi kunne sannsynligvis løst dette med å hente Youtube/Google API inn i appen for å få videospilleren integrert. Dette ville da kunne gått ut over kvaliteten på streamen, i og med at kallet til YouTube API vil ta ressurser. Men vi har da fått til å konfigurere stream med disse to tjenestene, hvis vi finner ut at vi ønsker å gå for dette i en mer ferdig utgave av systemet. Vi gikk til slutt tilbake til det første alternativet UV4L, hvor vi fant en tjeneste (Weaved) som tilbød gratis VPN-port forwarding fra localhost på Pi til client på Android/app siden. Denne

løsningen er ikke ideell, da det er en betalingsløsning og vi får generert en link som utløper etter 30 minutter. Så vi burde vurdere å implementere en egen SSH-tunnell, eller hente inn Youtube API i en senere sprint.

Denne sprinten har skilt seg ut fra de andre spesielt med tanke på at de på gruppen som har jobbet mindre på den tekniske delen av prosjektet nå har fått en sprint med rent teknisk arbeid. Dermed har alle gruppemedlemmene fått økt innsikt på den tekniske biten av prosjektet, samtidig som vi har sørget for at alle prøver seg frem i ulike roller. Det har vært viktig for oss å ha mer dynamiske roller i prosjektet, slik at medlemmer ikke ville vært fastlåst til en viss del av prosjektet.

Videre har gruppens beslutninger om å gå videre på alternative løsninger når vi har stått fast, resultert i at vi nå har 3 forskjellige løsninger for brukerhistorie 5. Dette har også medført at vi nå har en teknisk løsning for brukerhistorie 10 streaming av forelesninger, selv om vi endte opp med WebView og UV4L som var den første tenkte løsningen. Vi har dermed fått prøvd frem ulike tjenester, sett på videokvalitet og "delay" på streamingen før vi tok et endelig valg.

Når vi ser tilbake på sprint 5, har det vært en sprint vi kan si oss fornøyd med, selv om vi kanskje skulle ønske at vi hadde kommet lengre grunnet alt arbeid som gjenstår. Hva vi kunne gjort annerledes med denne sprinten er vanskelig å se i retrospect, dette skyldes blant annet at vi har hatt et ønske om å heve den tekniske forståelsen og gi mer dynamiske roller i gruppen, samt at vi har hatt fravær fra et gruppemedlem store deler av sprinten.

Sprint 5 review (Sprint 5: 19.04.17 - 28.04.17)

Det gjenstår fortsatt mye arbeid, samtidig som dette har vært den korteste sprinten i prosjektet så langt. Målet for denne sprinten var i hovedsak å fullføre funksjonalitet på brukerhistorie 5v2. Selv om vi hadde øvrige mål, var dette minstekravet for at vi kunne anse dette som en suksessfull sprint. Dette klarte vi.

Resultat:

Brukerhistorie 5 V2: Som en bruker ønsker jeg å kunne se video live fra hjemmet, slik at jeg kan følge med på hva som skjer til enhver tid.

Estimert i planning: 387
Estimert i scrumdesk: 168
Endelig timeantall: 115.5
Denne oppgaven var opprinnelig estimert til å ta en del flere timer enn det vi har brukt til nå. Noe av bakgrunnen for dette var at vi forventet å inkorporere det med AWS ved å ta i bruk CloudFront for streaming. Da det ikke ble tilfellet grunnet kostnader, ble det mer aktuelt å finne en annen løsning hvor Pi'en stod for streamingen og det ble hostet gjennom enten eget domene eller en leverandør som kunne holde oppe streamen så vi fikk fremvist den i appen. Til nå er vi under timetallet, men kommer til å måtte fikse opp i en del under oppryddingen og forventer

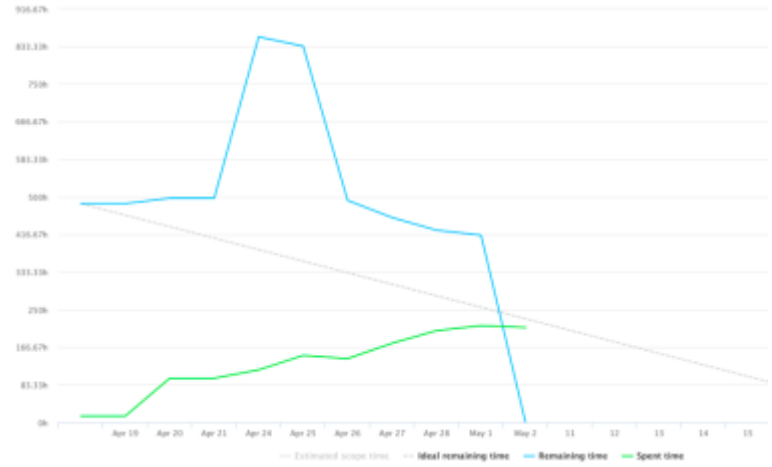
dermed å komme nærmere det faktiske estimatet som da også inneholder testing, dokumentasjon og GUI.

Brukerhistorie 6. Som bruker ønsker jeg å få push-notifikasjoner når systemet aktiveres og deaktiveres, for å vite om noen tukler med systemet.
Estimert i planning: 10
Estimert i scrumdesk: 0
Endelig timeantall: 0
Rakk ikke påbegynne denne i denne sprinten.

Brukerhistorie 8. For å skreddersy systemet til mitt behov til enhver tid, ønsker jeg som en bruker å kunne endre systemets innstillinger fra mobil-applikasjonen
Estimert i planning: 162
Estimert i scrumdesk: 0
Endelig timeantall: 0
Rakk ikke påbegynne denne i denne sprinten.

Rapportskriving.
Estimert i planning: 12
Estimert i scrumdesk: 90
Endelig timeantall: 92
Her bommet vi på estimat da vi ikke tok høyde for planning, review og retrospekt. Dette tok litt ekstra tid da vi hadde en del nye oppgaver å estimere for å få inn i backloggen. Da det ble gjort ga det oss muligheten til å bare trekke ut av backloggen til sprintene, og senker dermed tidsbruken vår ved senere planlegginger.

Burndown chart



Sprint 6

Planning meeting

Ettersom tidsfristen for rapporten i IS-304 nærmer seg blir dette en sprint som innledende vil fokusere på å delegere ut og jobbe med forskjellige emner innad i rapporten. Etter teknisk dokumentasjon og rapport er levert vil fokuset gå over på nye iterasjoner av tidligere brukerhistorier, samt påbegynt implementasjon av brukerhistorie 6 og 8(hvis tid). Brukerhistorie 5 finpusses frem mot styringsgruppemøte 8. mai.

Rapportskriving

Innledning
Teori
Teknologi-stack
Hoveddel
Implementasjon
Utfordringer
Scrum
Business-del
Refleksjon
Konklusjon
Vedlegg
Kilder (Få inn mye Google Scholar?)

Iterasjon 2 av brukerhistorie 1-5

Grensesnitt og grafiske forbedringer
Feilhåndtering
Øvrig finpuss
Brukertesting
Unit-testing

Brukerhistorie 1

Feilhåndtering (få opp feilmelding med en gang ved feil type tlf nummer, epost etc) (10)
Gui rework (5)
Firma logo (5)

Brukerhistorie 2

Dokumentere IoT oppkoblingen på teknisk plan (protokoll, hvordan informasjon hentes og leveres og hva slags type requests/svar) (6)
Skrive tester for funksjonene vi implementerer i brukerhistorien (17)

Brukerhistorie 3

Skrive tester for funksjonene vi implementerer i brukerhistorien (8)
Dokumentere filbehandlingen på teknisk plan (3)

GUI for arkiv (8)
Funksjonalitet til arkiv (få inn bildene i liste i GUI fra database i skyen) (22)
GUI rework (5)

Brukerhistorie 4
Skrive tester for funksjonene vi implementerer i brukerhistorien (21)
Dokumentere push-notifikasjonene på teknisk plan(8)

Brukerhistorie 5
Vurdere om vi skal bruke nåværende løsning eller gå over til SSH over Pi (2)
Pi må refactores så stream kan kjøres opp ved mottak av MQTT message (altså, on_msg_receive) (10)

Ang. testing: Vurdere å kjøre Firebase robotest / test analytics, får kanskje noe god informasjon ut av det?
Ang. testing: Vurdere å få Travis fungerende.

Brukerhistorie 6. Som bruker ønsker jeg å få push-notifikasjoner når systemet aktiveres og deaktiveres, for å vite om noen tukler med systemet.
Skrive ny logikk for flere push-varsler(10)

Grunnet at vi allerede har implementert push-varslinger for systemet, anser vi denne brukerhistorien som en mindre kompleks oppgave som kan utføres på en arbeidsdag. Eksisterende funksjonalitet kan brukes til å skrive ny applikasjonslogikk som håndterer en ny hendelse.

Antall timer estimert til oppgaver: 10

Brukerhistorie 8. For å skreddersy systemet til mitt behov til enhver tid, ønsker jeg som en bruker å kunne endre systemets innstillinger fra mobil-applikasjonen
Research(21)
GUI for innstillinger(10)
Logikk for å lagre innstillinger(89)
Testing(21)
Dokumentering(21)

Timer estimert til oppgaver: 162
Selv om brukerhistorie 8 er et stykke ned på listen, har vi valgt å endre prioritering for å implementere denne før brukerhistorie 7. Dette grunnet at vi føler at innstillinger er en viktigere del av en fungerende prototype og MVP. Vi er noe usikre på hvordan dette løses på en god måte, og setter derfor mye tid til research og logikk, og mindre fokus på grensesnitt og testing i denne iterasjonen.

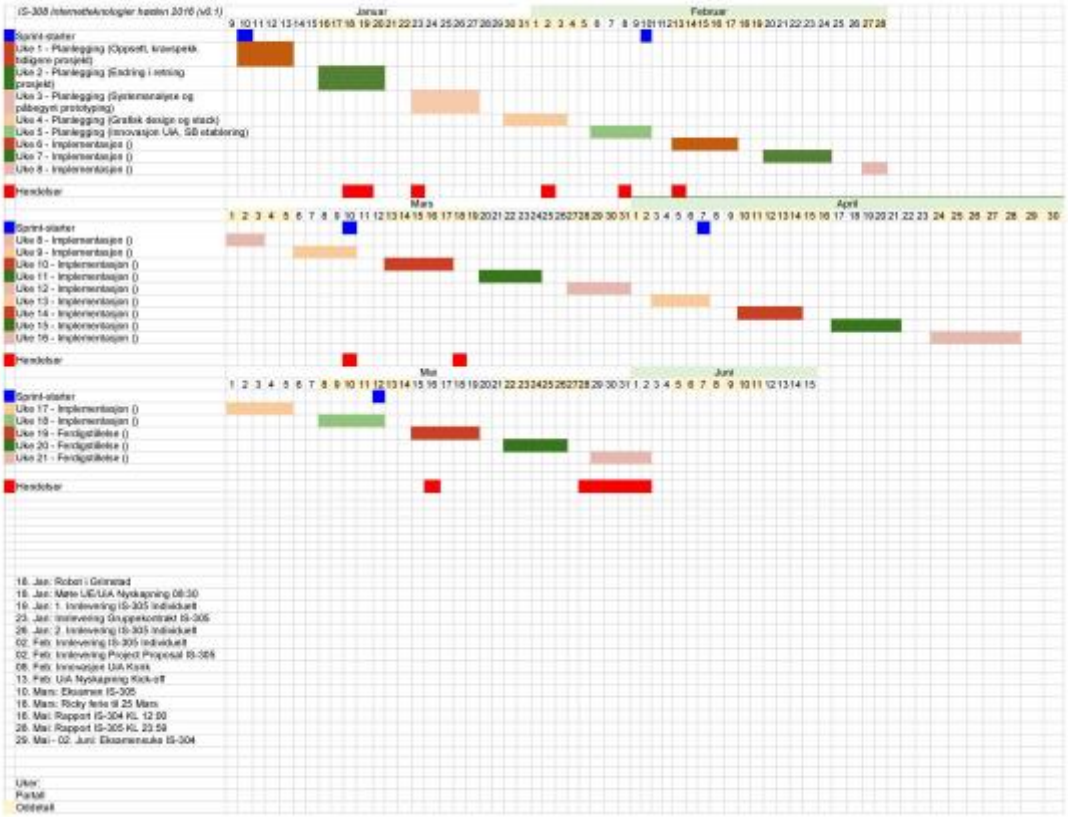
Sannsynligvis vil denne brukerhistorien måtte presses over til siste sprint (post-sprint) hvor vi rydder opp i prosjektet og klargjør til eksaminering. Dette grunnet tidspress og mangel på oversikt over eksakt hvor mange timer som må til for å fullføre oppryddingsprosessen vi påbegynner nå. Dette er også årsaken til en noe manglende estimeringsprosess i denne sprinten, da vi vet vi har nok oppgaver denne sprinten uansett og ikke tjener på å bruke for mye tid på å estimere oppbrutte og små oppgaver, kontra å faktisk arbeide.

Sprint 6 retrospect/review/burndown

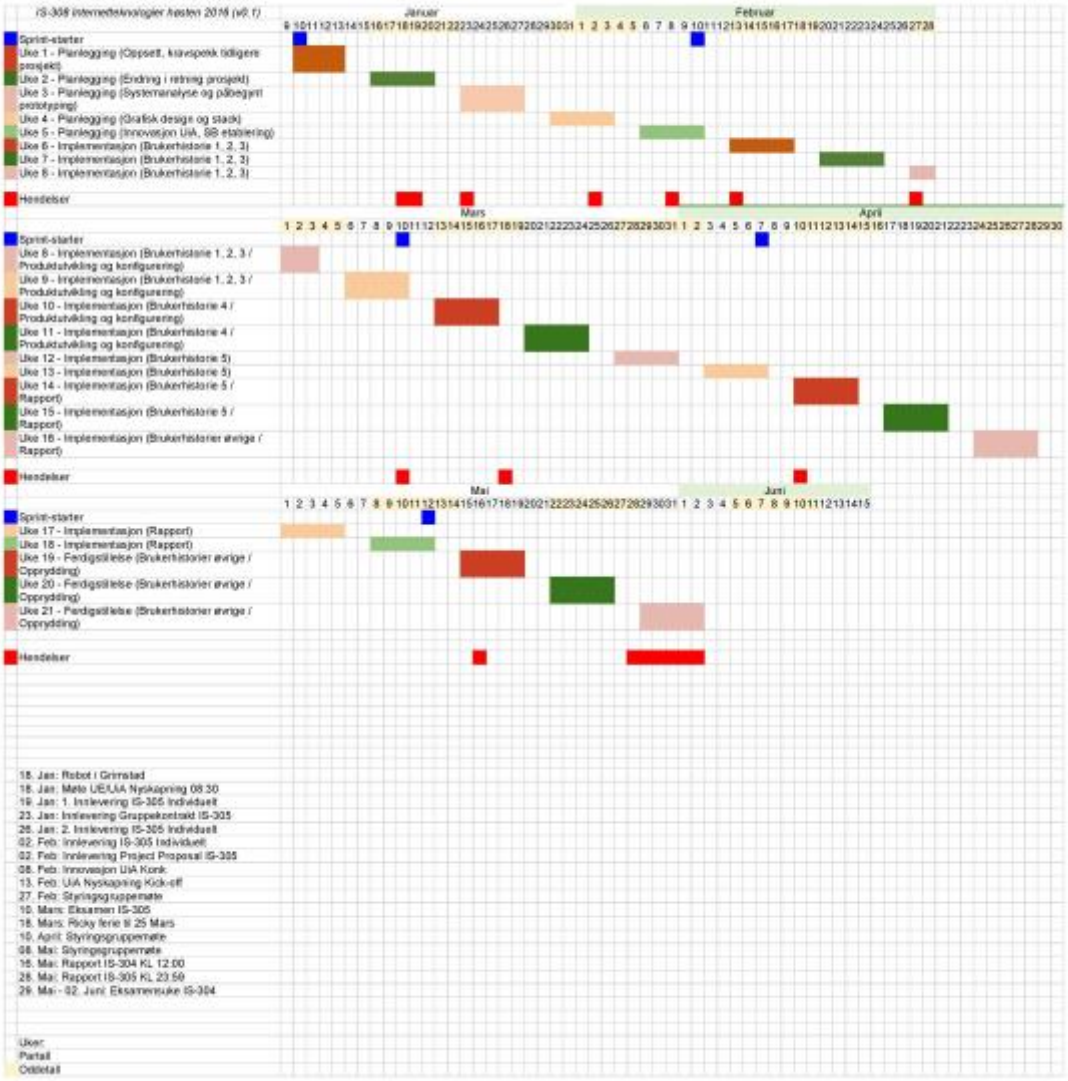
Dette vil bli lagt frem på eksamen ettersom sprint 6 avsluttes dagen etter innleveringsfristen for rapporten.

Vedlegg 5 - Prosjektplaner

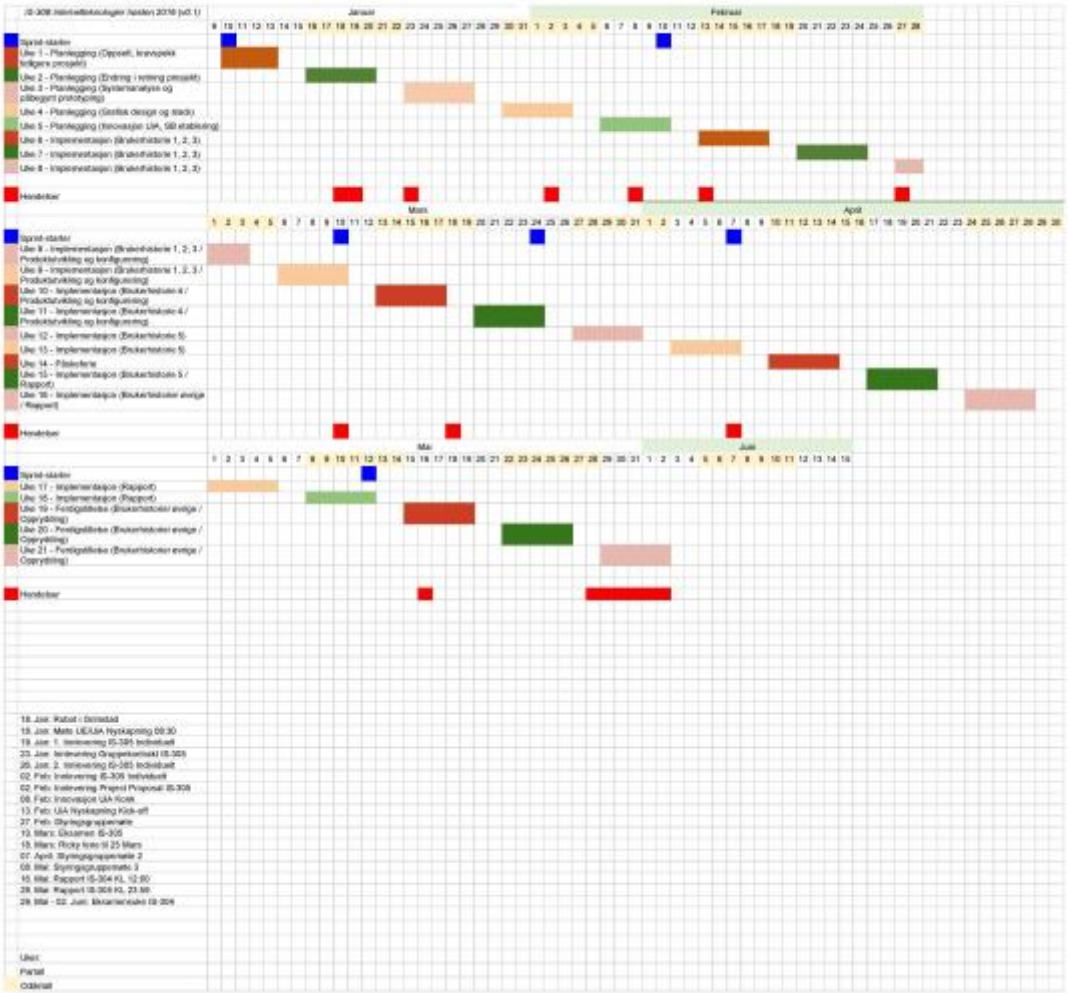
Vedlegg 5.1 - Gantt chart versjon 1



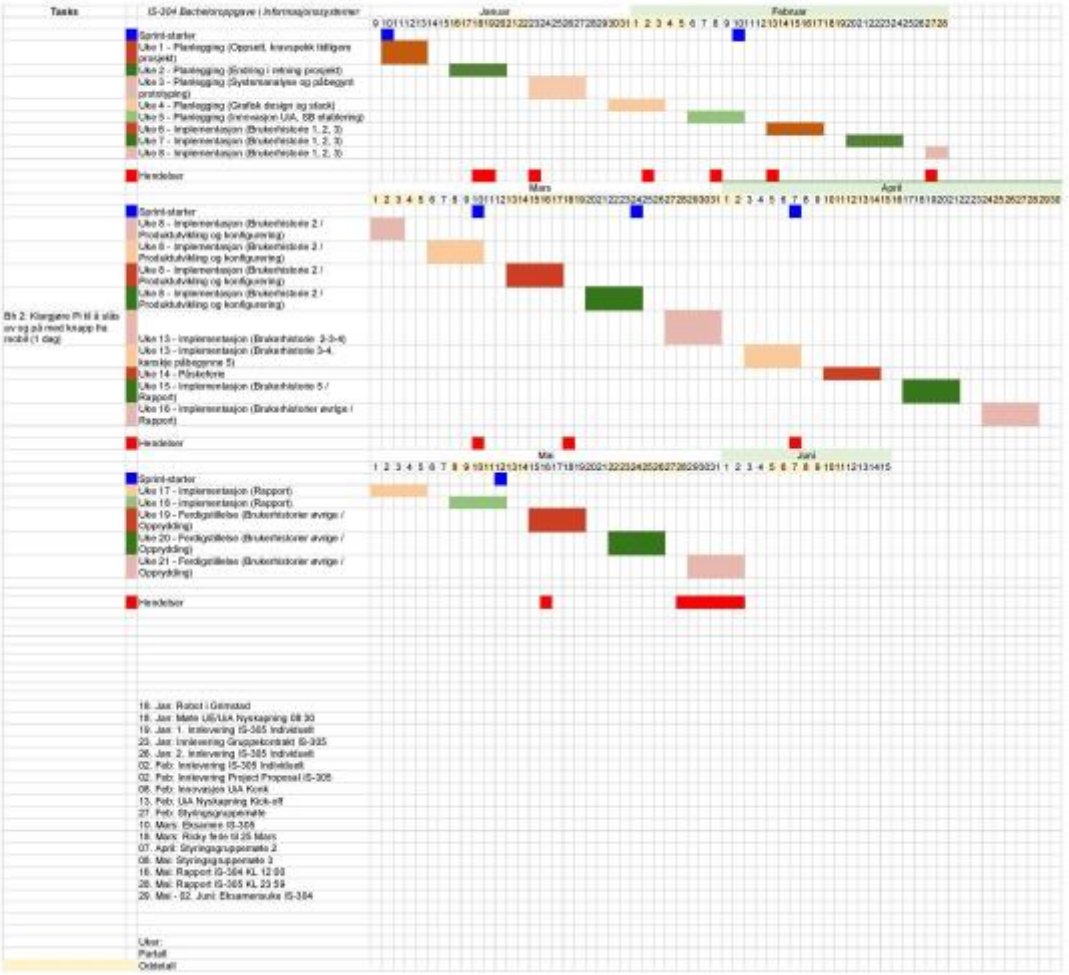
Vedlegg 5.2 - Gantt chart versjon 2



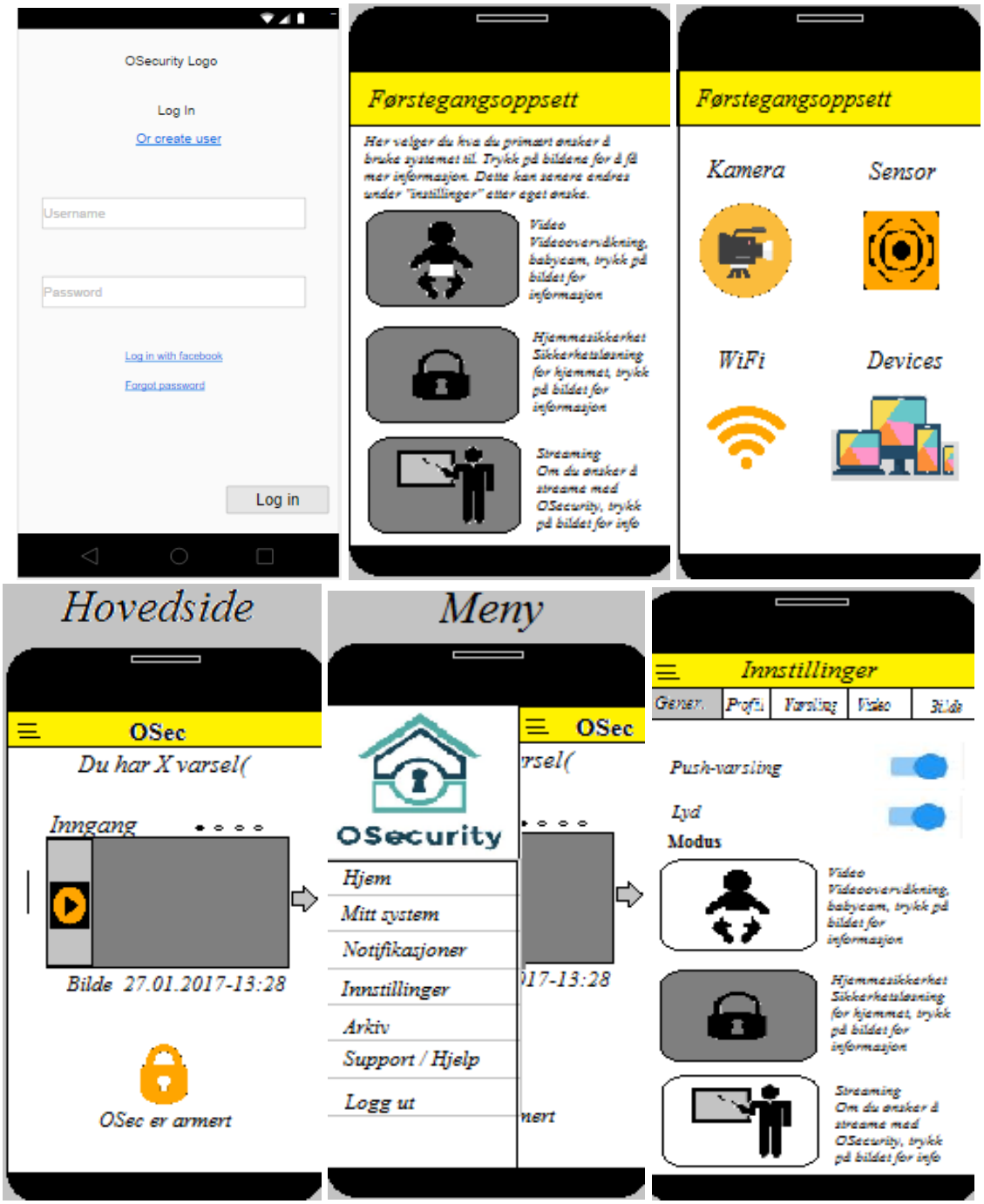
Vedlegg 5.3 - Gantt chart versjon 3



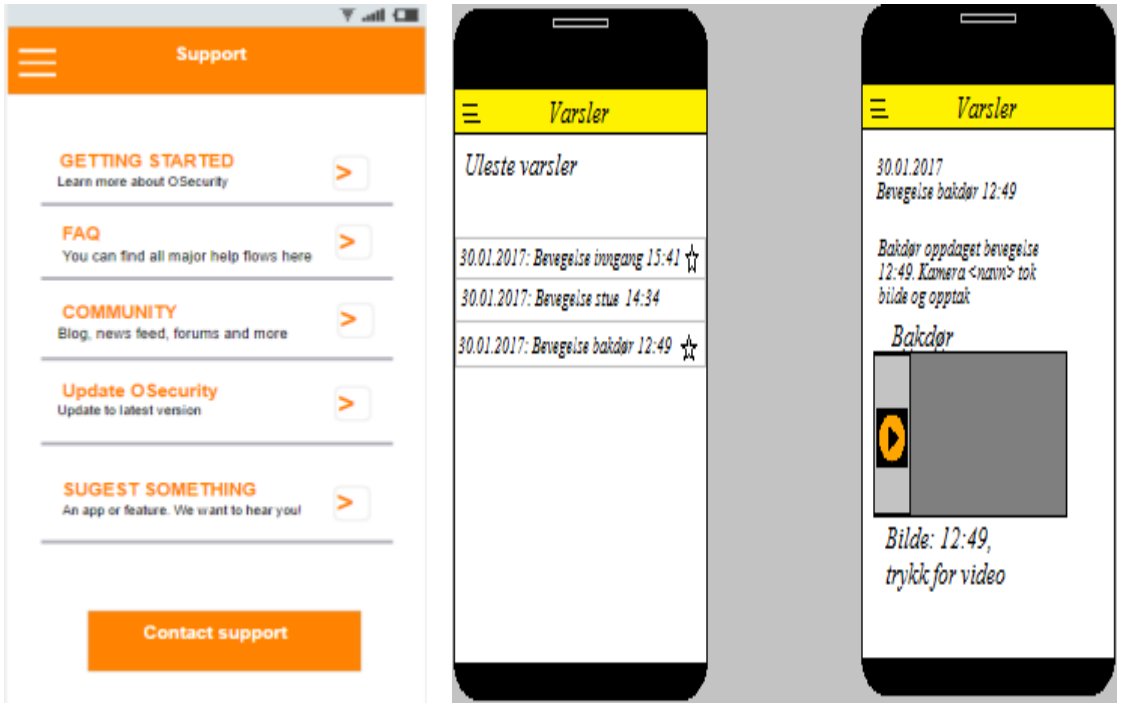
Vedlegg 5.4 - Gantt chart versjon 4

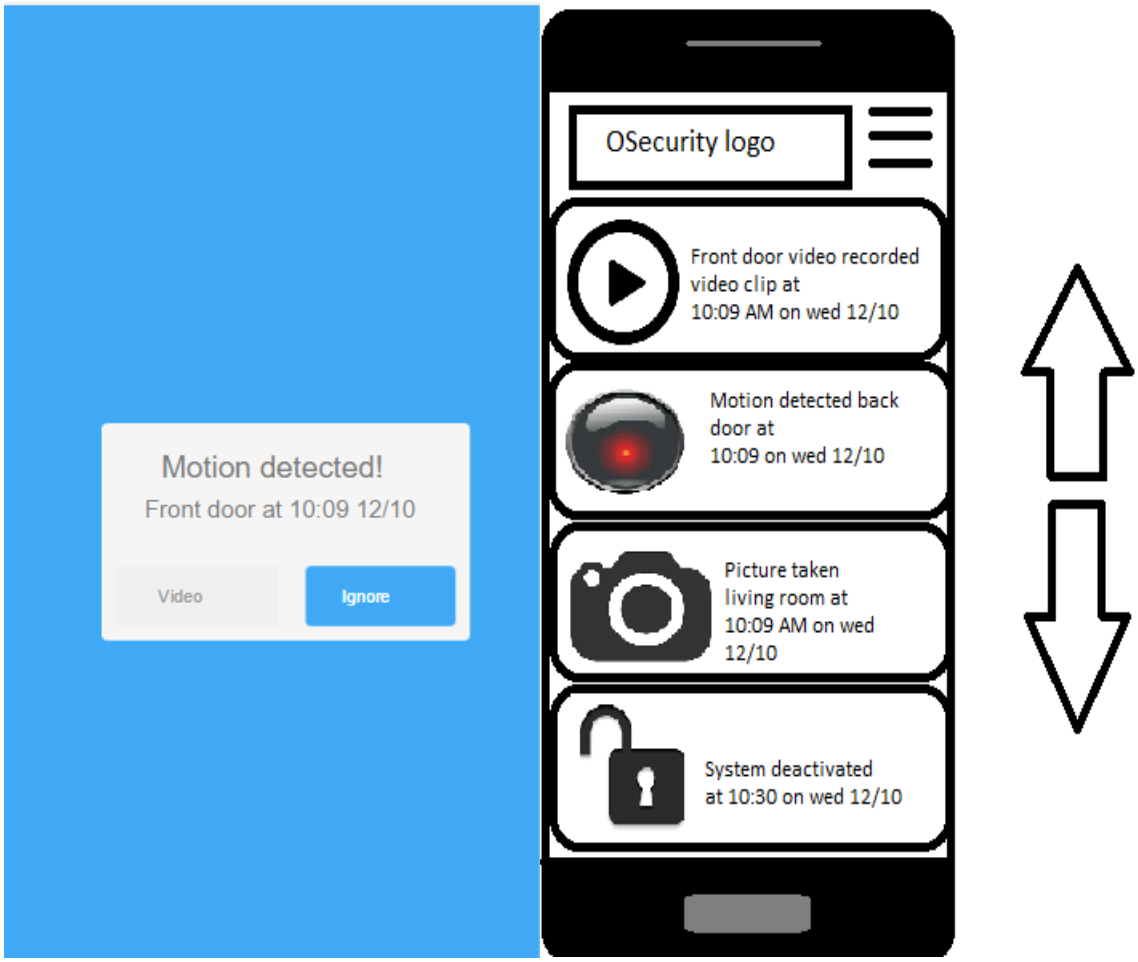


Vedlegg 6 – Grafisk design




Innstillinger
Profil
Varsling
Video
Bilde
Stillbilder
Hyppighet for oppdatering av stillbilde
Lav
30 sek
Høy
Bildekvalitet
Lav
Høy
Lagring
SD-kort terminal
Innstillinger
Push-varsling
E-post varsling
Vibrering
Lyd
Send bilde ved varsel
For videre bilde- og videoinstillinger gå til "bilde" eller "video" kategoriene
OSECURITY
OPEN-SOURCE SECURITY
Video
Bilde
Profil
Varsler
Robin Moudnib
E-post:
robin_rondestvedt@live.no
Passord
Lagre
Passord:
Gammelt passord
Nytt passord
Bekreft nytt passord
Lagre
Wi-Fi status:
Tilkoblet "eduroam"
Velg nettverk
Hjemnett
Koble til


Support
GETTING STARTED
Learn more about OSecurity
FAQ
You can find all major help flows here
COMMUNITY
Blog, news feed, forums and more
Update OSecurity
Update to latest version
SUGEST SOMETHING
An app or feature. We want to hear you!
Contact support
Varsler
Uleste varsler
30.01.2017: Bevegelse inngang 15:41
30.01.2017: Bevegelse stue 14:34
30.01.2017: Bevegelse bakdør 12:49
Varsler
30.01.2017
Bevegelse bakdør 12:49
Bakdør oppdaget bevegelse 12:49. Kamera <navn> tok bilde og opptak
Bakdør
Bilde: 12:49,
trykk for video


Motion detected!
Front door at 10:09 12/10
Video
Ignore
OSecurity logo
Front door video recorded video clip at 10:09 AM on wed 12/10
Motion detected back door at 10:09 on wed 12/10
Picture taken living room at 10:09 AM on wed 12/10
System deactivated at 10:30 on wed 12/10

Vedlegg 7 – Arbeidsflyt utvikling OSecurity

Arbeidsflyt utvikling OSecurity

Utviklingen vil gjøres hovedsakling i Android Studio for mobilapplikasjonen mens det vil være scripts laget i Python for å styre sikkerhetssystemet.
Det må bygges et program på Raspberry som kjører de ulike scriptene for sensorer, kamera osv.

Innlogging er den første biten (første brukerhistorie)

- Lage samme scripts i Python som snakker med sensorer og sender informasjon videre til server og database.
- Sette opp TomCat (e.l.) server node og enkel DB med grunnleggende entiteter
- Lage en enkel innlogging / fremvisning i Android Studio for testing av kommunikasjon med server / DB og fjernstyring av terminalapplikasjon

Github

- Arbeid gjøres i branches før det går inn til master
- Vi jobber i git flow med master, development og potensielt feature branches
- All kode dokumenteres og commites med engelsk kommentar

Utfordringer og research

Ved utfordringer oppfordres det enkelte gruppemedlem til å henvende seg til andre for input og hjelp. Det oppfordres også til å bruke tid dedikert til research for å trene seg opp og lære mer om programmeringsspråkene det utvikles i.

Loggføring av arbeid

Loggføringen av utført arbeid skjer hovedsaklig på Scrumdesk i likhet med resten av arbeidet vi gjør i IS-304.

Vedlegg 8 - Gruppekontrakt

Group Contract for Salt Mine Development

1. Group members

Name	**Email**	**Phone**
Sondre Flovik	sondrf14@uia.no	48227298
Robin A. R. Moudnib	robin_rondestvedt@live.no	45198084
Ricky L. Omland	ricky-omland@hotmail.com	41607209
Christian F. Thorne	cfthorne@hotmail.com	99448742
Erik O. Zetterquist	erik.zetterquist@gmail.com	99433705

2. Purpose

We wish to perform the required tasks in the course in a way that enables us to learn as much as possible and achieve satisfying academical results. We also wish to link it as much as possible to our project in IS-304, so that the compulsory work feels more productive towards a final goal.

3. Expectation and goal

- As defined in our purpose, the main goal is to learn and achieve.
- We expect every member to work hard.
- We expect that everyone has the opportunity to meet and that we meet at the agreed time unless a notice is given to the rest of the group.
- We expect people to participate in the whole process and take initiative to add their own value to the project.
- We expect that each member are responsible for their own learning and prepares accordingly for the tasks needed.
- We expect that people are honest and transparent with the group, and address issues when they arise.

4. Group administration

The name of the group is “Salt Mine Development”. Our “Scrum-Master” is Robin, who will keep track of tasks and delegations. Other administrative tasks that normally falls to a leader, is generally made in the group in a democratic manner. We believe this is the best way to function as a group, even though we are aware of the extra time it takes to agree on decisions.

5. Meet information
We try to operate with daily schedules which goes from 10 am to 4 pm every weekday, which includes lectures in courses. Individual work and assignments are external from this schedule and are done at home as homework.

Of these hours, we make sure to differentiate between where the hours are being put in, whether it's IS-304 or IS-305 and sometimes the work affects both courses.
We must follow the course structure and incorporate relevant course terminology and relevant literature.

We book rooms when we are together so that we are all aware of the location of our meetings.

6. General information
To share information, we use our own Slack channel, Facebook group and chat, Google Drive and Scrumdesk. With these tools every member has the opportunity to ask each other questions, and all the information is posted here. We use Google Drive to share the documents.

7. Group rules
- Meet at the agreed time or let the group know
- Do the work expected of the individual
- Everyone has the right to their own opinion, but it will be the majority of the group who will decide if there is a disagreement.
- Give constructive criticism and positive feedback, when needed.
- If we are behind schedule, we work overtime to a certain extent. We may still have to reorganize the work if we discover that we have miscalculated our estimates.
- If other people's work is used, it must be clearly stated by sources, and the rest of the group must be informed.
- All information will be updated continuously on our Slack channel, so that every member of the group has access to the newest information at all times.
- If a group member does not have the opportunity to attend a group meeting due to illness or other reason, this person must be available on telephone or other social medias, or contribute by submitting individual work.
- Every group member must deliver tasks within the time limits set by the group, or let the group know about issues or reasons you may not be able to finish in time. This must be done early enough for the group to be able to solve the problem.

If all the rules are maintained, the group will function well and we will be able to work better.

If the rules are not maintained, we must bring it up with the group, and if the situation does not improve we have to take it up with the teacher.

If the situation still does not improve it can/will result in exclusion from the group, which again can lead to loss of examination rights. But we will try to solve the situation before it reaches this point.

I approve that I have read and agree with this group contract.

Sondre Flovik

Robin A.R. Moudnib

Ricky Løtoft Omland

Christian Fredrik Thorne

Erik Oskar Zetterquist

Vedlegg 9 - Referat møte med veileder Janis Gailis 02.02.17

Fremviste gant chart
Fikk tilbakemelding om at vi bør legge inn brukerhistoriene i chartet. Lurt å prioritere og estimere på timene. I chartet har vi kun laget implementasjon til nå.

Sikkerhetsaspekt er viktig
Som markedsføring kan vi si at vi spesialiserer oss på internet of things generelt. Det finnes lite kompetanse tilgjengelig på det området, da det er et felt som kommer til å utvikle seg enormt og vise et behov. Når alle skal koble devicene sine, og mange ikke har sikret.

Tips: Hvis vi må selge oss for entreprenørskap kan vi selge det inn at vi satser på å levere sikre tjenester som gjør det sikrere for brukere å bruke Internet of Things.

Overordnede mål
Informerer om at vi vil bruke AWS som serverløsning, videreutvikle selve produktet OSecurity samt utvikle appen for systemet. Informerer om at vi ønsker å bruke trådløse sensorer og kameraer for den utfordringen.

Studentbedrift
Informerer om at vi ønsker å opprette studentbedrift og ha ham som mentor. Dette godtar han.

Bruk i rapporten
Skrive om at vi må avsette ressurser til dette, og at det er viktig, fordi det tar tid å planlegge.

Styringsmøter
Må bare ha 3 møter i løpet av semesteret hvor vi ser på hvordan vi ligger an i forhold til hvordan vi skulle ligge an. Da kan vi se noe på milepælene og hvordan vi ligger i forbindelse med de opprinnelige milepælene.

Informerer om at vi vurderer å ta med oss Vemund fra UE. Dette er i orden.

Slutten av Februar
Begynnelsen av April
Begynnelsen av Mai

Deliverables
Legg inn mer om hva vi leverer basert på brukerhistoriene. Vi burde reflektere rundt brukerhistoriene i rapporten etter at vi er ferdig med dannelsen og prioriteringen av dem.
Vi bør estimere timer, gjøre arbeid, dokumentere og fremvise resultat.

Vedlegg 10 – Systemanalyse-elementer og prioriteringer

Systemanalyse-elementer

For å avgjøre hvilke potensielle systemanalyse- og designelementer vi ønsket å ta i bruk for prosjektgjennomføringen vår, valgte vi å liste opp de seks metodene vi hadde best kjennskap til og deretter prioritere blant dem. Det vi endte opp med var en enighet om at brukerhistorier og klassediagram var de to viktigste elementene. De to vi anså som minst viktige var rikt bilde og sekvensdiagram. Vi avgjorde dermed i fellesskap at vi velger å gå for brukerhistorier, klassediagram, funksjonsliste og sekvensdiagram da vi oppfattet dette som mest hensiktsmessig for vårt prosjekt.

Brukerhistorier RR: 1 S: 2 CF: 2 E: 1 R: 1	**Klassediagram** E: 1 S: 1 R: 1 RR: 2 CF:1
Funksjonsliste S: 3 E:3 RR: 3 R:4 CF:3	**Sekvensdiagram** E: 4 R:4 CF: 3 S: 4 RR: 4
Rikt bilde RR: 6 CF: 6 S: 5 R: 5 E: 6	**Tilstandsdiagram** S: 6 RR: 5 R: 6 CF:5 E.6

S: Sondre Flovik
RR: Robin Amir Rondestvedt Moudnib
R: Ricky Løtoft Omland
CF: Christian Fredrik Thorne
E: Erik Oskar Zetterquist

Vedlegg 11 – Timelogg fra Scrumdesk

Timesheet - OSecurity

From Mar 27, 2017 ***To*** Apr 7, 2017 ***Group by*** NONE ***User*** ALL

unitylol .

When	Backlog Item	Task	Spent
Mar 27, 2017	SPLIT Business part	Board meeting	3
Mar 28, 2017	Report writing IS-304	Sprint 3 review & retrospect	4
Mar 28, 2017	Report writing IS-304	Sprint 4 planning	2
Mar 28, 2017	SPLIT Brukerhistorie 2: Som bruk...	Fikse kodebase på Raspberry Pi, ...	6
Mar 29, 2017	Brukerhistorie 4: Som bruker øn...	Research på AWS SNS for push v...	1
Mar 29, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	4
Mar 30, 2017	SPLIT Report writing IS-304	Brukertesting papir-prototype	9
Apr 3, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	6
Apr 4, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	9
Apr 5, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	7
Apr 6, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	6
Apr 7, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	1
Apr 7, 2017	Report writing IS-304	Steering commitee meeting	4
		Total	**62**

Some Guy

When	Backlog Item	Task	Spent
Mar 27, 2017	SPLIT Business part	Board meeting	3
Mar 27, 2017	Report writing IS-304	Sprint 3 review & retrospec	2
Mar 27, 2017	Report writing IS-304	Sprint 4 planning	2
Mar 28, 2017	SPLIT Brukerhistorie 3: Som en b...	Appen interacter med mobile hu...	6.5
Mar 30, 2017	SPLIT Brukerhistorie 3: Som en b...	Å interacte med "thingen" → Hen...	4
Mar 30, 2017	SPLIT Brukerhistorie 3: Som en b...	Appen interacter med mobile hu...	5
Apr 2, 2017	SPLIT Brukerhistorie 3: Som en b...	Å interacte med "thingen" → Hen...	2
Apr 2, 2017	Brukerhistorie 4: Som bruker øn...	Aktivering/implementering av et ...	0.5
Apr 2, 2017	Business part	Meetings with potential clients	5

When	Backlog Item	Task	Spent
Apr 3, 2017	SPLIT Brukerhistorie 3: Som en b...	Appen interacter med mobile hu...	7
Apr 4, 2017	Brukerhistorie 4: Som bruker øn...	Aktivering/implementering av et ...	8
Apr 4, 2017	Brukerhistorie 4: Som bruker øn...	Skreddersy push varsler til å kom...	1
Apr 5, 2017	Brukerhistorie 4: Som bruker øn...	Aktivering/implementering av et ...	8.5
Apr 5, 2017	Brukerhistorie 4: Som bruker øn...	Skreddersy push varsler til å kom...	1
Apr 6, 2017	Brukerhistorie 4: Som bruker øn...	Aktivering/implementering av et ...	3
Apr 6, 2017	Brukerhistorie 4: Som bruker øn...	Skreddersy push varsler til å kom...	7
		Total	**65.5**

Robin A. Rondestvedt Moudnib

When	Backlog Item	Task	Spent
Mar 27, 2017	SPLIT Report writing IS-304	Document implementation issue...	0
Mar 27, 2017	SPLIT Business part	Negotiate terms of Studiehjelpen...	3
Mar 27, 2017	SPLIT Report writing IS-304	Sprint 2 review / retrospect	6
Mar 27, 2017	SPLIT Report writing IS-304	Scrum-master work	1
Mar 27, 2017	Report writing IS-304	Sprint 3 review & retrospect	4
Mar 27, 2017	Report writing IS-304	Sprint 4 planning	2
Mar 28, 2017	SPLIT Brukerhistorie 2: Som bruk...	Fikse kodebase på Raspberry Pi, ...	7
Mar 30, 2017	SPLIT Brukerhistorie 3: Som en b...	Å interacte med "thingen" → Hen...	4
Mar 30, 2017	SPLIT Brukerhistorie 3: Som en b...	Appen interacter med mobile hu...	5
Mar 30, 2017	Business part	Meetings with potential clients	5
Apr 3, 2017	SPLIT Brukerhistorie 3: Som en b...	Appen interacter med mobile hu...	5
Apr 3, 2017	Report writing IS-304	Scrum-master work	2
Apr 4, 2017	SPLIT Report writing IS-304	Guidance from lecturer with inp...	1
Apr 4, 2017	Brukerhistorie 4: Som bruker øn...	Tilrettelegge scripts på Pi til å ku...	7
Apr 5, 2017	Brukerhistorie 4: Som bruker øn...	Aktivering/implementering av et ...	6
Apr 7, 2017	Report writing IS-304	Scrum-master work	2
Apr 7, 2017	Brukerhistorie 4: Som bruker øn...	Aktivering/implementering av et ...	7
Apr 7, 2017	Report writing IS-304	Steering commitee meeting	4
		Total	**71**

Ricky Omland

When	Backlog Item	Task	Spent
Mar 28, 2017	Report writing IS-304	Sprint 3 review & retrospect	4
Mar 28, 2017	Report writing IS-304	Sprint 4 planning	2
Mar 28, 2017	SPLIT SPLIT Brukerhistorie 2: So...	Skrive tester for funksjonene vi i...	4
Mar 29, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	4
Mar 29, 2017	Brukerhistorie 4: Som bruker øn...	Research på AWS SNS for push v...	1
Apr 3, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	6
Apr 4, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	7
Apr 5, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	6
Apr 6, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	9
Apr 7, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	1
Apr 7, 2017	Report writing IS-304	Steering commitee meeting	4
		Total	**48**

Erik Oskar Zetterquist

When	Backlog Item	Task	Spent
Mar 27, 2017	Report writing IS-304	Sprint 3 review & retrospect	4
Mar 27, 2017	Report writing IS-304	Sprint 4 planning	2
Mar 28, 2017	Report writing IS-304	BMC - report	1.5
Mar 28, 2017	SPLIT Brukerhistorie 2: Som bruk...	Fikse kodebase på Raspberry Pi, ...	4.5
Mar 29, 2017	Brukerhistorie 4: Som bruker øn...	Research på AWS SNS for push v...	1
Mar 29, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	5
Mar 30, 2017	SPLIT Report writing IS-304	brukertesting papir-prototype	9
Apr 3, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	7
Apr 4, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	7
Apr 5, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	8
Apr 6, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	6
Apr 7, 2017	SPLIT Report writing IS-304	Rapportskriving øvrige	1
Apr 7, 2017	Report writing IS-304	Steering commitee meeting	4
		Total	**60**

From Mar 27, 2017 to Apr 7, 2017 total spent time is 306.5 hours