





# Final team reflection - LifeStyle Application

DAT257 - TEAM 10













## My Lifestyle


< 2020-05-28 >




### Water intake 1,6 L



### Register your sleep



4 : 0



13 : 37


























Count Sleep 9 hours 37 minutes

### ToDo List







Team reflection ☒ Delete

Add

## Rate your day

DAT257					
Maria					
Team 10					
Håkan					
JP					

### Register todays weather



## Journal

Dat257 har varit en väldigt bra kurs! Väldigt bra handledare och lärare

Delete Save

## Medlemmar:

Hanna Tärnåsen  
Linnea Johanson  
Ebba Richnau  
Hanna Bergland  
Oscar Forsberg  
Johanna Wiberg  
Naomi Espinosa

# Innehållsförteckning

Customer Value and Scope	<b>1</b>
Stakeholder	1
Mål och kravsättning	1
Mockupen & framtagande av GUI	2
User Stories	4
KPI:er	6
Omfattning och skalbarhet	7
Social Contract and Effort	<b>10</b>
Effort	10
Social Contract	11
Generellt	11
Förväntningar på medlemmar - Specifika paragrafer	11
Kommunikation	12
Design Decisions and Product Structure	<b>14</b>
Design decisions	14
Tekniska dokumentationsmetoder	15
Användandet och uppdaterandet av dokumentation	16
Kodkvalité och kodstandarder	17
Andra reflektioner	19
Application of Scrum	<b>20</b>
Sprints	20
Scrum meetings	20
Scrum board	21
Scrum masters	22
Team reflection	23
Product owner	23
Sprint review och DoD	24
Empowered teams	25
Vertikala arbetsuppgifter	25
Utveckling inom Scrum och agila arbetssätt	26
Kunskapssökande	26

# Customer Value and Scope

## Stakeholder

I det första skedet av detta projekt behövde vi definiera vårt scope, alltså vad för typ av produkt vi skulle ha, för att sedan hitta en passande stakeholder vars krav vi kunde anpassa applikationen utefter. Efter en brainstorming-session i gruppen fastnade vi för idén att bygga en ”Lifestyle” dagbok där man kunde föra ner sina tankar, mål och data om sin dag, väldigt likt det analoga konceptet av ”Bullet Journals”. Då en i gruppen kände någon som brukar använda ”Bullet Journals” och liknande sätt att reflektera över sin livsstil, frågade vi denna person. Hon accepterade och nästa steg i processen var att hålla en intervju kring hennes krav, samt att bygga en första mockup med den nya informationen vi fått.



(Bildkälla 1: [Garrity & Schumer](#), 2019)

(Bildkälla 2: [Page](#), 2020)

## Mål och kravsättning

A:

Vid starten av projektet diskuterade vi med vår stakeholder om minimal viable product där vi fick en tydlig bild om vad som måste vara med för att hon skulle vilja använda sig utav applikationen. Stakeholdern ville prioritera funktionerna journal, day rating, sleep, to do-lists och möjligheten att byta dag i en kalender. Hon tyckte även att designen var väldigt viktig. Sedan tyckte hon att det hade varit bra om det fanns funktionalitet där man kunde lägga till dagligt vattenintag och privata inställningsmöjligheter. Detta tyckte hon dock inte vi borde prioritera högt till en början. Vi pratade även med henne om våra egna idéer där hon rankade dessa efter hur viktiga hon tyckte de var. Målbilden som sattes upp var att vi skulle hinna med att göra, enligt vår stakeholders prioriteringar, en snygg applikation som hade all funktionalitet som hon hade prioriterat högt.

Mot slutet av projektet ändrade vår stakeholder sig om en komponent. Hon såg inte längre poängen med att hennes eget namn skulle finnas längst upp på applikationen, utan ville hellre att man skulle kunna ändra vad som stod där i settings page.

**B:**

Vi borde ha använt oss mer av vår stakeholders tankar om minimal viable product. Trots att vi hade en tydlig bild av vad som var vår minimal viable product var, glömde vi att använda oss av den under de första sprintarna av projektets gång. Vi diskuterade mer om hur vi skulle prioritera vår user stories och vad som vi själva tyckte var värdeskapande när vi tog beslut. Vi hade också några tillfällen vi inte höll med vår stakeholder. Till exempel ville hon att vattenintagskomponenten skulle vara lågt prioriterad medan vi i gruppen tyckte den adderade stort värde till applikationen. Vi valde att lita på oss själva och gå emot vår stakeholder. I detta fall fungerar det bra då vår stakeholder faktiskt blev nöjd med vår prioritering. Å andra sidan är det riskfyllt att gå emot sin stakeholder och något som potentiellt skulle kunna förstöra förtroendet. Vi kanske bara hade tur att vår stakeholder gillade våra ändringar.

Vi i gruppen förde aldrig någon diskussion i början om varken individens- eller gruppens utveckling vilket senare har tagits upp på vårt sista handledarmöte. Där diskuterade vi om att projektet har gått så pass smidigt att man nu i efterhand kanske hade kunnat satsa på att göra något mer tekniskt utmanande för att utveckla oss själva ytterligare som programmerare. Dock har dessa tankar kommit ut i slutet av projektet när vi redan lärt oss om arbetssättet och om hur gruppen fungerar tillsammans. Med facit i hand hade vi nog klarat av en svårare utmaning med de kunskaper vi har nu men dessa kunskaper hade vi inte för 8 veckor sen.

**A→B:**

Genom att vid varje sprint planning försöka hålla stakeholders åsikt i fokus hade vi kunnat undvika att blanda in våra egna tankar i diskussionen. Vid varje oklarhet är det mer värt att be om förtydligande från vår stakeholder och/eller diskutera gruppens åsikt för att förklara varför gruppen inte håller med stakeholdern.

I början av projektet är det värt att hålla i ett mötet där individuella mål och förväntningar diskuteras.

## Mockupen & framtagande av GUI

**A:**

Efter att ha hållit en inledande intervju med vår stakeholder fick vi både data om vad som var minimal viable produkt och även pekare på hur hon ville att det grafiska gränssnittet se ut. Efter detta satte vi igång med att skapa en mockup. Syftet med att skapa en mockup var att dels att se

till så att vår grafiska bild av applikationen liknar hennes, samt att layouten och funktionaliteten av vissa komponenter som ratings fanns där hon ville ha dem och var utformade på ett sätt som hon tänkt.

Det grafiska användargränssnittet samt hur man presenterar datan är nästan den viktigaste komponenten för användare av Bullet Journals och därför la vi stor vikt vid att lyssna på hennes krav och preferenser kring applikationens utseende. Även hon betonade att design var en av de saker som kunde avgöra om hon skulle använda applikationen eller inte. Därför var mockupen ett viktigt hjälpmedel för att få input redan vid utformningen av komponenter som ratings, olika sidor etc.



Bild på mock-up:en som gjordes innan sprintarna startade.

Mockupen skapades i ett digitalt verktyg som kallas Figma. Där gjorde vi ett gränssnitt för vår homepage, den första sidan man möts av när man startar applikationen. I denna försökte vi skapa en stil, layout som byggde på vår stakeholders svar, samt utforma de komponenter (som skulle finnas med på homepage) på sidan.

Mockupen var till en början väldigt bra då den gav oss något att referera till när vi skulle börja med att skapa vår homepage och komponenterna som låg där, något som var det första vi gjorde i projektet. Detta gjorde att man slapp fundera på hur alla komponenter skulle få plats på sidan och hur man skulle uttrycka funktionaliteten i gränssnittet på bästa sätt - det var redan uttänkt i vår mockup som då godkändes av vår stakeholder.

Eftersom vi bara gjorde en mockup för vår homepage skapades det lite frågetecken i ett senare skede av arbetet där vi inte visste riktigt hur vi skulle designa komponenterna på andra sidor, samt hur layouten där skulle se ut. Detta gjorde att varje person som jobbade på komponenter eller sidor gjorde det lite efter eget behag och tolkning. Det skapade lite mer jobb i slutet när vi skulle se till så att alla sidor och komponenter såg liknande ut.

#### **B:**

Något som vi borde gjort var att göra en mockup för resten av sidorna i vår design. Detta hade gjort det lättare när man började på en specifik komponent att veta hur man skulle utforma den grafiskt, men även så att man slapp freestyla lite på vart på sidan man ville ha de olika sakerna.

Något vi gjorde i början av projektet var att vi sa att vi skulle fokusera på backend samt att placera ut komponenter i GUI:t och koppla samman med vår backend, och låta css:en och övrig design fixas sist. Att vi då hade en mockup klar för vår homepage gjorde att när vi kom till detta skede kunde det gå snabbt. Detta var något som inte gick lika snabbt eller var lika självklart när det var dags att fixa de andra sidorna då vi inte hade en klar bild eller design av dem.

#### **A→B:**

Något vi kan tänka på i framtiden är att få mer feedback på utformningen av specifika komponenter och deras funktionalitet i gränssnittet, samt att göra en mer omfattande mockup som visar alla sidor i applikationen. Detta kan ses som mycket att göra i början, men förslagsvis skulle man kanske kunna börja med en sida, programmera i den veckans sprint. När man sedan väljer att börja jobba på en ny page, och dess tillhörande komponenter, kan man sammanställa en ny mockup av den tänkta sidan som man visar stakeholders och får feedback på. Detta sätt känns mer iterativt, ger feedback direkt samt klargör för alla som jobbar på den sidan vad de har att jobba med samt hur det ska utformas stil och layoutmässigt.

För övrigt var det väldigt givande att använda sig av just figma för att skapa vår mockup/skiss av applikationen. Det är för att man kan välja hur interaktiv man vill ha den (går att klicka sig fram mellan olika sidor/komponenter), går smidigt att utforma och leka med olika design idéer samt att man kan överföra designen väldigt lätt till css-filer.

## User Stories

#### **A:**

Strax efter att vi haft vårt första möte med vår stakeholder, där vi fått en bättre inblick i vad hon vill ha med i applikationen, började vi skriva user stories. Vi skrev ned alla som vi diskuterat med stakeholders i ett tidigare möte i applikationen trello. Vi försökte göra alla stories så små som det bara går för att få en tydligare bild av vad det var som skulle göras. Varje user story

hade olika etiketter på sig. Etiketter som beskrev hur prioriterad ett user storyn var: "High Priority", "Medium Priority" eller "if there is time". Andra etiketter beskrev om den hade med det visuella eller funktionalitet att göra. Senare lade vi också till en etikett i denna kategori som handlade om tekniska svårigheter för att se till att det jobb som lades ned på detta inte gick okänt. Den sista etiketten rankade user storyn mellan easy, medium och hard. Denna etikett användes främst som material till vår burn up chart.

Vi bestämde att vi skulle lägga tasks på alla user stories när vi tog på oss dom. Alltså: om jag tog på mig till exempel tre user stories till en sprint skulle jag personligen skriva tasksen för dessa i början av sprinten.

### **B:**

I det stora hela fungerade våra user stories under hela projektets gång men det var några småsaker som vi fick ändra med tiden. Till exempel insåg vi i mitten av projektet att det läggs ner mycket tid på tekniska svårigheter och att lösa teknisk skuld. Dessa är saker som får projektet att gå framåt och borde speglas i vår burn up chart. Det är ju också synd att vissa personers hårda arbete annars går okänt.

Vi hade också lite problem med vårt system om att lägga till tasks i början på varje sprint. Under flera sprintar glömde vi att lägga till tasksen på förhand och dessa lades istället till i slutet på varje sprint, men då försvinner hela poängen med dem. Dessutom skapar detta förvirring inom gruppen. När man ska sätta sig in i ett helt nytt problem(user story) kan tasksen vara ett hjälpmedel för dig för att veta vart som skall göras eller eventuellt tänka igenom problemet och komma fram till vad som ska göras.

Det fungerade inte heller jättebra med att betygsätta varje user story efter easy, medium and hard då det är väldigt subjektivt. När alla medlemmar har olika kunskaper blir olika uppgifter olika svåra. Dessutom kan något som är väldigt lätt ändå vara väldigt tidskrävande och tvärtom. Då är det egentligen inte så rättvist med den betygsskalan.

### **A→B:**

Om vi hade lagt till tasksen på user storiesarna tillsammans i början av projektet hade det antagligen kunnat underlätta arbetet längre fram. Då hade vi kunnat bolla med varandra hur man löser user storyn på bästa sätt. Det hade också varit tydligare för alla förstå och komma överens om vad som ska ingå i varje user story.

Det kan vara värt att använda sig av ett annat system än easy, medium och hard när man rankar user storiesarna. En mer rättvis bild hade kanske varit om man gav alla user stories en poäng som var baserad på hur svårt personerna som skulle utföra user storyn är blandat med hur

tidskrävande den blev. Det är definitivt värt att ta en diskussion om hur denna typ av rankning ska ske efter eller innan uppdraget är utfört. När man gör uppgifter blir de sällan hur man tänkt sig från början.

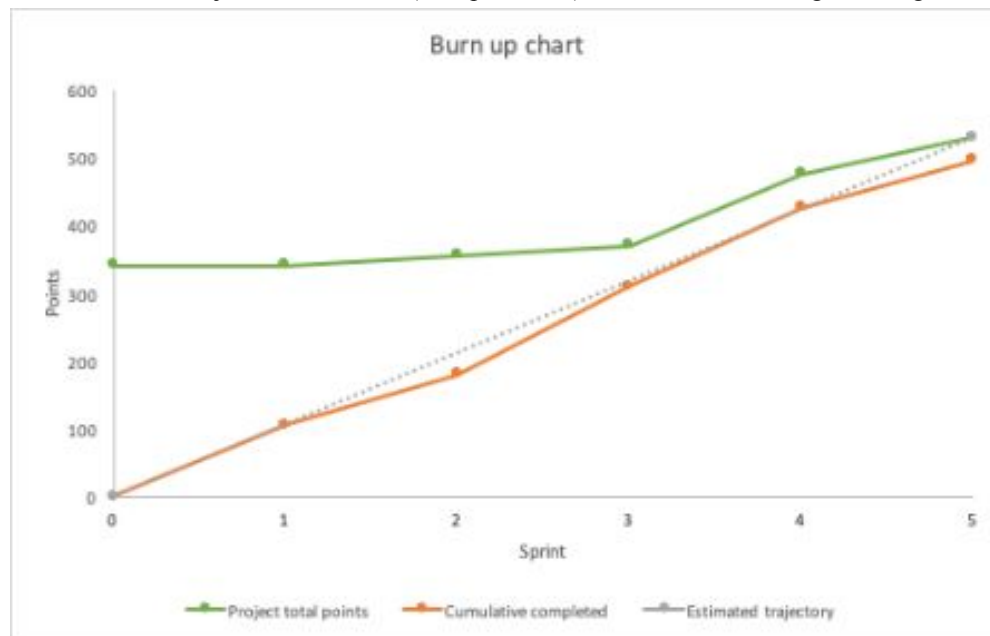
## KPI:er

Sprint	Snitt
1	3,9
2	3,5
3	3,5
4	4,0
5	3,7

Sprint	Betyg
1	7
2	7
3	8
4	7
5	8,5

Fist of five med varje individs känsla (snitt per vecka).

Progress enligt stakeholder (Skala 1-10).



Burn-up chart

### A:

En av de tre KPI:er som vi använde oss av var fist of five där alla medlemmar skulle ranka sitt eget mående från 1 till 5. Vi räknade ut ett snitt på den siffran varje vecka som vi sedan kunde använda för att se hur vårt mående påverkade projektets framsteg, eller om det är hur projektet går som påverkar hur vi mår. Vi märkte till exempel att veckan då vårt projekt fastnade som mest är även den veckan som vår fist of five var lägst. Den andra KPI:en vi använde oss av var



stakeholder satisfaction, där vår stakeholder betygsatte våra framsteg mellan 1 och 10. Detta var för att vi i gruppen skulle få en bättre bild av vad vår stakeholder såg som värdeskapande. Den tredje KPI:n ändrades med tidens gång. Från början valde vi att använda oss av burn down chart men efter diskussion med Håkan på ett handledarmöte kom vi fram till att ändra till burn up chart istället.

#### **B:**

Trots att vi varje vecka utfört våra KPIer så tog gruppen inte riktigt vara på det värdeskapande man kan få ut av dem. Vi i gruppen har inte riktigt gjort något med resultatet från dessa utan mer bara sagt hur varje sprint ser ut utifrån varje KPI. Om vi satt oss ner och reflekterat över resultaten istället hade vi eventuellt kunnat förstå varför resultatet blev som det blev och då förstå när det går bra för oss och när det går dåligt.

Vi har inom gruppen märkt att stakeholder satisfaction är väldigt användbar då det essentiellt är att kontinuerligt få återkoppling på projektet. Dock har vi diskuterat att det finns en risk att rating på en skala 1-10 inte är så nyanserat, vilket kan göra det svårtolkat eftersom många olika aspekter spelar in i betyget. Särskilt eftersom hon inte märker av mycket av det backend-arbete som vi gör eller problemlösning relaterat till tekniska svårigheter.

Anledningen till varför vi bytte från burn down chart till burn up chart var att det är svårt att från början veta hur många poäng som vi under projektet skulle hinna med. De var först när förslaget från Håkan kom om att byta som vi fick reda på att burn up chart fanns. Hade gruppen vetat om båda varianterna hade antagligen burn up chart valdes från början.

#### **A→B:**

Generellt hade gruppen gynnats av att tänka igenom våra val av KPIer lite noggrannare.

Stakeholder satisfaction hade man kunnat utforma lite annorlunda. Till exempel hade man kunnat be stakeholders att betygsätta olika delar av projektet och inte bara helhetsintrycket.

Eventuellt även be henne lägga till kommentarer eller motiveringar till hennes betygsättning så man får en bättre bild av vad det är hon är nöjd med respektive missnöjd med.

I och med att vi missade att burn up chart fanns så borde vi kollat igenom lite bättre vad det finns för olika KPIer och taktiker. Eventuellt kanske vi missat någon annan KPI som vi gärna velat använda.

Gruppen borde även använda sig lite mer av KPI-resultatet när vi utvärderar våra sprintar.

## **Omfattning och skalbarhet**

#### **A:**

Något värt att fundera på och som kommit upp som en diskussionspunkt är problem och saker att tänka på om appen börjar användas kommersiellt av flera användare. Hur ser vi på den personliga datan som sparas, integritet och hur vi ska förhålla oss till GDPR.

Just nu finns det stöd i applikation för att byta användare på varje enhet, men datan sparas fortfarande lokalt på varje dator i en JSON-fil. Något som vi inte lagt särskilt stor vikt vid är att kryptera varje användares data när den sparas ner. Detta är för att vi i första hand tänkt att den bara ska användas på ens personliga dator, men detta innebär även att om man förlorar tillgång till sin maskin där Lifestyle-app körs på, förlorar man all sin data. För att se till att detta inte händer, hade det behövt sparas remote i någon moln-service, och även detta medför nya problem att oroa sig kring.

Ifall detta var något vi bestämt oss för att göra hade vi behövt utforma applikationen och användaravtal för att förhålla oss till GDPR på ett lagligt sätt. Att sälja vidare användarinformation till en tredje part är i sådana fall en diskussion vi hade behövt ta innan utformning av detta arbete påbörjades. En viktig aspekt av alla produkter ute på marknaden är hur man ska kunna tjäna på den. I vårt fall hade just att sälja vidare användarinformation, reklam alternativt betalt medlemskap vart våra fokuspunkter.

En annan sak vi lade stor vikt på när vi skulle bestämma idé för projektet, var modularitet och skalbarhet. Att kunna börja med något mindre som MVP för att sedan skala upp och bygga funktionalitet och backend som enkelt kan anpassas för utökning var viktigt för att känna att vi gjorde projektet utifrån våra förmågor samtidigt som det gav utrymme för att testa nytt och pröva sig fram.

#### **B:**

Även om vi inte väljer att backa upp användares data i en moln-service, hade vi fortfarande kunna lägga mer åtanke till hur vi kan säkerställa säkerhet och integritet av varje användares lokala data, speciellt när man börjar tänka att det är mer än en person per dator som kommer använda applikationen.

I gränssnittet är vår applikation utformad som att det bara är en användare, därav finns inget behov för en login-sida. Vi har dock utformat backenden så att ifall vi ville ha funktionalitet för flera användare hade vi endast behövt lägga till det i vår frontend. I detta fall hade det varit relevant att fundera på ett användaravtal, där användare får gå med på att datan sparas lokalt, samt att även andra användare data kan sparas lokalt på datorn.

#### **A→B:**

Inför nästa projekt är det viktigt att lägga mer tanke kring om det främst är fler eller en användare som ska kunna använda applikationen, och även om det är för en dator eller flera datorer.

Något vi gjorde bra var att utforma backenden så att den är skalbar, där vi kunde lagt till funktionalitet för flera användare om vi hade haft tid / ansett det passande. Att applikationen har gjorts så pass skalbar och modulär som den är tror vi har bidragit något enormt till minskad

stress inför vår arbetsbörda. Vi har alltid känt att vi tagit på oss något genomförbart där vi sedan kan lägga till mer funktionalitet i framtiden, vilket gör att man kan lämna större beslut till längre fram i projektet när man har mer kunskap.

Det är även alltid värt att lägga ganska stor vikt vid säkerhetsfrågor i början av projektet; krypterar vi användares lösenord och data? Finns det ett behov? Detta är något vi inte tog särskilt stor hänsyn till i detta projekt.

# Social Contract and Effort

## Effort

### A:

Vid snack om effort så har man sett dalar och toppar om man skulle lägga upp efforten i en graf. Det som har varit absolut mest tidskrävande har varit merge problem när det kommer till github. Ja, det har sina fördelar att alla kan jobba i olika bransches och smälta ihop dem till en. Men det har dock blivit problem flertal gånger då vissa mindre arbetsgrupper glömt att uppdatera sin master version. På detta sätt har nykod skrivits över när andra redan har mergat med master. Den har även stannat upp lite vid ett annat specifikt tillfälle. Dessa tillfällen har uppstått då man fysiskt suttit tre personer och kodat i skolan till exempel. Man har då uppfattat det som att det är svårt att alla är delaktiga, att alla ska kunna se skärmen osv. Det är då svårt att ge all sin kunskap då det är restriktioner som förhindrar en.

### B:

I en ideal värld vi aldrig behöva sitta i totalt flera timmar för att lösa dessa problem och mergen till master skulle gå smidigt utan teknisk skuld sprinten därefter. Men tyvärr ser det inte direkt ut så. Det skulle inte bara spara tid, men även onödig frustration och temperament som påverkar humöret, vilket även då påverkar ens prestation. Helst vill man ju att sitta fysiskt tre pers ska funka men eftersom vi är sju personer och brukar dela upp oss 3-2-2 så fungerar det inte riktigt att dela upp oss på något annat sätt då vi värderar input från varandra högt. Att någon ska sitta själv varje vecka är därför något vi inte vill.

### A→B:

Men för att uppnå detta kom gruppen på en ide att lägga till en definition of done i det sociala kontrakt i sprint 3. I denna utökning av kontraktet är det skrivet att för ens uppgift ska anses som godkänd för veckan så ska den uppfylla speciella krav. En av dessa punkter är att koden i din bransch ska vara inlagd i mastern innan du kan se dig själv som klar. På detta sätt har gruppen på ett smidigt sätt löst problemet genom att utöka vårt sociala kontrakt. Detta har på lång tid gett gruppen en bättre framförhållning varje vecka då man inte kan skicka in sin kod i sista sekund och hoppas på att det ska fungera. Om det då skulle vara problem så kan vi som nödfall hjälpas åt så att vi slipper teknisk skuld till kommande sprint. Lösningen till andra problemet kom på av en slump. Lösningen var att man skulle kunna trio-programmera på distans och allt löser sig. Känslan förändrades drastiskt, man behöver inte slåss om plats. Alla kan turas om att dela sina skärmar och på så sätt kan alla vara delaktiga och koda så att alla lär sig något. Till nästa projekt är detta en viktig sak att ha i åtanke från början, så att ingen känner att man inte kan bidra för att det är för fullt kring datorn.

# Social Contract

## Generellt

### A:

I början på projektet så satte sig gruppen tillsammans och skapade ett social-contract som en typ av grund för hur man som individer ska fungera tillsammans. Nu i efterhand ser man saker man velat ha med från börjat eller ändrat tidigare. Vi insåg till exempel under de första två veckorna att gruppen inte hade den bästa kommunikationen om arbetsfördelning och åsikter.

### B:

Vi vill att projektet skulle ha en tydlig riktning/vision varje vecka, så att man känner att sina insatser lägger grund för nästkommande sprint och att alla har koll på varandras ideer och tankar kring arbetet.

### A→B:

För att skapa en bättre miljö, där var och en vet vad man ska göra, så införde man i vecka 2 bestämda tider för möten varje vecka så att kommunikationen är kontinuerlig och säker.

## Förväntningar på medlemmar - Specifika paragrafer

### A:

Den första saken att uppmärka i vårt kontrakt är att det står:

“... Man har själv ansvar för att sin del blir klar. Vid problem är man skyldig att meddela resten av gruppen så att problemet går att lösa.” vilket kan vara lite problematiskt. Den andra saken att uppmärka i vårt kontrakt är att det står: “Alla förväntas lägga ner lika mycket tid.”. Men tiden har man märkt att vissa kan mer/mindre än andra. Att då kräva att alla lägger ner exakt lika mycket tid kommer inte att funka då vissa löser sina user stories mycket snabbare än andra. För en person tar en uppgift 5 minuter och anses som enkel. För en annan tar det 5 h och anses som svår. Den tredje och sista saken att uppmärksamma är att paragrafen närvaro borde kollas upp: “Alla förväntas ha en rimlig närvaroprocent.”. Vad menas med rimlig? Och det är även svårt att räkna ut procenten när vi inte har en kontinuerlig lista på närvaron. Detta skulle kunna betyda att personer som inte aktivt deltar lätt kan komma undan. I detta projektet så har dock ingen känt något sådant men i framtiden i större projekt så kanske man inte har en lika nära/personlig grupp som vi har haft nu som kan träffas fysiskt varje möte i skolan. Då är det svårt att hålla koll på alla personer.

### B:

Nu i efterhand skulle vi velat ha den mer omformulerad så att man förstår att det inte är okej att meddela resten av gruppen 10 min innan deadline då vi även har en punkt under “kritik” som säger “Det är inte okej att framföra missnöje över någon annans arbete i sista minuten.”. Man

hinner inte ge konstruktiv kritik på något som läggs upp direkt innan deadline (konstruktiv kritik måste kunna ges och ändras innan nästa sprint då nästa sprint bygger på den.).

Därutöver så vill man ju i ett sånt här projekt att alla i gruppen ska utmanas lika mycket av sina arbetsuppgifter på ett sätt som gör att alla ungefär sitter lika mycket kring projektet. Detta för att få en så rättvis uppdelning av arbetet. På något vis vill man förebygga att den som kan mer blir klar på 5 min och att den som kan mindre sitter och stressar hela dagen för att bli klar med något som någon annan kunde gjort mycket snabbare.

#### **A→B:**

De specifika paragraferna är ju bara nu att ändra och ha i åtanke till framtida projekt. Men problemen vi framhåvt med dessa paragrafer har vi i gruppen ändå löst helt okej under arbetsgången genom att ibland successivt skicka bilder på idéer/resultatet i vår kommunikationskanal så att alla är uppdaterade och kan tycka till. För att inte det ska bli kaos och liknande i kanalen ibland så skulle man kunnat diskutera om att lägga in en dag mitt i veckan där man rapporterar hur det har gått, och om folk behöver hjälp/vill ge kritik. På detta sätt ges det utrymme för att fixa dessa sakerna på de resterande dagarna som är kvar på sprinten. För att lösa detta skulle man kunna tänka på detta vid uppdelning av arbetsuppgifter, något vi haft **lite** i åtanke (men inte så mycket).

För att förbättra arbetsfördelningen (så att alla utmanas och lär sig så mycket som möjligt), som vi sagt innan, borde svårighetsgraden på user stories bestämmas utefter de kunskaper personen som ska göra uppgiften innehar och inte utefter gruppen generella åsikt om svårighetsgraden. Just nu har vi endast arbetat kring att man lägger till svårighetsgrad efteråt när man är klar. En svårighetsgrad som utdelas innan folk börjar med sina uppgifter skulle kunna vara det första steget att minska skillnaden på tid man lägger ner i gruppen. För att förhindra att någon i framtida projekt fuskar med närvaron skulle man först och främst kunnat skapa en excel-fil med allas närvaro så att alla vet vart det finns och att alla ställs inför samma förutsättningar då man bestämmer ett procentantal alla ska uppnå.

## Kommunikation

### **A:**

Som diskuterats i texten ovan och i flera team-reflections genom projektets gång så har det varit lite strul med kommunikationen i gruppen (främst i början). Vi pratade först om att det kan vara för att vi inte haft regelbundna möten tillsammans de första två veckorna. Men vi tror även att det beror på vårt val av kommunikationsmedel. Det var till en början en diskussion mellan användandet av mobilappen Slack eller Facebooks messenger. Det var lite delade åsikter där om man säger så. Några hade problem med slacks notifikationer och några använde aldrig messenger. Så tillslut så valdes det indirekt Slack som kanal för projektet. Nu i efterhand är det

kanske lätt att se att detta inte har funkat då det varit kommunikation i båda medel (dock över 90 % messenger). Plötsligt blev valet Slack som skrevs in i kontraktet men alla var nog inte med på beslutet, det skedde mer indirekt med tiden. Detta bidrog nog till stor del till den lite sämre kommunikationen till en början. Det har hänt flera gånger att mötena krockar med personers andra planer. Schemat har ser olika ut för alla i gruppen från vecka till vecka. När en Scrum-master var sjuk/frånvarande så uppstod det lite problem för strukturen och kvaliteten av mötet. Det gick dock helt okej ändå tack vare att scrum master rollen var uppdelad i tre och att det fanns tydlig dokumentation.

## **B:**

Vi skulle vilja uppnå en godare kommunikation i ETT medel. På detta sätt kan man lättare lägga fokus på ett ställe och man vet vart man har allting. Vi vill alltså ha tydliga val när vi väl gör val i gruppen, inte något som rinner ut i sanden som sedan gör att vi väljer något snabbt bara för att vi är tvingade till det. Förhastade val är aldrig bra. Helst skulle också gruppen vilja att alla kan delta på alla möten så att alla är "up to speed" när det gäller tankeprocessen kring vårt arbete. Om det är så att någon inte kunnat medverka så vill man alltid ha en backup-plan i alla lägen.

**A→B:** För att slippa det skulle gruppen kunnat haft en omröstning till nästa projekt (vilket även står i vårt kontrakt att vi ska ha vid olika åsikter) om vilket kommunikationsmedel vi ska använda. Men detta glömdes av. För att förebygga detta problem till nästa projekt tycker vi att man skulle ha lagt till ett segment på de veckovisa mötena där man går igenom det sociala kontraktet och tar upp saker som behöver läggas till/tas bort/korrigeras. På detta sätt kanske vi skulle ha märkt att vi kanske skulle ha ändrat några formuleringar i kontraktet som diskuterats häröver i texten. Utöver att lägga till en del i mötena där man går igenom kontraktet så är det också en bra ide att lägga till en del i mötet där man går igenom schemat för gruppen, både individuellt och för gruppen. På detta sätt kan man lätt meddela de andra i gruppen om förhinder i god tid. Till exempel gjorde påsken och kristi himmelfärd så att den normala veckan blev mycket kortare. På detta sätt kan man även planera om tiden för ett speciellt möte om det skulle gå. Vi märkte till en början att om Scrum-mastern var sjuk så kunde vi minska förlusten av denna person genom att dela upp denna roll i flera mindre roller. På detta vis klarade vi oss bättre i dessa situationer. Då fick även mer folk mer ökad förståelse till scrum och tillhörande processer/rutiner. Det är även mycket viktigt här att ha ett valt kommunikationsmedel så att alla vet var man ska lägga frågor till nästa gång.

# Design Decisions and Product Structure

## Design decisions

A:

I projektet har vi inte tänkt på alltför många mönster. Mycket av processen har flutit på och vi har valt några större mönster att följa. Dessa är följande: Model View Controller (MVC), Observer pattern, Single responsibility principle och Open-closed principle. Alla dessa mönster bestämdes vecka två förutom Single responsibility principle och Open-closed principle. Dessa var mer eller mindre underförstådda och en riktning snarare än ett tydligt mönster

I bas till projektet följs MVC detta är bra och nästintill ett måste när vi arbetar med det visuella på detta sätt. Dock är inte detta indelat i tre delar utan är mer i 2 då view och controller är sammankopplade eftersom respektive FXML har en controller. Detta anses dock ändå följa mönstret då applikationen kan utnyttja dess fördelar. Fördelen men detta är att viewcontrollern (VC) kan kommunicera med en representant från applikationen hellre än att hela VC behöver ha koll på hela modellen. Dock har varje controller koll på "MainModel" som är en del av modellen vilket kan ifrågasättas men med tanke på projektets storlek är detta definitivt en bra lösning. Vi har använt oss av Observer pattern för att uppdatera respektive klasser när någonting den klassen är intresserad i uppdateras. Detta har varit ett väldigt bra system som förbättrat kodstrukturen då de antingen finns internt i VC eller kan hämtas för "MainModel" där man lagt till Observers.

Single responsibility principle har varit av stor vikt i kodandet där varje klass har haft en minimal uppgift och kan beskrivas med en mening eller mindre. Detta är bra ur det aspekt att ingen klass blir för tung. Om något är eventuellt "MainModel" för tung men den är en agerade mellanhand som kan hämta information från resten av modellen och föra vidare det till VC.

Open-closed principle (OCP) är ett väldigt löst baserat mönster. Många delar har haft detta hårt implementerat. Exempelvis modellen där man använder användare för JSON. Detta är väldigt öppet och vill vi ha mer än en användare kommer detta vara väldigt lätt att lägga till. Däremot implementeras det inte alltid i andra delar av koden. Exempelvis var många metoder i VC väldigt hårdkodade vilket innebar att när ändringar behövde ske behöver vi göra om allt.

Just nu följer koden nästan Dependency injection men några klasser bryter mot detta. Exempelvis en rating som har en rating och använder sig av dess metoder. Detta skapar relationer som inte är bra då det skapar coupling i koden.



## **B:**

Något vårt projekt verkligen saknar är först att förbättra Open-closed principle. Detta är en otroligt viktig princip att ha, då det i nuläget är väldigt svårt att vidareutveckla koden. Det skulle därför vara önskvärt att ha en mer anpassad kod.

Dessutom skulle det vara bra med mer fokus på Dependency injection. Nu används MainModel till lite av detta men det används inte helt på rätt sätt. Det skulle vara bra för att skapa mer simplicitet i koden och minska coupling.

Low in coupling and high in cohesion är ytterligare något som hade gynnat koden lite med samma argument som Dependency injection. Coupling gör att projektet blir rörigt och en liten ändring kan helt plötsligt få flera följder.

## **A→B:**

För att åstadkomma de flesta av dessa ändringar handlar det mycket om att gå igenom koden och refaktorera.

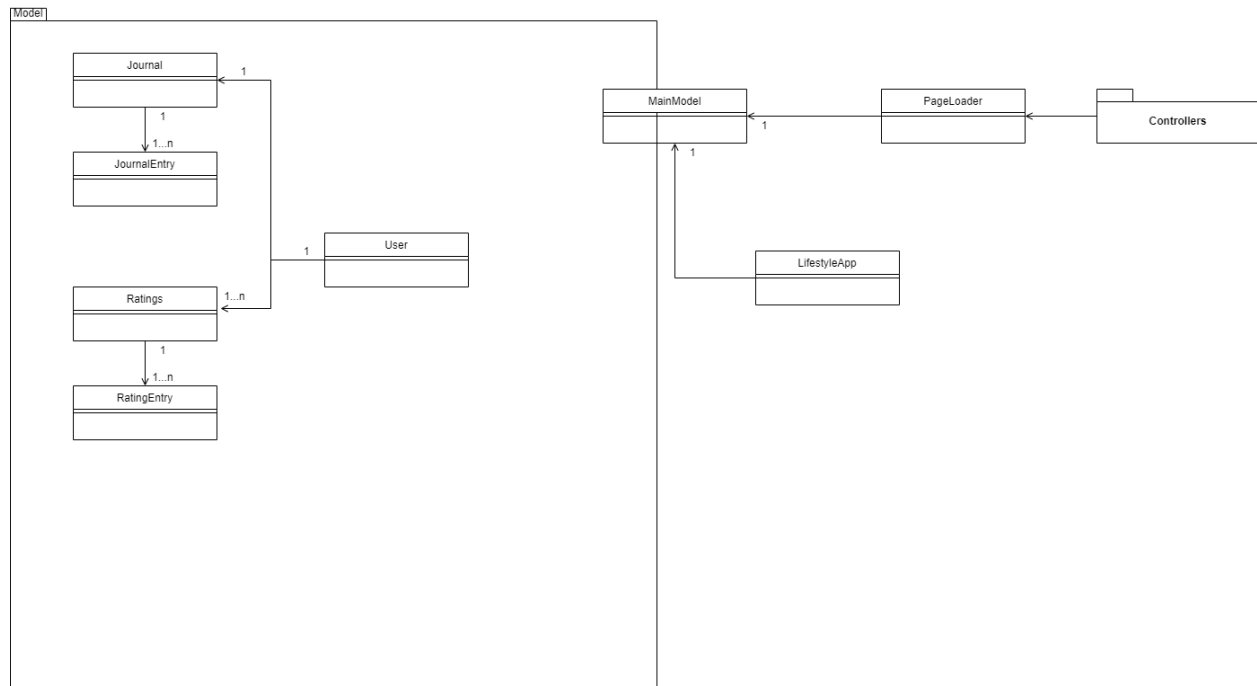
För att förbättra OCP handlar det om att refaktorera och tänka om i uppläggen på samtliga klasser i VC. För Low coupling och high cohesion samt Dependency injection är det viktigt att implementera dessa på rätt på rätt sätt. Många klasser innehåller metoder från andra klasser än MainModel. Detta hade relativt enkelt kunnat lösas med att helt enkelt strukturera upp koden bättre.

Allt detta kan vara en lång process men det vore en nödvändigt om projektet var i större skala. Med storleken projektet som den är nu så är detta inte överdrivet nödvändigt då det funkar som det ska och det blir inte alltför stora konflikter när man försöker ändra i koden.

## **Tekniska dokumentationsmetoder**

### **A.**

Vi valde att inte använda oss av särskilt många tekniska dokumentationsmetoder. I början ritade vi upp en domänmodell som skulle skapa tydlig strukturering av projektet samt skulle hjälpa till att få en överblick över applikationens olika funktioner samt hur dessa hänger ihop. Även Use cases skapades för att definiera hur applikationen kan användas. Det skulle bidra till att gruppen reflekterar över användningen. Vår tanke var att Use cases senare kunde användas tillsammans med vår stakeholder för ökad förståelse. Domänmodellen och våra Use cases glömdes sedan bort och användes inte i praktiken.



Domänmodellen som ritades upp i början av projektet.

## B.

Vi borde ha uppdaterat domänmodellen och utöka Use cases under hela projektets gång, samt använda sig av fler metoder (till exempel klassdiagram, dokument som beskriver hur vi ska strukturera koden etc).

## A→B:

För att ha dragit nytta av dessa skulle de kunna uppdateras under processens gång och diskuteras på möten. Detta hade däremot tagit mycket tid från möten som annars kunnat gå till annat, så det skulle krävas en balans.

## Användandet och uppdaterandet av dokumentation

### A:

Under projektets gång uppdaterades vår Definition of done. Beslutet var att konceptet även ska inkludera att koden ska mergeas in master, vilket innebär att det måste beaktas vid estimering av user stories. Detta skrevs även in under vårt Social contract som vi redan diskuterat i tidigare delar av denna Team reflection. Utöver det ska Definition of done användas vid code review och vi kom internt överens om att vara lite mer noggranna med detta. Beslutet härstammar i att det är viktigt att vara hårda mot varandra för att säkerställa kvaliteten på respektive del som ska läggas till i mastern. Ännu en ändring var att framöver hjälpa till att testköra varandras branches för att hitta buggar som kanske inte är direkt uppenbara för skaparen av koden. Vidare var målsättningen att dela upp stora pull requests i mindre delar som därmed skickas oftare, i syftet

att underlätta merges och review. Genom att meddela varandra när nya uppdateringar av mastern sker kunde vi också uppdatera kontinuerligt för att motverka konflikter.

Vårt social contract uppdaterades några gånger, och vårt Burn down chart ändrades till ett Burn up chart. Som diskuterats ovan uppdaterades varken domänmodellen eller use cases under projektets gång.

**B:**

Dokumentationen som skapades borde ha använts i större utsträckning under möten och uppdaterats mer vid behov, vilket även diskuterats i tidigare delar av denna team reflection. I vissa fall skapades överflödiga klasser och om vi haft en uppdaterad domänmodell hade vi kunnat se vad som skulle kopplas till modellen.

**A→B:**

Ett tips från vår handledare i projektet var att kontinuerligt uppdatera modellen då det kan underlätta förståelsen för programmets uppbyggnad. Detta hade som diskuterats i B använts för att undvika vissa aspekter av den tekniska skulden som uppstod. För att uppdatera det mer kontinuerligt hade det krävts att i början diskutera det i större utsträckning så att alla håller med om vikten av detta.

## Kodkvalité och kodstandarder

**A:**

Gällande kodkvalitet var ambitionen i början att vanliga kodstandarder skulle följas, som exempel att alla skulle kommentera sin kod för att göra det lättare att förstå. Kodkommentarer var en punkt vi inte efterföljde vår ambition, utan detta användes väldigt sparsamt. Däremot använde vi separata Git-branches och pull requests, vilket möjliggjorde att andra medlemmar i gruppen kunde kontrollera kod innan den mergades med mastern. I början av projektet accepterade vi ofta våra egna pull-requests vilket efter ett handledningstillfälle uppmärksammades.

**B:**

Vår ambition att kommentera koden är ett exempel på vad som borde ha gjorts. Tanken var att det var ännu viktigare än vanligt under kursen då vi arbetar i större grupp och man inte alltid har koll på de andra medlemmarnas arbete. Alltså var tanken god men detta syntes inte i vårt arbete. Att acceptera egna pull-requests är absolut inte positivt, det borde vara en annan person som gör detta för att fånga upp problem och fel i koden.

**A→B:**

För att detta skulle efterföljas skulle det krävas att varje individ i gruppen la lite tid till att skriva kommentarer. Kodkommentarer är dock inte värdeskapande för vår stakeholder, då hon inte ser dem eller har nytta av dem och därför prioriterades detta inte högt i vårt projekt. I ett framtida projekt hade det däremot varit en idé att ha fler kommentarer, särskilt om det är ett större projekt där det i större utsträckning krävs att man sätter sig in i andra personers kod. En anledning till att det har gått bra för oss i detta projekt utan kommentarer är för att vår grupp varit en storlek där man enkelt har kunnat fråga den som skrivit koden i fråga. Vi valde att inte ha en designerad person att acceptera pull-requests, men det hade kanske varit fördelaktigt eftersom man då är säker på att någon har det ansvaret och alltid ser till att kolla detta.

## Andra reflektioner

### A:

För att spara data så applicerades JSON i projektet. Detta är ett smidigt och funktionellt system att enkelt spara data. Strukturen för detta är väldigt öppet uppbyggd då vi har ett interface för persistence. Detta innebär att om vi i framtiden vill byta databas-system så är detta väldigt simpelt att lägga till, så länge den implementerar interfacet.

För att implementera JSON fanns det många incitament att använda sig utav Gson. Detta är ett externt bibliotek som behövdes hämtas. För att få tag på denna dependency så implementerades ett automationssystem, Gradle. Det är ett fördelaktigt system som är allmänt känt och finns i IntelliJ. Ett problem med detta var dock att vi insåg detta mot projektets slut. Vi behövde därför implementera detta senare vilket blev stökigt då implementering av Gradle var svårare än förväntat, vilket även diskuteras i Team reflection 5. Problemen gick dock alla lösa men tog längre tid än förväntat.

Det finns alternativ till både JSON och Gradle. JSON hade kunnat bytas ut mot SQL och Gradle hade kunnats bytas ut mot Maven. Överlag så är JSON något vi alla kände till någorlunda struktur på vilket blev anledningen till valet. Anledningen till valet av gradle är då det är det främst använda systemet. Dessutom är det väldigt användarvänligt.

### B:

Implementeringen av JSON och Gradle fungerade generellt sett bra men vi hade kunnat göra det tidigare i projektet. Gradle hade kunnat funnits sedan sprint 1 medan JSON hade kunnat skapas under sprint 2 exempelvis.

### A→B:

JSON och Gradle hade som sagt kunnat implementeras tidigare men då hade det behövts bestämmas redan i början av projektet. Som det var nu prioriterade vi att få applikationen att fungera innan vi implementerade att data skulle kunna sparas för en användare. Det hade kunnat inkluderas mer tydligt i vår Minimal Viable Product (MVP) så att det var förbestämt att det skulle prioriteras tidigare.

# Application of Scrum

## Sprints

**A:**

Vi hade sprints omfattande en vecka som officiellt löpte fredag till fredag. Anledningen var kursens upplägg med veckovisa reflektioner som skulle lämnas in på fredagar. Totalt omfattade projektet fem stycken sprints.

**B:**

Vi hade kunnat ha längre sprints, omfattande exempelvis två veckor istället för en. Det stöds av diskussioner med en utomstående person som i arbetslivet arbetar med längre sprints. Reflektionerna hade då antagligen fått gälla en halv sprint.

**A→B:**

Att sprintarna är relativt korta gör att marginalerna i estimeringen blir små och mötena tar upp relativt stor del av tiden. Tidsbegränsningen kan liknas vid vad som upplevdes vid Mona Lisa-övningen och planeringen och reflektionen blir därför viktig för att lyckas hantera arbetsbördan på ett bra sätt. Vissa veckor var även ännu kortare på grund av påsk och Kristi-himmelsfärd. Det talar för att längre sprintar skulle vara fördelaktigt.

Å andra sidan ger kortare sprinter fler möjligheter till reflektion, vilket har bidragit till att vi lärt oss mer under kursen. På grund av kursens upplägg hade det varit svårt med längre sprints än en vecka. Det är dock något vi kan ta med oss till framtida projekt och eventuellt även arbetslivet

## Scrum meetings

**A:**

Vi valde att slå samman våra möten sprint planning, sprint review, och sprint retrospective till ett enda stort möte som hölls varje måndag som avslutning på förra veckans sprint och inledning av nästa. Med det upplägget kunde själva arbetet avslutas på fredagen, med helgen som eventuell felmarginal och gemensamt möte på måndagen. Vi hade även ett gemensamt möte på torsdagen då team reflection diskuterades och strukturerades upp tillsammans. Mötena var strukturerade så att ansvarig Scrum master inom ett visst område styrde diskussion och antecknade i ett drive-dokument.

**B:**

Vi borde kanske ha delat upp mötena så att vi hade sprint review och retrospective på fredagen och sprint planning på måndagen. Det hade eventuellt gjort mötena kortare och mer fokuserade

inom ett specifikt område. Dock förlorar man då felmarginalen som helgen erbjuder som chans att färdigställa sprintens user stories.

Ett annat alternativ är att ha flera kortare möten, till exempel dagliga standup-meetings för mer kontinuerliga uppdateringar. Utöver det kan strukturen med anteckning förändras så att man kör datorlösa möten för bättre fokus, då missar man dock chansen till dokumentation.

#### **A→ B:**

I kommande projekt kan mötena delas upp för att de ska bli mer effektiva och fokuserade. I så fall skulle sprint review och retrospective ske i slutet av varje sprint och planning i början på nästa. Trots att det hade tagit upp mer tid hade det alltså kanske varit fördelaktigt att diskutera en sak i taget.

Även dagliga standup-meetings eller möten utan dator hade kunnat användas men på grund av kursens uppbyggnad valdes det bort. Då vi inte arbetade varje dag med projektet hade det känts överflödigt att varje dag ge en uppdatering på framstegen man gjort. Även datorlösa möten hade varit svårt då dokumentationen utgjorde ett viktigt underlag för veckans team reflections. Dock är det koncept man kan ta med sig till framtida projekt eller arbetslivet då det kanske passar bättre sådan utformning av projekt.

## Scrum board

#### **A:**

Vi har använt oss av en Scrum board i form av en gemensam Trello med rubrikerna Backlog, ToDo, In Progress, Testning, Sprint X – Complete samt Final deliveries. Där har vi lagt in user stories med task breakdown och efforts i form av labels med antingen easy, medium eller hard. Vi har även lagt till tekniska kort som inte är programmering, vilka representerar team reflections och scrum meetings. Även tekniska problem har lagts till som kort då de tar upp tid och energi under sprinten att lösa dessa. Korten har delats upp mellan gruppens medlemmar enligt överenskommen arbetsfördelning. Dock hade vi kunnat uppdatera Scrum boarden oftare för att hålla bättre koll på vad som ska göras respektive vad som är gjort och vilka tasks som ingår. Trellon har även legat som grund till KPI:n burn-up chart då den diskuterar vad som gjorts under varje sprint.

#### **B:**

Uppdatera Scrum-board direkt efter att ändringar har gjorts för att hålla övriga teamet uppdaterade och undvika en ”uppdateringsskuld” i slutet av sprinterna.

**A→B:**

Kontinuerlig uppdatering kan uppnås genom att tillsammans diskutera frågan och komma överens om gruppen gällande hur ofta och när det ska göras. Eventuellt kan man även lägga till exempelvis i social contract eller DoD att Scrum board ska uppdateras efter varje utförd user story eller liknande.

## Scrum masters

**A:**

Efter första veckorna, när sprinterna började, valde vi att dela in rollerna så att en person behöll samma roll under hela projektets gång. Vi hade tre olika scrummasters: en för sprint planning, en för sprint review och en för retrospective. Detta gjorde att de tre personerna kunde växa in i sin roll vilket alla kände att de gjorde. Samtidigt undvek vi problem vid sjukdom genom att sprida ut ansvaret, likt diskuterat under avsnittet Kommunikation. En sak som blev negativt med detta var dock att resterande i gruppen inte fick chansen att testa. Samtidigt hade det istället kunnat leda till att ingen hade känt sig säker i rollerna då man bara fått chansen att testa någon enstaka gång och inte växa i rollen.

Ytterligare en tanke är kring hur det varit om vi haft en scrum master som höll alla tre delarna. Detta skulle göra att personen som var scrum master fick en större helhetsbild och kanske en mer verklighetsbaserad erfarenhet. Samtidigt skulle det ju göra att ännu färre i gruppen fick möjlighet att få erfarenhet som scrum master vilket vi ändå ville att flera i gruppen skulle få.

En tydlig rollfördelning tillsammans med väl utförd dokumentation gjorde att det var lättare att fördela ut rollen tillfälligt när en person var sjuk. Dock kan det ju då istället vara svårt för personen som tillfälligt tar på sig rollen att veta vad som ingår som scrum master.

**B:**

Det varit bra om alla hann testa rollen som scrum master för att alla ska testa på att känna det ansvaret och få växa in i den rollen. Det är trots allt skillnad på att göra något i teorin (genom att lära sig om roller på en föreläsning) eller att testa det i praktiken.

**A→B:**

Om framtida projekt sträcker sig över en längre tid kan det vara värt att byta scrum masters efter ett par sprintar för att alla ska få den praktiska erfarenheten. Det skulle också kunna lösas genom att de personer som har varit scrum masters får berätta om sina erfarenheter för gruppen.

Hade projektet sträckt sig längre hade man kanske också kunnat testa att vara scrum master för alla tre delar för att få helhetsperspektivet. Dock hade detta kunnat medföra att antingen en



mycket längre projektperiod krävts, att färre fått testa att vara scrum master eller att man haft ansvaret under en kort period.

## Team reflection

**A:**

En person i gruppen hade som ansvar att samla in gruppens tankar under varje sprint och sammanställa ett utkast till en team reflection. Denna team reflection gick sedan igenom under torsdagsmötet. Då lades det till/togs bort grejer baserad på den diskussion som uppstod på mötena.

**B:**

Vi hade kunnat skriva alltihopa i grupp tillsammans för att fler än en person ska skriva själva texten.

**A→B:**

I så fall hade vi kunnat skriva team reflection under torsdagsmötena. Det skulle dock göra att mer tid lades på att faktiskt få ner tankar i text istället för vidare diskussion. Med det sättet vi gjorde nu sammanställdes allas tankar och sedan kunde diskussionen tas vidare från det och utvecklas under torsdagsmötena vilket förhoppningsvis lett till en bättre reflektion.

## Product owner

**A:**

Vi hade en i gruppen som var product owner och höll kontakten med vår stakeholder, vilket var väldigt bra för att undvika att det blir rörigt för stakeholdern om vem som varit kontaktperson. Det var dock tur att denna personen aldrig var sjuk eller borta då vår PO satt på all information från vår stakeholder.

**B:** Vi hade kunnat ha flera product owners för att undvika krångel om en person blir sjuk (speciellt nu med corona).

**A→B:** Om vi haft två product owners varav en var huvudansvarig och en var med som support hade vi garderat oss bättre vid eventuell frånvaro. Det hade heller inte gjort det mer rörigt för vår stakeholder då en person varit huvudansvarig men att support personen ändå varit med och träffat stakeholdern så att det inte helt plötsligt blir ett nytt ansikte vid frånvaro.

**A:**

Varje sprint demonstrerade vår PO produkten för vår stakeholder och fick input. Detta gjorde att vi vid sprint review visste vad vår stakeholder tyckt om det vi gjorde. Vi fick också input om vad

som var mer eller mindre värdeskapande för vår stakeholder vilket vi kunde använda i vår sprint planning. Under sprint planning kunde vi då prioritera om utifrån vad stakeholdern senast sagt. Dock tappade vi ibland fokus på vår stakeholders krav för MVP vilket gjorde att det lätt blev att vi kom in i våra egna tankar om vad som skulle prioriteras. På så sätt var det tur att vår product owner hade ett möte i veckan med stakeholdern för att påminnas om dennes prioriteringar.

**B:**

Vi kunde haft MVP bättre integrerat i våra tankar under hela projektets gång.

**A→B:**

Det kunde vi gjort genom att ha varit noggrannare vid sprint review och sprint planning med att gå tillbaka till vad vår stakeholder beskrivit som MVP och tänka och prioritera utifrån det. Det kunde vi ha gjort tillsammans med hur vi nu gjorde med att se till stakeholderns veckovisa värderingar för att undvika problem i slutet. Nu gick det bra för oss ändå för vi hann med det som var tänkt, men till framtida projekt kan det vara bra att ha detta i åtanke.

## Sprint review och DoD

**A:**

Som tidigare nämnt hade vi från början ingen tydlig definition of done vilket ledde till en del problem för oss. Det gjorde det svårt under sprint reviewen att avgöra om något var helt klart eller inte. Genom detta satte vi oss själva i teknisk skuld då vi trodde att vi var klara med något men sedan vid merge så fungerade det inte och därmed låg vi back vid start på nästa sprint. Detta diskuterades vi en handledning i mitten av projektet och en tydlig DoD lades då till i vårt social contract och efter det hade vi inte de problem som uppstått i början.

**B:**

Ha en tydlig DoD från start, men med möjlighet till utveckling om så behövs.

**A→B:**

För att uppnå detta kan det vara värdefullt att vid kommande projekt ta sig tiden att fundera över hur gruppen definierar vad som är färdigt redan från start och försöker tänka tillbaka på tidigare situationer (som till exempel början av detta projekt). På så sätt undviker teamet att få den tekniska skuld som vi fick i början av detta projekt. I efterhand tror vi att tiden som gick åt till att lösa tekniska problem nog var betydligt längre än den tid det tagit för oss att vid start bestämma en tydlig och välformulerad DoD. Under projektets gång kan det också vara bra att uppdatera teamets DoD för att få den ännu tydligare och mer välanpassad för projektet.

## Empowered teams

### A:

Vi har utgått ifrån det arbetssätt som Jonas Florvik på Husqvarna berättade om med Empowered teams. Det innebär att vi har arbetet med mycket frihet under ansvar eftersom vi delat upp arbetsuppgifter inom gruppen för att sedan lämna det upp till de ansvariga när samt hur de löser dessa. Det har gjort schemat mer flexibelt och lösningarna mer kreativa eftersom det har funnits mycket rum för olika typer av lösningar, så länge man kommer fram till ett acceptabelt slutresultat som uppfyller Definition of Done.

### B:

Man hade kunnat strukturera upp tydligare hur olika uppgifter skulle lösas, genom användning av exempelvis en domänmodell som visar hur strukturen ska vara uppbyggd.

### A→B:

Som diskuterat tidigare hade vi kunnat uppdatera vår domänmodell under projektets gång för att tydligare strukturera upp de uppgifter som skulle utföras, även om vi valt att använda Empowered teams. Tillsammans skulle de två faktorerna kunna bidra till ett ännu mer samordnat och välstrukturerat slutresultat med flexibla arbetstider och frihet under ansvar.

## Vertikala arbetsuppgifter

### A:

Vi hade som mål att ha ett projekt som går att dela in i vertikala delar, likt exemplet med tårtskivor från föreläsningen. Det åstadkoms eftersom vi hade ett modulärt uppbyggt projekt med olika separata funktioner som kombinerades i en och samma applikation. Det var också fördelaktigt vid planering eftersom vi snabbt kunde bygga upp en fungerande produkt som sedan kunde utvecklas genom tillägg av fler funktioner.

Till största del fungerade det alltså bra då vi hade mest vertikala deluppgifter. Dock frångicks det till viss del i slutet av projektets gång då vi på grund av bristande kommunikation fick en del konflikter vid merge av olika delar till master. Det berodde på att de sista sprintarna innehöll en del övergripande designaspekter som omfattade många klasser, och gjorde det svårare att programmera separat.

### B:

Ännu mer vertikal uppdelning av arbetsuppgifter för att undvika konflikter.

### A→B:

Målet kan nås genom att verkligen eftersträva att alltid ha vertikal indelning av olika delar av

projektet. Dock kan det vara svårt gällande slutgiltig design och liknande. Då kan konflikter istället undvikas genom tydligare kommunikation inom gruppen där man redogör för vem som arbetar på vilka delar av koden.

## Utveckling inom Scrum och agila arbetssätt

### A:

I början var det lite stapplade gällande hur gruppen bäst skulle applicera Scrum och agila arbetssätt under projektet. Då ingen av gruppens medlemmar hade tidigare erfarenhet tog det lite tid att sätta sig in i vad de olika koncepten innebar och hur vi kunde använda oss av det. Det är alltid svårt att ta information direkt från en föreläsning och försöka applicera det på projektet på ett bra sätt. Det gjorde att mötena i början tog längre tid och att det inte alltid var helt självklart vad olika delar av mötet skulle innefatta.

Under projektets gång har vi dock utvecklats och fokus har flyttats från att förstå vad Scrum innebär till att faktiskt kunna applicera det på ett fördelaktigt sätt, vilket har medfört mer effektiva och givande möten. Det har även gjort att vi kan fokusera mer på att skapa värde för vår stakeholder, vilket har gjort oss mer produktiva allteftersom projektet fortgått.

### B:

I kommande projekt kommer vi förhoppningsvis till viss del kunna undvika den startsträcka det innebär att vi inte hade arbetat med Scrum tidigare.

### A→B:

Förhoppningsvis kan vi applicera det vi lärt oss i kursen i framtida projekt för att snabbare komma in i rutinerna det innebär att arbeta i ett agilt projekt med Scrum. Dock innebär ett nytt projekt med ett nytt team alltid en viss omställning, något vi diskuterade med Håkan Burden under ett handledningstillfälle. Förhoppningsvis blir dock som sagt den startsträckan mindre i och med att vi har reflekterat över och kan applicera våra lärdomar från kursen.

## Kunskapssökande

### A:

För att lära oss om Scrum använde vi oss främst av föreläsningarna som erbjöds och var aktiva där. Utöver det kollade vi på youtube om det var något vi inte riktigt förstod, till exempel kollade flera av oss på denna video Introduction-to-scrum. Vi har också haft kontakt med andra grupper samt fått veta om andra gruppers strategier genom vår handledare. Detta har gett oss nya perspektiv under vägen gång till exempel kring vilka KPI:er vi valt och hur vi kunnat göra annorlunda. Slutligen har vi pratat med en bekant till en i gruppen som arbetar med Scrum i verkliga livet och som vi bland annat hade en intressant diskussion om längder på sprinter med.

**B:**

Eftersom vi inte gjort så många projekt som detta innan så kan det vara svårt att ta med sig kunskap från tidigare projekt. I framtida projekt kan det dock vara värt att ta lärdom av både egna och andra gruppers erfarenheter. Det är även bra att komplettera föreläsningar med videos eller litteratur från andra källor för att få en djupare förståelse.

**A→B:**

Genom att dokumentera sina erfarenheter och reflektioner likt vi gör i denna kurs blir det enkelt att gå tillbaka. Framtida projekt kan också innebära nya team vilket ger fler erfarenheter som kan användas för att välja strategier i gruppen. Youtube, bloggar och andra lättillgängliga källor fungerar bra för att komplettera kunskapen som förmedlas under föreläsningarna för att få en djupare förståelse.

Snipp snapp snut nu var team reflections slut