CHALMERS
UNIVERSITY OF TECHNOLOGY

DAT256 - Software engineering project

Course responsible:

Håkan Burden

Examiner:

Jan-Philipp Steghöfer

Final Team Reflection - Hoi

Group:

Tangela

Members:

Jesper Berglind

Henrik Edstrand

Edvin Leidö

Ivan Lyesnukhin

Karin Sjödin

Carlos Yechouh

VT 2019

Customer Value and Scope

1. the chosen scope of the application under development including the priority of features and for whom you are creating value

A:
Vår applikation, *Hoi*, är en plattform för cyklar där uthyrare och hyrare kan mötas. Grundtanken är att sidan ska vara enkel och användarvänlig. Efter att ha påbörjat diskussioner med cykelglada människor har vi identifierat att de behöver veta var cykeln befinner sig, vilka datum den är tillgänglig, typ av cykel, storlek och antal växlar. De vill också att processen ska vara snabb och enkel och att betalningen ska ske med hjälp av en erkänd betalningslösning. Därför behöver till exempel hyraren aldrig skapa ett konto, utan kan hyra en cykel direkt genom att betala med sitt kreditkort. Uthyraren behöver däremot skapa ett konto för att vi ska kunna administrera utbetalningar till dennes Paypal-konto.

I dagsläget fungerar det att som uthyrare 1. Skapa ett konto, 2. Lägga ut en cykel med ett datumintervall då den är tillgänglig, 3. Se cykeln på sin profilsida och som hyrare kan man 1. Söka fram en cykel man vill ha med rätt plats, typ, antal växlar och ram, 2. Genomföra betalning. 24 timmar (eller mindre om sen bokning görs) innan cykeln ska hämtas upp delas kontaktinformation mellan de två parterna. Hela köpprocessen fungerar alltså i sin enklaste form och vi skapar värde för uthyraren som nu kan hyra ut cykeln som står och skräpar i garaget. Vi skapar också värde för hyraren som har fått en möjlighet att hyra cyklar under längre perioder (minst en dag) jämfört med Styr och Ställ eller liknande som ofta har en maxgräns på 30 minuter per hyrd period.

B:
Framöver vill vi inleda ett samarbete med ett cykellås, Bitlock, för att underlätta för tjänstens användare. I dagsläget ligger ansvaret på uthyraren att se till att cykeln når hyraren. Om man använder Bitlock kan cykelns position kommuniceras från låset, via Hoi, direkt till hyraren. Detta innebär att man kan ställa en cykel vid Göteborgs central som sedan hyr ut sig själv gång på gång och genererar intäkter av sig själv. Förutom Bitlock så vill vi också inleda ett samarbete med en försäkringspartner. I dagsläget kan vi ta straffavgifter från hyraren om cykeln lämnas tillbaka i dålig skick eller om den inte återlämnas alls. Detta kan dock skapa viss friktion vilket gör att hyraren inte vågar hyra cykeln. Istället vill vi ha en försäkringspartner som erbjuder försäkring i samband med köp. Förutom en tryggare tjänst kan detta också generera en ny intäktsström till Hoi, då vi blir mellanhand mellan försäkringsgivare och försäkringstagare.

A->B:
För att åstadkomma samarbetet med Bitlock har vi tänkt att en uthyrare ska behöva värva 5 nya kunder till Hoi, och då kan vi tillsammans med Bitlock skänka ett lås.

Låset kostar 129 USD enligt Bitlocks hemsida. Vi tänker att de kan gå ner lite i pris, då de säkert också vill satsa på tillväxt och få ut många lås på markanden. För Hois del är tillväxtsiffrorna viktigare än att vi ska göra vinst så snabbt som möjligt. Om användarbasen växer med 5 kunder, som i sin tur värvar 5 nya kunder och så vidare, så kan denna tillväxt väga upp för Hois marginalkostnad att köpa in ett lås.

Vad gäller samarbetet med ett försäkringsbolag så tror vi inte att det blir lika svårt. Om försäkringsgivaren visar stort intresse lägger vi till försäkringen som en del av vårt produkterbjudande som alltid ingår, för att förenkla och ta bort hinder i processen. Annars läggs möjligheten till att teckna försäkring för hyrare/uthyrare när man genomför köp, vilket gör att den används om parterna anser det vara värt det. Kostnaden läggs då på utöver våra kostnader.

Vi tänker oss att aktörer så som försäkringsbolaget Hedvig kan vara intresserade, då de redan profilerar sig som en digital aktör, riktar sig mot unga samt sätter enkelhet i fokus. Om inte de är intresserade kommer vi vända oss till mer traditionella försäkringsaktörer.

> 2. the success criteria for the team in terms of what you want to achieve within the project (this can include the application, but also your learning outcomes, your teamwork, or your effort)

A:
I början av projektet bestämde vi oss att göra en webbapplikation som ska kunna erbjuda följande tjänster för de två typer av användarna (hyrare och uthyrare):

- Hyrare ska kunna välja datum när den vill hyra en cykel
- Hyrare ska kunna se beskrivning av en cykel
- Hyrare ska kunna se cykels placering
- Hyrare ska kunna betala på ett enkelt sätt
- Uthyrare ska kunna lägga till cykel, dess tillgänglighet och placering

Eftersom ingen i gruppen hade någon erfarenhet i React och JavaScript var vi lite rädda i början för att alla dessa kriterier inte ska kunna fyllas. Det absolut lägsta acceptanskriterium för oss blev ett körbart program som skulle kunna visa någonting, samtidigt bestämde vi att fylla de ovanstående kriterierna beroende på hur bra och snabbt vi lär oss webbprogrammering. Men för att bli nöjda med vårt arbete så ville vi att de fyra ovanstående punkterna skulle vara med i vårt program.

Å andra sidan, ville vi lära oss någonting nytt som vi aldrig arbetat med tidigare, därför bestämde vi oss att utveckla vår applikation i React och JavaScript. Anledningen var också att just det språket är bäst anpassat för webbprogrammering.

Så vårt andra mål blev att lära sig React, JavaScript samt nå en sådan nivå i dem så att man ska kunna använda dessa verktyg för att skapa webbplattformar.

Med andra ord, våra kriterier kunde delas på två delar: den funktionella och den tekniska. Det funktionella innebär att applikationen ska ha en viss funktionalitet som är beskriven ovan, samtidigt som den tekniska innebär att vi ska få en tillräcklig hög nivå på kunskaper inom ett valt programmeringsspråk. De två delarna hänger ihop med varandra, eftersom om man inte kan den tekniska, så kan man inte göra någonting i den funktionella, samtidigt som om man kan den tekniska men inte kan tillämpa det på rätt sätt så blir resultatet samma. Därför försökte vi att göra båda.

I dagsläget kan vi säga att vi har nått alla kriterier som vi ställde framför oss i början av projektet. Applikationen erbjuder en mängd av olika tjänster, man kan välja datum, se placering, genomföra betalning, man kan registrera sig och lägga sina egna cyklar och allt detta sparas i lokala databasen.

Vi har också lärt oss grunderna av React och JavaScript och vi kan använda dem för att skriva olika program.

När det gäller acceptanskriterium för koden så gör vi det på följande sätt. Eftersom varje person i gruppen får en task att genomföra under sprinten, skriver vi alltid detaljerad information av vilka element den ska innehålla, detta var skrivet i punktform. Efter man har fyllt alla dessa punkter, får man acceptera tasken. Men för att flytta tasken till “Done” kolumnen i Trello, så ska man också merga in sin kod med Master-branch utan att skapa konflikter, med andra ord taskens kod måste vara i master för att fullgöra tasken.

B:
Om vi hade fått mer tid på oss så skulle vi fortsätta utveckla vår applikation. Vi hade tänkt oss att täcka större geografiskt område, dvs inte bara Göteborg, utan resten av Sverige också, och på sikt, hela världen. Mer funktionalitet ska också kunna läggas till och särskilt vill vi fixa betalningsprocessen så transaktioner utförs på ett säkert och kontrollerat sätt. Dessutom ska bägge parter få ta del av en lyckad eller misslyckad transaktion i form av automatgenererade mejl.

Vi behöver även hosta applikationen på ett antal webbservrar för att hantera HTTP förfrågningar. Dessutom behöver vi ett domännamn vilket kommer att göra vår plattform tillgänglig världen runt.

A->B:
För att nå den nivån och för att utöka vår plattform skulle vi fortsätta nästan exakt som vi gjort hittills. Vi skulle behöva lite mer resurser, eftersom för att hosta ett antal webbservrar (alternativt låta någon tredje part hantera denna del) och förvärva ett domännamn behöver man betala. När det gäller utökning av funktionalitet så skulle vi fortsätta på samma sätt som vi gjort tidigare: dela uppgifter bland alla och sedan

genomföra dem. Samtidigt ska vi fortsätta lära oss React för att lösa komplexa problem. Arbetet under senaste 2-3 veckorna har fått väldigt bra flytt, vi har kommit igång med projektet och ifall vi hade haft fler sprintar så skulle vi absolut kunna utöka applikationen med fler funktionalitet.

> 3. your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation and how this influenced the way you worked and created value

A:
Eftersom gruppen hade väldigt begränsade erfarenheter av att använda user stories och tasks så fick vi initialt lägga mycket tid på just denna del av projektet. Dessutom hade de flesta i gruppen aldrig tidigare stött på de olika verktyg vi behövde använda för att skapa vår webbapplikation, vilket gjorde det näst intill omöjligt att sätta en rättvisande effort på varje task.

Under arbetets gång förstod vi, tack vare handledarens input, att våra tasks egentligen var user stories och därmed alltför stora. Vi behövde dessutom göra fler mergear under sprintarnas gång för att undvika onödiga omarbetningar eller att någon gruppmedlem inte kunde arbeta på grund av sekvenseringen av de tasks vi skrivit.

Under projektets första veckor skrev vi user stories och tasks baserat på vad vi trodde skulle behöva göras för att färdigställa projektet. Därefter delade vi upp dem och tog i tu med dem. Mot arbetets slutfas, däremot, skrev vi ofta tasks under arbetets gång för att vi då upptäckte buggar eller andra saker som behövde fixas för att säkerställa att vi stod med en färdig produkt inför redovisningen.

När det gäller acceptanskriterier så skrev vi ofta dessa i samband med att tasks bildades och i dessa fall låg det på den som var ansvarig för tasken att se till att acceptanskriterierna uppfylldes. De tasks som skrevs löpande under arbetets gång fick tyvärr inte alltid acceptanskriterier.

B:
I framtida projekt vill vi redan i ett tidigt skede kunna dela in arbetet i lämpliga “tårtbitar” och tilldela dessa en mer passande effort. Dessutom behöver vi ha tydligare “regler” för när vi mergar och hur vi delar upp arbetet för att det ska fortlöpa på ett så smidigt sätt som möjligt.

En bättre struktur, mer standardiserade processer och mer tydlighet hade vi definitivt haft hjälp av i andra delar av projektet också, såsom när det gäller acceptanskriterier och uppdelning av tasks.

A->B:
Vi lärde oss otroligt mycket om detta arbetssätt under det här projektet, vilket vi tar med oss framöver. Vi har nog alla mycket att vinna på att arbeta med andra med mer erfarenhet, eftersom vi då kan snappa upp än mer kunskap om hur man planerar och organiserar ett software-projekt. Dessutom kan vi utvärdera de olika sprinterna under arbetet med Hoi och därmed se hur många effort points varje person faktiskt klarar av under en vecka och anpassa framtida projekt efter det. Förhoppningsvis blir vi även bättre och bättre på att använda alla verktyg vilket borde innebära att vi kan ta på oss mer effort per person och vecka framöver.

4. your acceptance tests, such as how they were performed, with whom, and which value they provided for you and the other stakeholders

A:
När vi påbörjade projektet hade vi ingen tidigare erfarenhet av JavaScript. På GitHub konfigurerade vi Travis CI, continuous integration-tool, som skulle genomföra automatiska tester. Förutom det testade vi också små funktionalitetskomponenter manuellt efter hand som de byggdes. När samtliga acceptanskriterier för en task var färdiga samt inga varningar fanns i console-fönstret mergeas mastern in till den aktuella branchen. När ev. merge konflikter är lösta kan pull request skapas. Taskens kort på Trello flyttas i sprint backlog från *In progress* till *Testing*. Därefter behöver **två** utomstående gå in på github och reviewa koden. (I samband med att vi arbetade för att få mindre tasks ändrade vi också så att bara **en** utomstående behövde reviewa) Den som reviewar koden ska dels titta på koden och förstå lösningen. Därefter kör man programmet och testar all funktionalitet som hittills har byggts, alltså både den nya men också allt som fanns sedan tidigare. Om allt fortfarande fungerar kan man acceptera, annars får man rejecta och ge feedback till den som skapade pull requesten. Efter två godkännanden mergeas den aktuella branchen in till master, och kortet flyttas till *Done* på Trello.

B:
Även om dessa tester fungerar att genomföra samt säkerställer att man inte “förstör” allt som fungerar i master branchen när man utvecklar ny funktionalitet så kan man fortfarande inte veta att alla genomför Testing så noggrant som det är sagt. För att höja gruppens kompetensnivå och för att inspireras av varandras programmeringsfärdigheter är vår testmetod ändå bra, då man får läsa mycket kod och lära sig felsöka manuellt. När vi inte längre är nybörjare i React/JavaScript känns det dock aktuellt att börja använda Travis CI i högre utsträckning.

A->B:
I kommande utvecklingsprojekt ska vi skriva egna enhetstester som automatiskt kan köras av Travis för att säkerställa att nyutvecklad funktionalitet fungerar på önskat

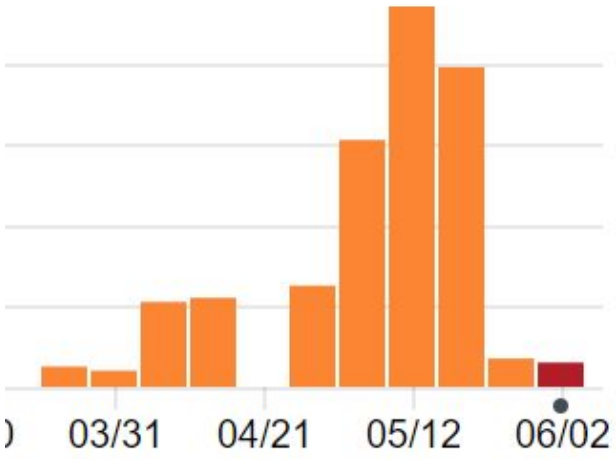
vis, men vi kommer fortfarande ha kvar att man ska läsa och förstå den nya koden då det har bidragit stort till vår inlärning under detta projektet. När vi inte längre behöver lägga fokus på att lära oss JavaScript och React från grunden kan vi istället lägga denna tid på att lära oss hantera automatiska tester.

> 5. the three KPIs you use for monitoring your progress and how you use them to improve your process

A:
Till en början hade vi inga uttalade KPIs förutom att en viss velocity skulle avklaras varje vecka. Under arbetets gång utvecklades andra mätbara mål, som exempelvis att vi ville öka antalet pull requests varje sprint för att underlätta de tasks som till viss del berodde av varandra.

Velocityn satte vi till 10 per person och sprint, vilket vi i stort sett höll fast under arbetets gång. Däremot anpassade vi effort på varje tasks för att uppnå 60 i velocity varje sprint. Därför fick varje task färre poäng mot slutet av projektet, för att vi skulle kunna genomföra mer arbete och slutföra projektet.

Grafen nedan visar hur antalet commits ökade under arbetets gång, fram till redovisningen den 27/5 (varje horisontellt streck är 20 commits). Vi upplevde också att gruppen under de senare veckorna arbetade mer kontinuerligt under veckans alla dagar och inte bara precis innan sprintens slut, men detta hittar vi ingen graf som visar på.



Även antalet pull requests ökade och då framför allt efter den handledning vi hade med Håkan där han poängterade hur viktigt detta var för oss. Under projektets första två sprintar hade vi 5 respektive 4 pull requests, medan den tredje sprinten hade 21 och den fjärde, slutliga, sprinten hade 17. Värt att poängtera är att vi hade en sprint

för att introducera oss alla till verktygen innan vi började med det “riktiga projektet”. Denna är inte med i statistiken.

B:
Om vi hade fortsatt arbeta med Hoi hade vi definitivt använt oss av mindre tasks och fler pull request, då vi snabbt insåg att detta verkligen underlättade arbetet! Vi fick inte bara mer inblick i de andras arbete utan även färre konflikter då de olika brancherna skulle mergas in i mastern. Allt som allt innebar det här att alla i gruppen får en större bild av hur alla komponenter i programmet fungerar, vilket i sin tur snabbar på vidareutveckling av nya komponenter.

Dessutom skulle vi ha en betydligt bättre uppfattning om hur stor effort som krävs för specifika tasks och därmed kunna tilldela en mer korrekt siffra till varje uppgift. Då hade det också blivit uppenbart att velocityn kunnat öka när gruppens medlemmar blev kunnigare och snabbare.

A->B:
De lärdomar vi fått med oss från kursen kommer definitivt hjälpa oss framöver, men vi tror också att det hade varit bra om vi läste på om olika KPI;er och använt dem på ett mer formellt sätt. Nu gick vi mer på känsla än att utgå från det faktiska konceptet.

Social Contract and Effort

> 6. your social contract, Link to an external site, i.e., the rules that define how you work together as a team, how it influenced your work, and how it evolved during the project (this means, of course, you should create one in the first week and continuously update it when the need arrives)

A:
Under första veckan skrev vi vår sociala kontrakt där det står alla regler, instruktioner och rekommendationer på gruppens arbetssätt och kooperation. Alla medlemmar har följt kontraktet noggrant och vi hade inget behov att ändra någonting i det.

Vi hade mötet minst en gång varje vecka som det var skrivit, men under de första 2-3 veckorna så träffades vi flera gånger. All kommunikation har genomförts genom Slack. Varje gruppmedlem har respekterat andra och ifall man var upptagen och inte kunde komma på mötet så meddelades det i god tid.

B:
I dagsläget har vi inget behov att ändra kontraktet och allt som står där följs noggrant. Om vi ska fortsätta med samma tempo som vi arbetar just nu så tänker vi inte att ändra någonting. Däremot om projektet kommer att utöka sig, så behöver vi ändra hela upplägget. Ifall projektet blir så pass stort och det blir mycket arbete att genomföra så måste vi anpassa kontraktet till nya utmaningar. Beroende på situationen kan vi kanske skriva att man behöver träffas minst 3-4 gånger i veckan

eller om man ska koda bara tillsammans eller om man ska använda ett annat kommunikationsmedel. Allt detta kommer att bero på de krav som vi ska ha framför oss.

A->B:
Att ändra social kontrakt är inte någon komplicerad uppgift. Vi behöver framförallt titta på de uppgifter vi ska ha och lösa, sedan hur mycket tid vi kan lägga på projektet. Utifrån det kan vi skriva det nya sociala kontraktet.

7. the time you have spent on the course and how it relates to what you delivered (so keep track of your hours so you can describe the current situation)

A:
Eftersom kursen tar upp halva vår heltidsutbildning bör vi lägga ungefär 20 h per person och per vecka på kursen. Vi lade 4h på måndagsmöte där följande genomfördes: review och retrospective för föregående sprint, samt uppdatering av sprint backlog för kommande sprint. Några veckor i början hade vi ett möte till i helgrupp på ytterligare 2h, men de flesta veckorna sågs vi bara 1 gång i helgrupp. Vi genomförde också en variant av daily scrum över Slack varje dag under veckan när någon hade arbetat med projektet. Resterande 16 h disponerade var och en för att genomföra sina tasks och reviewa andra gruppmedlemmars pull requests. Under kursens gång har vi lagt ungefär samma antal timmar alla veckor, men tiden kunde användas mer effektivt mot slutet då vi gemensamt satte upp målet att börja arbeta med våra tasks tidigare i veckan och började med mindre/kortare tasks. En del gruppmedlemmar hade problem med att man fastnade flera timmar i att lösa små detaljer, vilket i början gjorde oss mindre effektiva. Detta problem försvann längre fram när vi arbetade med projektet fler gånger i veckan till antalet, men kortare stunder. Så initialt disponerade nog de flesta av oss våra 16 h/vecka som två hela arbetsdagar, men mot slutet arbetade man med projektet en eller flera gånger per dag. Om man fastnar med ett problem kan man alltså lägga projektet åt sidan och fråga om hjälp så man själv kan lösa det nästa gång man ska arbeta.

B:
I denna projektgrupp fanns studenter med minst tre olika scheman, vilket gjorde det svårt att ses i helgrupp så ofta som vi hade behövt. Om vi i nästa projekt kan anta att man hade haft möjlighet att träffas oftare hade vi disponerat de 16 timmarna förutom måndagsmötet annorlunda för att kunna ta hjälp av varandra snabbare än att skriva över Slack. Till exempel hade vi resterande 4 dagar i veckan kunnat ha 4h workshops, där man kunde sitta gemensamt och arbeta med sina tasks för veckan. Detta hade också blivit smidigare då alla arbetar med projektet samtidigt.

A->B:
För att ta oss dit behöver vi förutom att få mer lika scheman också övertyga hela projektgruppen om att detta arbetssätt är bra för oss. I dagsläget trivs vissa bättre med att arbeta individuellt medan andra gärna arbetar i lag. Eftersom ett projekt som detta kräver lagarbete i ganska hög utsträckning behöver man som gruppmedlem anpassa sig, men samtidigt måste alla få möjlighet att arbeta på bästa sätt utifrån sina förutsättningar och preferenser. För att få en 4h workshop som denna effektiv hade man kanske kunnat börja med ett kort daily scrum där man presenterar vad man gjorde igår, vad man tänker göra idag och om man behöver hjälp med något. Kanske bör man även ha en struktur där man går ut ur rummet om man ska diskutera något längre för att inte störa varandra. En annan fördel med detta arbetssättet är att vi lägger all tid gemensamt, och den blir effektiv för att man resten av tiden måste arbeta med sin andra kurs som går parallellt med projektkursen. Denna press tror vi gör att man blir effektiv, planerar sitt arbete och arbetar smart.

Design decisions and product structure

> 8. how your design decisions (e.g., choice of APIs, architecture patterns, behaviour) support customer value

A:
Vi valde att bygga vår webbapplikation med React som frontend. Detta skapade kundvärde eftersom det var ett lätt ramverk att lära sig, trots att ingen i gruppen kunde det innan projektet, och endast en hade arbetat med webbutveckling. Det stora utbudet på färdiggjorda komponenter gjorde att vi snabbt kunde bygga ut vår applikations funktionalitet. Vi använde till exempel kalenderkomponenter från Airbnb, betalningskomponenter från Stripe, och färdiga komponenter för kartor, som byggde på OpenStreetMap’s api. Vi använde en klient och en server, men vi körde bara servern lokalt, så vi kunde inte testa programmet med flera klienter samtidigt.

B:
I framtida projekt ska vi se till att vi har en server åtkomlig för flera samtidigt, så att vi kan testa och utveckla med hjälp av den. Vi ska också strukturera vår kod mer. Detta kan ge mer värde till kunden eftersom det kan minska risken att vi behöver gå igenom storskalig refactoring som tar tid.

A->B:
För att ha en åtkomlig server skulle vi kunna använda en av många tjänster där man kan lägga upp en sådan. För att strukturera vår kod mer skulle vi kunna följa designmönster som MVC, MVVM eller MVP mer formellt. Vi skulle även kunna använda andra mönster som Observer-mönstret för att lättare skicka information mellan objekt.

9. which technical documentation you use and why (e.g. use cases, interaction diagrams, class diagrams, domain models or component diagrams, text documents)

A:
Eftersom vi utvecklade vår applikation i JavaScript hade vi inget behov att göra något klassdiagram. Däremot hade vi några skisser på produktens utseende (dvs hur grafiskt det kommer att se ut), samt Business Model Canvas. Det sistnämnda gjorde att vi alltid visste vilka våra kunder var och därmed kunde säkerställa att vi alltid levererade värde till dem. Vi gjorde också en skiss över domänmodellen i början av projektet.

B:
Om vi ska fortsätta att arbeta i JavaScript så behöver vi inte rita något klassdiagram, däremot kan Business Model Canvas behöva ändras beroende på åt vilket kundsegment vi ska vända oss. Det kan vara bra att ha en mer färdigställd domänmodell, som hålls uppdaterad genom projektets gång. Det finns också annan dokumentation att göra, som till exempel sekvensdiagram.

A->B:
Vi skulle kunna lägga mer tid på dokumentationen, och framför allt på att uppdatera den genom projektets gång.

10. how you use and update your documentation throughout the sprints

A:
Vi försökte dokumentera vår kod med kommentarer medan vi skrev den, men följde inte någon strikt standard för det, vilket gjorde att en del kod fick mindre kommentarer. Det vi skrev om dokumentation i det sociala kontraktet var att det alltid skulle delas med alla i gruppen, vilket vi följde.

B:
I ett framtida projekt borde vi se till att uppdatera vår dokumentation utanför koden, som domänmodell, sekvensdiagram och liknande.

A->B:
Vi skulle kunna införa uppdatering av dokumentation som en del av våra sprint reviews, och på så sätt se över vår dokumentation varje sprint.

11. how you ensure code quality and enforce coding standards

A:
Vårt enda konkreta krav på inlämnad kod har varit att den ska fungera, men gällande olika specifika situationer och funktioner (input bla.) i programmet har vi kommit överens om att skriva det uniformt så att det stämmer överens med liknande delar på andra ställen i koden.

Om vi skulle fått göra om programmet skulle vi från början bestämt oss för hur React komponenterna ska vara uppbyggda.

B:
Vi vill såklart att koden ska ha en gemensam struktur, att allt ska vara tydligt, samt det ska vara mer kommentarer i koden. Det beror på att om vi ska fortsätta skriva mycket kod utan att "städa" efter oss, blir det till slut svårt att förstå vad vi skrev tidigare samt att vi kommer lagra onödiga saker vilket kommer att påverka hastighet och minnet.

A->B:
Genom att sätta nya acceptanskriterier och bestämda standarder på koden som säga hur man ska skriva (konventioner för namn på variabler, funktioner, etc), vilken struktur man ska ha samt vad och hur man behöver kommentera.

Application of Scrum

12. the roles you have used within the team and their impact on your work

A:
Varje lagmedlem har fått möjligheten att ta del av alla roller inom Scrum. Som minst har varje enskild individ varit både produktägare samtidigt som man är en del av utvecklingsteamet. Under de första sprintarna tog enskilda personer rollen som Scrum master. Tanken från början var att låta rollen vara flyktig och låta varje gruppmedlem testa på rollen under en Sprint. Längre fram blev det dock inte så och vi delade istället rollen inom gruppen.

B:
I framtiden skulle vi vilja ha tydliga, distinkta roller inom teamet. En produktägare, en Scrum master, och fyra utvecklare. Möjligheten att rotera Scrum master rollen kan vara kvar till en början såvida inte rollen faller naturligt på en viss individ. Dock bör man få testa på rollen under en längre period. Dessutom behöver Scrum mastern först och främst vara just Scrum master och inte samtidigt ta del av de andra rollerna. Det var kanske just det som ledde till att rollen aldrig riktigt tog fart i vår grupp. Detsamma gäller rollen som produktägare. Vi vill ha en tydlig sådan roll.

A->B:
För att nå denna separation av roller behöver vi tydligt dokumentera vad varje roll innebär och ansvarar för. Dessutom måste man inte testa på de andra rollerna om man redan har en roll man är bekväm med. På så sätt behöver inte alla i gruppen vara med i Scrum master rotationen utan bara de som vill och är engagerade.

13. the agile practices you have used and their impact on your work

A:
Vi har använt oss av en Scrum board på Trello som delats upp i sju kolumner: *Epics*, *User stories*, *Tasks*, *Bugs*, *In progress*, *Testing*, och *Done*. Denna har ändrats allteftersom projektets gång, exempelvis la vi till *Bugs*-kolumnen mot slutet av projektet. Uppgifterna i kolumnerna är listade efter graden värdeskapande. Ju mer värde uppgiften genererar, ju högre upp i kolumnen hamnar den.

Vår användning av tasks har påverkat vårt arbetssätt väsentligt, speciellt under de sista sprintarna. Då insåg vi, tack vare handledaren, att våra tasks var för stora. Vi tog till oss feedbacken och började istället generera mindre tasks som mergeades in i master branchen allteftersom tasken blev färdig. Detta bidrog till ett bättre flyt i sprintarna. Man hade mer koll på vad andra medlemmar höll på med och vi slutade merga in allting i slutet av sprinten.

Istället för att ha en daily scrum har vi kört med en modifierad version där vi dedikerat en Slack-kanal för detta syfte. Om man ska börja jobba, håller på att jobba, eller har jobbat på någonting meddelar man detta till resten av gruppen i *dailyscrum* kanalen på Slack. Vi valde detta tillvägagångssätt eftersom det var svårt att samla hela gruppen för ett möte varje dag. Dessutom var det inte varje dag man jobbade på projektet och då hade man inget att berätta. Alla var heller inte i skolan varje dag och då kändes inte ett 10 minuters som skäl nog att ta sig till skolan för att sedan berätta att man inte hade gjort något.

Genomgående använde sig de som ville av parprogrammering. Under de första sprinterna matchade vi I-studenter med mer erfarna IT-studenter för att jämna ut kunskapen mellan samtliga gruppmedlemmar. Mot slutet av arbetet så fortsatte ibland parprogrammeringen, men inte i samma grupperingar.

B:
Vad gäller storleken på våra tasks kommer vi definitivt hålla oss till mindre, mer hanterbara, tasks som vi gjort mot slutet. Dessutom tar vi med oss INVEST och SMART kriterierna som stöd för utveckling av framtida user stories och tasks.

Något vi skulle göra annorlunda i nästa arbete är att applicera daily scrum som den borde. Eftersom vi inte har något kontor att gå till eller ett schema som alla håller sig

till har det varit svårt att applicera riktig daily scrum. Men har vi möjligheten att sitta på kontor och jobba 9-17 varje dag hade daily scrum möten varit att föredra.

En till sak vi kan ha med oss till nästa projekt är att ha en burndown chart och sätta upp andra tydliga KPI:er.

Parprogrammeringen skulle också med fördel kunna nyttjas på en bredare front. Detta hade även underlättat vår problematik med att endast en gruppmedlem förstod sig på servern (se punkt 15).

A->B:
Vi kan sätta upp INVEST och SMART kriterierna vid user stories och tasks kolumnen i vår Scrum board så att de är nära till hands. På så sätt ser vi bättre till att dessa riktlinjer följs.

För att ha kontinuerlig daily scrum behöver vi synka våra scheman och om möjligheten finns även boka tid i ett rum där vi kan ses kontinuerligt.

Vi ska dedikera ett eller flera möten till att diskutera och sätta upp fler tydliga KPI;er.

> 14. the sprint review and how it relates to your scope and customer value (Did you have a PO, if yes, who?, if no, how did you carry out the review? Did the review result in a re-prioritisation of user stories? How did the reviews relate to your DoD? Did the feedback change your way of working?)

A:
Efter varje sprint har vi haft en sprint review samt sprint retrospective. Sprint reviewn utfördes genom att någon i laget öppnade mötet och vi gick laget runt så varje person fick berätta vad man hade jobbat med och lite om hur det funkade för att ge resterande lagmedlemmar en inblick. Direkt efter sprint reviewn har vi haft en sprint retrospective där vi även reflekterat kring hur sprinten har gått.

Som sagt har vi inte haft *en* produktägare utan vi har gemensamt agerat produktägare. När tvister har uppkommit har vi löst det genom en omröstning. Allteftersom projektets gång har vi omprioriterat vissa grejer och flyttat de högre upp i backloggens kolumner då vi ansett att vissa funktioner levererar mer kundvärde.

B:
I nästa projekt ska vi fortsätta med sprint review och retrospective. Vi ska även ha med oss tankegången om vad som genererar mest kundvärde och försöka få ut det så fort som möjligt.

Vi vill även ha en produktägare som vi kan ha nära kontakt med genom hela projektet.

A->B:
Vi ska helst ha produktägaren nära till hands, t.ex. ska denne sitta på samma kontor. Dessutom ska vi ha en dedikerad Slack-kanal där diskussioner med produktägare kan ta plats. Om vi fortfarande inte löser att få en fysisk person som produktägare kan vi ändå sätta upp en Slack-kanal. Vi har i dagsläget en Slack-kanal för daily-scrum, vilket gör att man ändrar mindset när man deltar i diskussioner där. Om samma effekt kan fås av en framtida PO-kanal kanske vi kan få en kontinuitet även i dessa diskussioner och undvika att saker kommer upp först vid sprint review.

15. best practices for learning and using new tools and technologies (IDEs, version control, scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them)

A:
För de flesta i gruppen var många, om inte alla, verktyg vi använde oss av helt nya. Här kommer en förklaring till vilka de var och hur vi lärde oss använda dem:

Trello - trial and error. Vi skapade ett scrum board och provade oss fram till dess att vi förstod de olika funktionerna.

Visual Studio Code - Några i gruppen hade använt sig av VS Code tidigare, medan de andra hade använt liknande programvara. De “nya” användarna kunde ställa frågor till de med erfarenhet vilket gjorde att alla snabbt blev bekväma med mjukvaran.

Git - även här hade några i gruppen kunskaper om verktyget, vilka de delade med sig till resterande gruppmedlemmar av. Alla i gruppen kunde skapa nya grenar, pusha, göra pull requests och commits.

React, JavaScript, CSS, HTML, etc - ett par gruppmedlemmar var mer bekväma med dessa språk och komponenter redan i startskedet och då var parprogrammering väldigt viktigt för att få med övriga gruppmedlemmar. Dessutom kan man hitta otroligt mycket information på Google!

Servern - här gjorde vi ett misstag och endast en gruppmedlem förstod och kunde hantera servern. Han lärde sig genom att läsa på om det i dokumentation om Node js, Express och serverkommunikation.

B:
Fortsättningsvis kommer vi se till att ingen kunskap ligger hos endast en gruppmedlem, vilket servern gjorde för oss under arbetet med Hoi. De andra verktygen kunde samtliga i gruppen hantera (även om kunskapsnivån skiftade), vilket gjorde att arbetet kunde delas upp mellan oss och om någon inte hann klart

kunde en annan gruppmedlem ta vid där den första slutade. Detta är något som hade varit önskvärt även med arbetet med servern, varför vi måste se till att all kunskap sprids inom gruppen framöver.

A->B:
Att ha fler fysiska möten inom gruppen där den personen som sitter på kunskap kan dela med sig av denna till andra skulle underlätta lärandeprocessen. Dessutom kommer vi alla att fortsätta söka information på internet samt fråga andra som besitter kunskap för att på så vis lära oss mer om verktygen, processerna och software engineering i stort.

16. relation to literature and guest lectures (how do your reflections relate to what others have to say?)

A:
Vad gäller kurslitteraturen har vi inte läst de rekommenderade böckerna, men vi har sökt oss fram till mycket information på internet. Främst har vi använt oss av hjälp-artiklar och annan dokumentation om det vi arbetade med, men även YouTube-videos har använts. Några av oss har läst hamburgar-artikeln samt andra artiklar online för att få en bättre förståelse för dels Scrum men även andra teknologier som vi använde under projektets gång.

B:
Inför nästa projekt vill vi ha läst den rekommenderade kurslitteraturen. Inte pga brist på liknande information online, men pga av att dessa rekommendationer kommer från mer erfarna utvecklare och då bör vi ta del av tipsen och nyttja de mer erfarnas kunskap till fullo. Vi kommer med största säkerhet behöva komplettera kurslitteraturen med annan litteratur nästa gång också, eftersom vårt projekt antagligen inkluderar språk eller funktioner som inte är gemensamma för samtliga studenter i kursen.

A->B:
Vi ska ta del av tipsen från mer erfarna utvecklare då detta oftast är en bra kunskapskälla. Vi ska även själva aktivt söka hjälp i form av böcker, gästföreläsningar, samt mer erfarna kollegor. Förutom det kommer vi också fortsätta använda vanliga sökmotorer för att hitta information om tekniska detaljer och om hur man kan lösa problem. Det finns mycket lättillgänglig information och ofta även på olika kunskapsnivåer, vilket gör att man med lite envishet kan lösa de flesta problem genom att läsa hur andra har gjort på forum och dylikt.