supportMe

Grupp 30

Projektplan

V. 0.1.3

250321

Dokumenthistorik

Datum	Version	Beskrivning	Författare
210309 090321	0.1	Milstolpar för första sprint	Gabriel Modin
210309 090321	0.1.1	Produktbeskrivning första utkast	Nicholas Narvell
210311 110321	0.1.2	Lagt till *Syfte* och sidhuvud	Marcus Juninger
210315 150321	0.1.3	Påbörjat *Ordlista* samt *Utvecklingsprocess*	Marcus Juninger
210316 160321	0.1.4	Milstolpar första version	Marcus Juninger, Gabriel Modin Bärzén, Nicholas Narvell, Isak Holmqvist
210316 160321	0.1.5	Riskanalys första utkast	Isak Holmqvist
210317 170321	0.1.6	Utvecklingsprocess första utkast klart (förutom testning)	Marcus Juninger
210317 170321	0.1.7	Första version projektöversikt	Gabriel Modin Bärzén
210317 170321	0.1.8	*Utvecklingsprocess:* Lagt till *Testning,* kompletterat kopplingar till fördjupningar, ändringar efter gruppfeedback. Avsnitt klart.	Marcus Juninger
210318 180321	0.1.9	Lagt till Intressenter	Nicholas Narvell
210318 180321	0.1.10	Lagt till övergripande information om planering och timbudget	Marcus Juninger
210319 190321	0.1.11	Lagt till riskanalysens syfte	Isak Holmqvist
210325 250321	0.1.2	Riskdiagram och riskanalys uppdaterade	Isak Holmqvist
250321	v.1.0	Uppdaterat referens [1], småfix med struktur, sparat ner version 1.0 inför RS-möte 1.	Marcus Juninger

Innehåll

Projektplan

Syfte

Dokumentet används som ett övergripande samlingsdokumentet och fastslår projektets arbetssätt och definierar de risker samt milstolpar som identifierats. Utöver detta presenteras även projektets syfte, omfattning och mål, följt av avsnitt som behandlar intressenter och produktbeskrivning. För ökad förståelse kring eventuella otydligheter i dokumentet inkluderas också en ord- samt referenslista.

Dokumentet används som underlag för att kontinuerligt och iterativt kunna genomföra och strukturera arbetet på ett genomgående effektivt sätt.

Ordlista

Card(s) - Ett steg av en Guide, innehåller alltid text, kan även innehålla bild eller video.

Guide(r) - En samling av Cards.

Author(s) - En användare som kan skapa eller visa Cards och Guider, samt spara dessa lokalt.

End-User(s) - En användare som kan visa Guider och spara dessa lokalt.

Referenser

[1] I. Sommerville, *Software Engineering, Global Edition,* 10th ed. London, England: Pearson Education Limited. 2016.

Översikt av projekt

Syfte

Det huvudsakliga syfte projektet har för gruppen är att få en bättre förståelse för den övergripande processen för mjukvaruutveckling. Hur tar man de teoretiska delarna och applicerar dem för att kunna skapa en så färdig produkt som möjligt med den kunskap vi har.

I projektansökan uttrycker gruppen en gemensam vilja att i första hand förbättra sina kunskaper och kompetenser inom mjukvaruutveckling. Samtliga medlemmar uttryckte att de i första hand på något vis ville förbättra sina kunskaper inom programmering, objektorientering och projektarbeten. Somliga medlemmar uttryckte, till en mindre grad, viljan att förbättra sina kunskaper inom databashantering samt webbutveckling. Gruppen strävar mot att bygga en fungerande produkt vilket innebär att kunskaper kring projektorganisering behöver finslipas för att kunna bygga upp en struktur som kan stödja gruppens uppsatta mål.

En annan förekommande del i gruppmedlemmarnas projektansökningar var viljan att förbättra sina färdigheter inom grupparbeten för att förbereda inför projektarbeten i framtida jobb. Att känna på hur det är att arbeta med lite mer planerade processer och arkitekturer.

Omfattning

Gruppmedlemmarna delade från början på tanken att skapa programmet som en mobilapplikation samt tillhörande hemsida. När det diskuterades bestämde sig gruppen gemensamt för att erfarenheten eller kunskapen inte finns bygga ett sådant system. När gruppen undersökte frågan mer noggrant så bestämdes det att det inte fanns tid att utbilda medlemmarna inom webb- eller mobil-utveckling.

Detta innebar att beslut togs om att begränsa projektet till utveckling i Java. Gruppen beslutade att skriva programmet med hjälp av ramverket JavaFX då gui-designen är enklare än javas motsvarande inbyggda ramverk Swing. Användningen av javaFX innebär även att koden håller bättre struktur och leder till en slutprodukt av högre kvalitet.

Gruppen diskuterade även möjligheten att utveckla produkten för att möjliggöra för företag att skapa interna guider för processer som är unika till dem. Detta moment beslutades för att vara större än vad som rimligtvis går att genomföra i projektet och kan komma att diskuteras igen ifall tiden finns.

Mål

Den ambition de flesta av gruppmedlemmarna delar på i projektet är att kunna bygga en så pass färdig produkt att den anses brukbar för privat bruk. Detta trots att det huvudsakliga syftet fokuserar på fördjupning av kunskaper.

Intressenter

Intressenter innefattar alla som på något sätt är inblandade projektet eller kan dra nytta av slutprodukten. Dessa delas upp i tre olika grupper, interna, externa och slutanvändare.

Interna

Isak Holmqvist - utvecklare, ansvarig för användbarhetstestning
Gabriel Modin Bärzén - utvecklare, ansvarig för kodgranskning
Marcus Juninger - utvecklare, ansvarig för användbarhetsanalys
Ahmad Toron - utvecklare, ansvarig för användbarhetsanalys
Nicholas Narvell - utvecklare, ansvarig för dokumentgranskning

Mats Syde - handledare

Externa

Mau - Malmö universitet
Retrospektgrupp - studentgrupper från Malmö Universitet som deltar i granskningsarbete

Slutanvändare

Guide-författare - Användare som skapar guider
Guide-konsument - Användare som läser guider

Produktbeskrivning

Projektet är baserat runt en idé om teknisk support för dina nära och kära. Programmet ska fungera som en central för användare som inte är tekniskt lagda och kan behöva hjälp med sin teknik hemma. Den andra typen av användare är de som är tekniskt lagda som kan skapa och dela skräddarsydda felsökningsguider. Programmet kommer att ha två olika vyer, en för att skapa guider och en för att visa guiderna på ett enkelt och tydligt sätt. Målgruppen för vårt program är personer som ofta får hjälpa personer i sin omgivning med teknisk support och ger de en möjlighet att skapa tydliga guider.

Utvecklingsprocess

Övergripande process

Projektet avses drivas med agila utvecklingsmetoder där såväl dokumentation som produkt bearbetas inkrementellt och iterativt. Den övergripande processen influeras huvudsakligen av två väletablerade modeller inom projektarbete och mjukvaruutveckling; Scrum samt XP (Extreme Programming) [1, s. 77], [1, s. 84-87]. Projektets storlek innebär att ingen av dessa modeller kan anses som lämplig i sin renodlade form, därför används de till stor del som inspirationskällor och de mest relevanta elementen abstraheras för detta projekt. Delar av Scrum används främst för projektarbetet i form av mötestyper och struktur för planering, samtidigt som delar av XP rent generellt används mer i de olika utvecklingsfaserna. Projektet genomsyras av kollektivt ägande i samtliga delar, där alla gruppmedlemmar är gemensamt ansvariga och uppmuntras bidra med förslag och lösningar där det anses passande.

Nedan beskrivs mer detaljerat hur Scrum och XP implementeras inom projektets olika arbetsområden.

Mötesformer och planering

Projektet är på förhand indelat i fyra sprintar som är tre veckor långa, undantaget sprint fyra som är en vecka längre. Varje sprint inleds med ett större planeringsmöte för att fastställa en sprint backlog i form av milstolpar med leverabler som gruppen strävar efter att uppnå under de närmsta veckorna. Utöver detta hålls även måndagsmöten för att ytterligare bryta ned milstolpar och leverabler i mer konkreta arbetsuppgifter en vecka åt gången, samt fredagsmöten för uppföljning och planering inför kommande vecka. Övriga dagar hålls ett kort Daily Scrum för att säkerställa att arbetet flyter på för samtliga medlemmar och alla är medvetna om vad de bör arbeta med under dagen.

Vid slutet av varje sprint genomförs ett obligatoriskt retrospektmöte där projektgruppen presenterar sitt nuvarande arbete. Inför detta hålls en Sprint Review där gruppen diskuterar det arbete som gjorts och inte gjorts de senaste veckorna, samt övriga tankar eller problem som uppkommit. Här granskas även arbetade timmar för att säkerställa att angiven timbudget hålls. På tisdagar hålls ett kortare handledningsmöte de veckor då retrospektmöten ej genomförs.

Kravhantering

Den första kravupptäckten genomförs med hjälp av brainstorming och produktplanering med hela projektgruppen. Detta ligger till grund för att lyfta fram de mest centrala och grundläggande kraven som behöver implementeras för att produkten ska anses vara fullständig. Efter detta används en processmodell likt XP där krav hanteras inkrementellt och förändring välkomnas. Flera krav kan uppkomma genom user-stories eller scenarion för att tydligare specificera betydelsefull funktionalitet. Eventuellt kan även prototyper komma att brukas för att få feedback på design eller specifika funktioner, vilket i sin tur kan ge upphov till nya eller förändrade krav.

Kraven prioriteras efter MoSCoW-modellen, vilken ytterligare specificeras i kompletterande kravdokument som tillhör projektets kravspecifikation.

Design och arkitektur

Produktens utseende bestäms gemensamt av projektgruppen genom kontinuerliga diskussioner och kommer att förändras med tidens gång. Generellt följs tankesättet från XP där en simpel men effektiv design eftersträvas. Eventuellt kommer prototyper användas för att samla in ytterligare extern feedback angående designen från produktens förväntade målgrupp.
De två primära arkitekturmönster som kommer användas är *klient-server-* samt *data-centrerad arkitektur*. Dessa används som övergripande mönster för produktens uppbyggnad och funktionalitet i sin helhet, sedan används sekundära arkitekturmönster för att strukturera den kod som skrivs. För klientsidan används exempelvis en semi-strikt version av MVC.
Produktens design och arkitektur presenteras mer detaljerat i projektets designdokument.

Utveckling och implementation

För både utveckling och implementation hämtas inspiration från främst XP. Kodning kommer ofta kombineras med parprogrammering för att snabbt upptäcka eventuella problem och säkerställa en högre genomgående kvalitet på koden. Detta medför samtidigt att gruppen kontinuerligt arbetar med refactoring och genomför detta direkt när möjligheten upptäcks.

Produktens kommer att byggas inkrementellt och innebära flertalet mindre releaser som iterativt förbättras genom feedback från gruppmedlemmar eller externa intressenter. För att behålla en hög standard på det som produceras kommer gruppen använda sig av både peer-reviews och walkthroughs. I de individuella fördjupningarna kring dokument- samt kodgranskning finns mer information om processen för dessa.

Testning

Kravbaserad testning kommer utgöra majoriteten av projektets iterativa testprocess. Löpande unit-testing planeras att användas för att testa specifika delfunktioner inom systemet. För varje slutfört inkrement kommer även integrationstester genomföras, oftast vid sprintarnas slut, för att bekräfta att de separata enheterna fungerar i symbios med varandra enligt planerad arkitektur. Dessa två testmetoder används främst för att verifiera och validera de funktionella kraven.

I slutet av sprint tre och fyra planerar projektgruppen att genomföra större systemtester enligt skriptade och explorativa metoder, samt regressionstester som kommer behövas. Detta görs främst för att verifiera de icke-funktionella kraven och bekräfta att eventuella åtgärder löst tidigare problem.

Samtliga gruppmedlemmar kommer att vara delaktiga i någon typ av testning. Samtliga tester genomförs av minst en person som inte varit delaktig i att utveckla det aktuella stycket kod. Mer information om projektets testprocesser finns i *Testdokument – Kravbaserad Testning* samt i den individuella fördjupningen kring användbarhetstester.

Planering

Projektet pågår under totalt 12 veckor och är uppdelat i fyra sprintar. Den totala timbudgeten består av 220 timmar per gruppmedlem. Nedan finns en tabell som innehåller en ungefärlig uppskattning om antalet timmar varje person bör ha arbetat per vecka och sprint. Mängden arbete som ska utföras inom varje sprint definieras i form av milstolpar med tillhörande leverabler. Hur mycket tid som krävs för att uppfylla de olika milstolparna kommer att variera och därför förväntas de faktiska tidsrapporterna innehålla cirka 10% fler eller färre timmar än planerat per sprint.

Sista deadline för projektet sker i slutet av vecka 22, därför är denna vecka fullständigt reserverad för oförutsedda händelser och eventuell problemlösning. De enskilda gruppmedlemmarna disponerar i största mån själva hur de vill fördela sin arbetstid, inom ramarna för att den övergripande projektplaneringen inte blir lidande. Detta innebär bland annat att individerna även bestämmer hur de väljer att lägga upp sitt projektarbete i relation till externa faktorer som eventuella helgdagar och andra kurser.

Timbudget per vecka och sprint

Vecka	Per gruppmedlem	Totalt för grupp
10	18	90
11	36	180
12	54	270
13	72	360
14	90	450
15	108	540
16	126	630
17	144	720
18	162	810
19	180	900
20	198	990
21	216	1080

Milstolpar

Sprint 1:

Milstolpe 1, Visning av guide-kort, vecka 12, 26/3

Vi skapar en körbar minsta version av klientdelen med ett dumt GUI. Ingen kommunikation med server implementerad.

Leverabler:

- Uppvisning av ett guide-kort.
- Första gui iteration för visning av guide-kort.

Milstolpe 2, Första kravspec, vecka 12, 22/3

En första fullständig kravspecifikation klar.

Leverabler:

- Kravspecifikation V.1.0 sparad på Drive + GitHub.

Milstolpe 3, Projektplan v1.0, vecka 12, 24/3

Första version av projektplan klar.

Leverabler:

- Projektplan V.1.0 sparad på Drive + GitHub.

Sprint 2:

Milstolpe 4 Kravbaserad testning inkl testrapport, vecka 14, 7/4

Leverabler:

- Minst en testrapport skapad

Milstolpe 5 Designdokument v1.0, vecka 15, 14/4

Första version av designdokument ska vara klara.

Leverabler:

- Designdokument V.1.0 sparad på Drive + GitHub.
- Use case-beskrivningar/scenarion färdigställda.

Milstolpe 6 Klient & Server v0.5, vecka 15, 14/4

Fungerande klientdel med motsvarande tester.

Leverabler:

- Fungerande GUI.
- Enkel kommunikation mellan klient och server.

Milstolpe 7 Individuell fördjupning, vecka 15, 15/4

Leverabler:

- Metoder och anpassningar för användbarhetstest ska finnas beskrivna.
- Första kodgranskning är genomförd och dokumenterad.
- Första dokumentgranskning är genomförd och dokumenterad.

Sprint 3:

Milstolpe 8 Klient & Server prerelease v1.0, vecka 17, 23/4

Första versionen av både klient och server färdigskrivna i kod. Kommunikation mellan dessa ska fungera och en databas för hantering av användare ska finnas.

Leverabler:

- Testbar färdig kod för första release.
- Databas för användarhantering implementerad.

Milstolpe 9 Test för release av v1.0, vecka 17, 30/4

Kravbaserad testning av prerelease v1.0.

Leverabler:

- Testfall med acceptabel felmarginal.
- Testrapporter sammanställda av utförda tester.

Milstolpe 10 Release 1.0, vecka 18, 6/5

Första releasen som inkluderar alla must-krav och de should-krav som har hunnits med. Leverabler:

- Upptäckta problem vid milstolpe 9 åtgärdade.
- Program paketerade för användning.

Sprint 4:

Milstolpe 12 Regressionstester ,vecka 19, 14/5

En sista testningsfas i syfte att upptäcka problem för funktioner i konflikt.

Leverabler:

- Test-rapport.
- Översiktlig plan inför sista release.

Milstolpe 13 Färdig produkt,vecka 22, 6/6

Leverabler:

- Slutgiltig produkt färdigställd.
- Majoriteten av “should-krav” ska vara implementerade.

Riskanalys

Syftet med denna riskanalys är att identifiera generella risker som kan ske under projekttiden. Dessa risker har en kort förklarande text och rangordnas senare ur ett sannolikhets- och konsekvensperspektiv. Denna rangordning går från 1-10 där 1 är lägst konsekvens eller sannolikhet och 10 är högst. Längst ner i denna del finns även ett spridningsdiagram som skapats av att man har satt in värdena på sannolikhets- och konsekvensgraden. Syftet med det diagrammet är att man enklare ska kunna se vilka risker man behöver lägga mer fokus på. Man kan dela in diagrammet i kvadranter, där rutorna som bildas står för vilka risker som behöver adresseras. De med hög sannolikhet och konsekvens är de risker som behöver adresseras först.

Identifierade risker

Projektrisker

PJ:1 Ökade kostnader i form av tid på grund av frånvaro av teammedlem

Om någon i teamet är frånvarande pga uppbokad aktivitet på max en dag

Sannolikhet: 5

Konsekvens: 3

Handlingsplan: Om någon skulle bli sjuk eller annan frånvaro sker så kan den personen alltid ta ikapp det innan deadline även om det innebär längre tid att sitta med projektet utanför de planerade timmarna arbetet var tänkt att ta innan deadline.

PJ:2 Verktygsinlärning

Denna risk är kring om teamet eller någon i teamet måste lära sig ett nytt verktyg

Sannolikhet: 3

Konsekvens: 6

Handlingsplan: Tidsallokering och förberedelse för vilka verktyg vi behöver lära oss är ett bra sätt att minska risken. Om detta trots allt sker kan tiden ses över direkt så man ser vad som behövs läggas tid åt från början.

PJ:3 Ökade kostnader i form av tid på grund av sjukdom hos teammedlem

Om någon i teamet är frånvarande pga sjukdom i mer än 2 dagar där man under denna tiden har svårt att arbeta.

Sannolikhet: 3

Konsekvens: 6

Handlingsplan: Att minska denna risken anses vara svår, man vet aldrig när man kommer bli sjuk. Om detta skulle inträffa får teamet allokera timmar för att gemensamt hjälpa den sjuke medlemmen.

PJ:4 Ökade kostnader i form av tid på grund av teammedlem hoppar av utbildning

Om någon i teamet eventuellt skulle hoppa av utbildningen under tiden som vi arbetar med projektet.

Sannolikhet: 1

Konsekvens 9

Handlingsplan: Det finns inte något sätt att förhindra att detta sker. Tidsallokering och tidsuppskattningar blir ogiltiga. Teamet måste göra nya uppskattningar och allokera tiden på de återstående medlemmarna vilket resulterar i mer timmar för att få en färdig produkt.

Produktrisker

PD:1 Kravtolkning

Denna risk är kring om och hur användbarheten kan försämras på grund av olik-tolkade krav hos produkten.

Sannolikhet: 6

Konsekvens: 9

Handlingsplan: Tydligare krav tidigt i projektet kan göra det enklare att tolka och implementera kraven och rätt sätt så det blir tydligare användbarhet. Om risken inträffar ska man direkt gå igenom kraven med intressenterna och diskutera återigen vilka funktionella och kvalitativa krav intressenten vill ha, och förklara vad som menas med dessa kraven.

PD:2 Design-risk

Denna risk är kring om och hur användbarheten kan försämras på grund av dålig design av produkten.

Sannolikhet: 3

Konsekvens: 9

Handlingsplan: Tydligare designdokumentation och uml-diagram före utveckling av produkt kan minska risken för att produkten har dålig design som i sin tur påverkar användbarheten för slutanvändarna. Om denna risk inträffar så ska man direkt sätta sig och formulera nya förbättrade designdokument och se över koden och se vad man kan ändra så att det blir liknande de förbättrade dokumenten.

PD:3 Arkitektur

Denna risk är kring hur en dåligt valt arkitektur påverkar prestandan på produkten

Sannolikhet: 3

Konsekvens: 7

Handlingsplan. För att undvika denna risk måste det aktivt tänka på vilken arkitektur som ska väljas och vad detta har för konsekvenser innan början på implementationen sker. Vid val av dålig arkitektur kan man få refactora koden till den nya arkitekturen. Det kostar tid men förbättrar prestandan i det långa loppet.

Riskdiagram

Här finns en risktabell kopplat till ett riskdiagram, för att ge en snabb överblick av risker riskernas sannolikhet och konsekvens och vilka risker man behöver prioritera att åtgärda och minska risken av att de sker.

	Sannolikhet	Konsekvens
PD:1	6	9
PD:2	3	9
PD:3	3	7
PJ:1	5	3
PJ:2	3	6
PJ:3	3	6
PJ:4	1	9