Paca

Grupp 5

Kravdokument

V. 1.5

22 mars 2018

Dokumenthistorik

Datum	Version	Beskrivning	Författare
180320	1.1	Ändring av förstasidan samt sidhuvud, Syfte, Teori (MoSCoW + namngivnings-standard) samt beskrivning av FK och IFK under respektive huvudkategori.	Milo Bengtsson
180320	1.2	Produktbeskrivning, Målgrupp, Intressenter samt påbörjat Funktionella krav	Hanna Fellwing
180320	1.3	Fortsatt arbete med Funktionella krav F-1 - F25	Hanna Fellwing och Milo Bengtsson
180322	1.4	Strukturerade om i rubriknivåerna, och gav varje rubrik en mörkt gråfärgad ordningsposition (t.ex. 3.1 och dess första underrubrik 3.1.1) för enklare förståelse och översikt av rubrikstrukturen. Lade till ord i ordlistan.	Milo Bengtsson
180322	1.5	Rättat stavning och formuleringar i alla F krav samt systemkrav. Skrivit användbarhetskrav 1-4, begränsningar 1-4 och ordlista.	Hanna Fellwing

Innehåll

1. Inledning

1.1 Syfte

Syftet med kravspecifikationen är att definiera och beskriva vad produkten ska göra såväl som vad det inte förväntas att göra. Detta tillåter en noggrann bedömning av krav innan design påbörjas, reducerar behov av omdesign samt förhindrar programvarufel. Kraven delas in i funktionella respektive icke-funktionella krav, för att därefter specificeras och kategoriseras ytterligare. Vidare är de formulerade utifrån produktens målgrupp och produktbeskrivning. Detta dokument innefattar även vilka som anses vara intressenter i projektet.

1.2 MoSCoW-modellen

MoSCoW är en modell för prioritering av funktioner och krav som kan användas vid mjukvaruutveckling [1].

1.2.1 Must

"*Användaren **måste** kunna logga in med ett användarnamn och ett lösenord*"

Krav som *måste* uppfyllas i webbapplikationen för att den ska fungera över huvud taget, det vill säga de viktigaste och mest kritiska funktionerna.

1.2.2 Should

"*Användaren **bör** kunna få mer information om ett pass genom att klicka på det*"

Should-kraven gäller funktioner som högst sannolikt kommer implementeras, men som inte nödvändigtvis är kritiska för att mjukvaran ska fungera.

1.2.3 Could

"*Användaren **skulle** kunna prenumerera på sin jobbkalender i personliga kalender*"

Funktioner som varken är nödvändiga eller kritiska, men som skulle vara "bra att ha" ("*nice to have"*), kategoriseras som *Could*-krav. Ofta representerar dessa förbättringar av

användbarhet eller funktioner som inte är nödvändiga för att produktrelease. Det är först och främst här som nedskärningar sker när åtgärder behöver tas vid tidsbrist.

1.2.4 Won't

"*Användaren **ska inte** kunna skapa aviseringar för sina pass i Paca*"

Funktioner som utesluts från applikationen – åtminstone tillfälligt – kategoriseras som *Won't*-krav. Istället för att radera dessa krav från dokumentationen, är de viktiga att behålla eftersom tankegången bakom beslutet är relevant. Vidare förhindrar detta att samma *Won't*-funktion föreslås igen på grund av att det inte är dokumenterat någonstans.

1.3 Standard för namngivning av krav

Processen genom vilken produktens krav definieras har sin grund i diskussioner kring *user stories* [2], där enklare påståenden såsom "Användaren kan logga in med ett användarnamn och ett lösenord" som så småningom översätts till mer specificerade krav i enlighet med namngivningsstandarden nedan. Förkortningarna skrivs i versaler och separeras med punkter, med undantag för fjärde och sista positionen i namnet som ska föregås av ett bindestreck.

Position		**Beskrivning**	**Förkortning i kravnamnet**
1	Kravtyp nivå 1	Funktionella krav	F
		Icke-funktionella krav	IF
2	Kravtyp nivå 2	F: Användarkrav	AK
		F: Systemkrav	SK
		IF: Användbarhet	A
		IF: Begränsningar	B
3	MoSCoW	Must	M
		Should	S
		Could	C
		Won't	W
4	ID	Unikt ID-nummer	1, 2, 3, osv. (varje nytt krav får nästa tal i ordningen)

Ett exempel på detta är F.AK.M-1: F står för funktionellt krav, AK för användarkrav, M för *Must* och 2 är kravets unika ID-nummer.

1.5 Ordlista

Ord	**Förklaring**
användargränssnitt	Ibland förkortat GUI (eng. *graphical user interface*). Är en metod för att underlätta interaktionen mellan människa och dator, vanligtvis mellan grafiska metaforer och bilder.
användarkrav	*Eng. user requirements.* Är krav som kommer från användaren.
boolesk (*adj.*)	Har antingen värdet sant eller falskt
dashboard	Visar övergripande och väsentlig funktionalitet
funktionella krav	Krav som beskriver hur systemet interagerar med användaren
default	Förinställt eller förvalt värde
intressent	Alla delaktiga och berörda parter
kvalitativa (icke-funktionella) krav	Krav som specificerar kriterier som kan användas för att bedöma ett system på en skala
mikrointeraktioner	Funktionalitet som endast gör en sak. En liten isolerad händelse med ett specifikt mål.
målgrupp	De privatpersoner, företag eller organisationer som tros ha störst behov av produkten
schemaläggare	En person med administratörsrättigheter i webbapplikationen som ansvarar för den dagliga aktiviteten, såsom att boka anställda och godkänna pass.
slutanvändare	En person som använder en färdig vara eller tjänst
supervisor	Ansvarig person för inköp och vid onboarding av webbapplikationen.
systemkrav	Nödvändiga förutsättningar i mjukvaran för att användarkrav ska kunna uppfyllas

user story	Enklare påstående kring funktionalitet, t.ex. “Användaren kan logga in med ett användarnamn och ett lösenord”.
UX	*Eng. User experience*. Användarupplevelse.
white-box testing	*Eng.* Testning av kod genom exekvering

1.6 Referenser

[1] S. Eklund, *Arbeta i projekt*. Lund: Studentlitteratur, 2010.

[2] Grupp 5, “s1_v2”, *Paca*. GitHub Repository, 2018. Tillgänglig: https://github.com/projectpaca/paca/tree/master/protokoll/Veckom%C3%B6ten/s1_v2 [Hämtad: 23-03-2018]

[3] F. Tsui och O. Karam, *Essentials of Software Engineering*. Sudbury, Mass: Jones and Bartlett, 2014.

2. Produktbeskrivning

Webbapplikationen Paca är tänkt att vara en schemaläggningsapplikation för företag där anställda överskådligt och enkelt ska kunna se sina arbetspass, boka lediga pass samt lägga in om de kan jobba utöver sitt schema. Paca kommer vara en desktop-applikation.

I applikationen loggar användaren in med användarnamn och lösenord. Förstasidan kommer vara en *dashboard* där användaren ska se sina nästkommande bokade pass, tillgängliga arbetspass och övrig information och nyheter som till exempel numret till receptionen för att sjukanmäla och liknande. Användaren kan även gå in i kalendern för att se sina pass, såväl i veckovy som dag och månad, och lägga in tider då hen är tillgänglig och villig att jobba. Vill användaren byta ett bokat pass görs detta genom att anmäla passet som inte längre tillgänglig. Det passet som användaren vill byta bort finns synligt för övriga anställdas dashboard och schema. Schemaläggaren eller den som ansvarar för schemat och tider får även en notis om att ett pass finns ute för byte och när ett passbyte gått igenom. Unikt med Paca är även att passbyte kan se mellan kollegor utan att den schema ansvariga involveras, med undantag för de tillfälle då alla krav för passet inte uppfylls av den som vill ta över passet.

Vilka kunskaper och färdigheter som en anställd besitter delas in i kategorier/avdelningar och subkategorier av den anställda under dennes profil. Exempelvis om Paca används för anställda i en dagligvaruhandel bör avdelningarna “Kassa”, “Chark”, “Lager”, “Golv”, “Frukt” och “Bröd” finnas. Och subkategorier kan vara “ATG-behörig”, “Beställare”, “Fronta”, “Egen larmkod” m.m. I användarens profilsida finns även kontaktuppgifter som namn och telefonnummer men också närmast anhörig och adress. Dessa uppgifter kan användaren själv redigera och uppdatera allt om det skulle ske några ändringar.

Den anställda kan även i schema göra personliga inställningar för om hens pass ska kunna synas för de övriga anställda och om hen vill se de andra anställdas tider eller ej. Dock så kan den som är ansvarig, så kallad *supervisor*, har en öppnare vy och har tillgång till att se de anställdas personuppgifter och schema samt pass oavsett personlig inställning.

Sen har vi även vår alpacka-avatar, Al, som ska fungera som en röd tråd och en hjälp i applikationen. Al är främst tänkt för att öka användbarheten och förbättra *UX* med *mikrointeraktioner*. Exempelvis ger Al en high five efter ett bokat pass och välkomnar användaren vid inloggning.

2.1 Målgrupp

Målgruppen är väldigt bred eftersom Paca är tänkt att kunna fungera som schemaläggning och bokning av pass för företag oavsett domän. För att vi ska kunna arbeta mer konkret har vi skapat en exempel kund, ett fiktivt ICA Supermarket i Malmö. En dagligvaruhandel med ungefär 30 anställda varav 10 stycken är heltidsanställda, 10 stycken är deltidsanställda och 10 stycken har en timanställning. Detta för att alla dessa anställningsformer förekommer hos de flesta företag. Det är även dessa företag som inte endast har heltidsanställa och fast scheman som vår webbapplikation är användbart och intressant.

Vår målgrupp är stor och har därmed ett stort spann mellan användarnas åldrar, erfarenhet av internet och datakunskaper samt erfarenheter. Allt från en student med timanställning till en snart pensionerad charkuterist. Detta påverkar vår utformning och ställer stora krav på användbarheten för Paca. Här är Al en hjälpande hand och ska fungera som en guide och vän till användaren. Målgruppen behöver kunna läsa och ha förståelse för skrift samt fungerande uppkoppling till internet för att använda och utnyttja Pacas funktioner.

2.2 Intressenter

För vårt projekt har vi intressenterna:

- Gruppmedlemmar och utvecklare av Paca
- Testgrupp som får medverka i användbarhetstester
- Handledare och lärare i kursen Systemutveckling och Projekt 1
- *Slutanvändare* av webbapplikationen
- Testgruppen som ska utföra *white-box testing* på projektkoden

Alla intressenter har en påverkan i projektet och på produkten. Handledare och lärare i kursen påverkar vi i gruppen som utvecklar webbapplikationen i form av lärandemål och teoretiskt underlag samt praktisk hjälp i form av verktyg och föreläsningsmaterial. Utvecklarna påverkar produkten i såväl inre som yttre utformning, det vill säga funktionalitet och grafiskt utseende. Här kan handledare och lärare även påverka och begränsa utvecklarna i denna utformning. Testgruppen för användbarhetstest avgör hur väl funktionaliteten och *användargränssnittet* fungerar och testgruppen för white-box testning avgör hur väl koden är strukturerar, kommenterad och fungerar. Slutanvändarna avgör hur väl den färdiga produkten är utformad.

3. Kravdokumentation

3.1 Funktionella krav

Funktionella krav berör vad ett program behöver kunna utföra [3]. Vanligen kan dessa krav mätas i *booleska* termer, det vill säga att de antingen uppfylls eller inte uppfylls. De funktionella kraven delas i sin tur in i användarkrav respektive systemkrav.

De funktionella kraven delas inledningsvis in i användarkrav, för att sedan översättas till en rad systemkrav. De sistnämnda indenterade under sina respektive användarkrav. Vidare följer alla krav den namnstandard som anges i inledningen (1.3 Standard för namngivning av krav), vilket gör det möjligt att se vilken kravtyp ett krav är baserat på dess ID. Kraven är dessutom formulerade i enlighet med MoSCoW-modellen för att på ett ännu tydligare sätt kunna avgöra dess prioritering.

3.1.1 Inloggning

Användaren (anställd eller chef) måste kunna logga in med användarnamn och lösenord, och kunna återställa lösenord med hjälp av att ange användarnamn eller e-postadress.

F.AK.M-1 Användaren måste kunna logga in med användarnamn och lösenord

En personlig inloggning för att skapa integritet för användaren och en personlig interaktion.

F.SK.M-2 Databas måste lagra lösenord, användarnamn och kontaktuppgifter

I databasen måste alla användarnas uppgifter vara lagrade, så att varje anställd har sitt eget konto med sparade uppgifter (synliga för dem och/eller deras chef). Vid inloggning kontrolleras det användarnamn och lösenord som användaren anger med de uppgifter som står i databasen. Vidare måste användaren få tillgång till dessa vid bortglömda inloggningsuppgifter.

F.SK.M-3 Databasen måste tillåta ändringar från användaren

När användaren ändrar sina uppgifter ska dessa skriva över dem gamla i databasen och beständigt lagras.

F.SK.M-4 Användaren måste inte logga in varenda gång hen besöker webbapplikationen

Användaren måste kunna kryssa för en "Håll mig inloggad på denna enheten" som accepterar och planterar cookies på användarens enhet.

3.1.2 Kalender

Användaren ska se en grafisk kalender och göra personliga inställningar för hur denna ska visas.

F.AK.M-5 Användaren måste kunna se schema i vecko- och månadsvy

Användaren måste kunna se schemat i såväl veckovy som månadsvy och själv kunna ställa in detta.

F.SK.M-6 Presentera schema veckovis

Användargränssnittet ska visuellt, med JavaScript, HTML och CSS presentera kalendern i veckovy och svagt men tydligt markera den aktuella dagen.

F.SK.M-7 Presentera schema månadsvis

Användargränssnittet ska visuellt med JavaScript, HTML och CSS presentera kalendern i månadsvy och svagt men tydligt markera den aktuella dagen.

F.AK.S-8 Vid klickning på ett specifikt pass bör mer information om passet presenteras

Användaren bör vid klickning på ett pass få fram mer information om passet. Starttid, sluttid, rasttid och behörighet/avdelning.

F.SK.S-9 Visa/Dölj information

När användaren klickar på ett pass ska information om passet hämtas från databasen och med JavaScript visas i användargränssnittet intill kalendern. Vid andra klicket ska informationen döljas igen.

F.AK.M-10 Användaren måste kunna boka tillgängliga pass

Användaren måste kunna boka tillgängliga pass via dashboard på startsidan eller i kalendern. Användaren ska vid bokning även ha möjligheten att lämna en kommentar.

F.SK.M-11 När ett pass skapas skall dess status lagras i databasen

När användaren bokar ett pass uppdateras databasen och det lagrade passet ändrar status och "ägare" av passet.

F.SK.M-12 Visa tillgängliga pass i kalendern

Användargränssnittet ska med JavaScript, HTML och CSS visuellt visa de pass med statusen tillgänglig i databasen.

F.AK.S-13 Användaren bör kunna lägga ut ett bokat pass för passbyte

Användaren bör kunna ändra statusen på pass, från bokat till vill byta bort.

F.SK.S-14 Databasen ändrar datan i passet på användarens direktiv

När användaren ändrar statusen på sitt pass i schemat ska databasen via pythonkoden även uppdatera databasschemat och passet ska visas för övriga användare i dashboard och i kalendern.

F.AK.S-15 Användaren bör kunna ta över ett pass i passbyte

När en anställd har lagt ut ett pass för passbyte bör en kollega till denne kunna boka det passet och få in det i sitt schema.

F.SK.S-16 Byta ägare till specifikt pass i databasen, vid passbyte (utan involvering av chef)

När användaren bokar ett pass som en annan användare vill byta bort bör Python kontrollera om behörigheten är likvärdig, och via databasen uppdatera ägare till passet.

F.SK.S-17 Byta ägare till specifikt pass i databasen, vid passbyte (med involvering av chef)

När användaren vill boka ett pass som en annan användare vill byta bort, men behörighet ej är likvärdig behöver supervisor/schemaansvarig godkänna bytet.

F.AK.C-18 Användare skulle kunna exportera schema till iCal, XLS-fil och pdf

Användaren ska möjligen kunna exportera sitt schema till externa system för att samverka med dessa samt för att öka användbarheten.

F.SK.C-19 API mellan databas och externa system

Användaren ska möjligtvis kunna exportera sitt aktuella schema till externa system via en API mellan databasen och dessa system.

F.AK.C-20 Användaren bör kunna ändra inställningar för individuell schemavisning

Välja och bestämma om man vill se sina egna bokade pass och schema, tillgängliga pass för bokning (såväl lediga som utbytbara pass) och andras schema.

F.SK.C-21 Visuellt anpassa vy i personligt informationsflöde

Användargränssnittet bör kunna anpassas efter användarens inställningar och med en filterfunktion med Pythonkod och JavaScript visuellt visa användaren valda alternativ.

F.AK.S-22 Användaren bör kunna genomföra individuella schemainställningar om publik schemavisning

Användaren ska kunna bestämma om hens schema ska kunna visas för kollegorna eller ej, dock kommer pass alltid visas för supervisorn.

F.SK.S-23 Hanter integritet status med visningsalternativ

Användaren bör kunna styra huruvida dennes pass ska synas för kollegor eller ej genom JavaScript eller pythonkod.

3.1.3 Profil

Varje användare har tillgång till en egen profil i användargränssnittet som hen själv kan hantera.

F.AK.M-24 Användaren måste kunna ändra personuppgifter och kontaktuppgifter

Användaren ska kunna ändra sina personuppgifter och kontaktuppgifter i profilen.

F.SK.M-25 Databasen måste tillåta användarnas ändringar

När användaren ändrar sina kontaktuppgifter och personuppgifter ska dessa skrivas över i databasen och de nya ändringarna sparas.

F.SK.M-26 Regelbundna transaktioner

Databasen ska genomföra regelbundna transaktioner för att ändrad data ska sparas. Skulle detta misslyckas och en rollback ske så ska användaren meddelas att ändringen misslyckades.

F.AK.C-27 Användare ska ha en profilbild

Varje användare ska ha möjlighet till en synlig profilbild och ska kunna ändra denna själv.

F.SK.C-28 Hämta färdiga bibliotek av bilder

När en användare skapas skulle tilldelas denne en *default* profilbild, vilken är hämtad från ett externt bibliotek med API. I databasen skulle ett tiotal bilder som slumpvis tilldelas en användare.

F.SK.C-29 Ändra profilbild

Användaren skulle kunna ändra sin profilbild mellan andra bilder inlagda i databasen samt ladda upp en egen bild från sin enhet och lagra i databasen.

3.2 Kvalitativa krav

Kvalitativa, eller icke-funktionella, *krav* handlar istället om på vilket sätt programmet måste bete sig [3]. Detta innefattar vanligtvis en mätskala, i kontrast till ett allt-eller-inget-värde, såsom hur en mjukvara kan vara mer eller mindre användarvänlig.

3.2.1 Användbarhet

Krav för användbarhet syftar till att bryta ner hur projektet ska förmedla UX.

IF.A-1 Enhändig användning på skärm, touchplatta eller mus

Användaren ska endast behöva använda en hand för att integrera med webbapplikationen och utföra sin tänka uppgift.

IF.A-2 Intuitivt användargränssnitt

Användarna ska inte behöva någon introduktion eller utbildning av användargränssnittet för att effektivt kunna använda webbapplikationen.
Intuitiv design och användargränssnitt med självförklarande namn och funktioner för en god UX.

IF.A-3 Feldensitet < 3 per användningstillfälle

Användaren ska inte klicka fel, i tron om att hen klickat rätt och därmed felanvända webbapplikationen, mer än tre gånger per användningstillfälle.

IF.A-4 Enminutsbokning

Om en användare endast ska boka ett pass eller lägga ut ett pass för passbyte ska det inte ta längre tid är 60 sekunder/ 1 minut för användaren att logga in, utföra sin tänkta syssla, få feedback på att ändringen genomförts och sen logga ut eller stänga ner applikationen.

3.2.2 Begränsningar

De begränsade kraven begränsar ramen för projektet och ska inte brytas eller gå utanför.

IF.B-1 Programkod Python

Programmet ska skrivas i programspråket Python.
Projektmedlemmarna är som grupp mest bekväma med programspråket Python.

IF.B-2 Interaktivitet JavaScript

Programspråket JavaScript ska användas för att öka interaktiviteten och det dynamiska på webbapplikationen. Ska ta över där CSS och HTML inte räcker till för användargränssnittet.
CSS och HTML är alla i gruppen bekväma med men vi är införstådda med bristerna dessa kodspråk har för interaktiviteten och därför är en ansvarig och ska sätta sig in i JavaScript.

IF.B-3 Portabilitet

Webbapplikationen ska vara responsiv och fungera på såväl desktop som mobil och alla skärmstorlekar däremellan.

IF.B-4 Webbläsare

Webbapplikationen ska fungera på de största webbläsarna, det vill säga Chrome, Firefox och Opera.

IF.B Kodstruktur och standarder

Vi har bestämt att följa vedertagna och globalt använda standarder och strukturer för att underlätta för externa parter som ska läsa kod dokumenten, och för att skapa en enhetlig struktur från början.

IF.B-5 Kodstruktur Python

För kod skriven i Python ska standarden PEP8 följas.

IF.B-6 Kodstruktur JavaScript

För kod skriven i JavaScript ska standarden i enlighet med ECMAScript följas.

IF.B-7 Standarder HTML och CSS

För HTML och CSS ska W3C:s standarder för kod och tillgänglighet (WCAG) följas.

IF.B-5 Kodstruktur CSS

För kod skriven i CSS ska filstrukturen och riktlinjerna för skriptspråk Sass följas.

IF.B-5 Kodkonventioner PostgreSQL

Postgres egna dokumentation för kodkonventioner ska följas i databasen.