# KTH Projekt och projektmetoder II-1302

# Kravspecifikation
# Flower Power

*Skapad datum: 180427*
*Senast ändrad: 180521*
*Dokumentansvarig: Erik Holm*
*Version: [4]*
*Godkänd av:*

## Bilagor

*1. Use case Model sprint 3*
*2. Exempel på use Case uppdelat i tasks för arbetsgrupp*
*3. Exempel på use case i fall av anpassad user slice i sprint 3*

## Innehåll

# 1. Introduktion

## *1.1. Syfte*

*Syftet med kravspecifikationen är att ge en inblick i projektet "FlowerPower":s projektmetod och hur projektet har arbetat utifrån kundens krav. På ett kort och koncist sätt ska en eventuell efterträdare snabbt kunna sätta sig in i det utförda arbetet och kunna fortsätta i linje med den hittills utarbetade strukturen. Projektet använder ett agilt arbetssätt enligt Scrum vilket kommer sammanfattas och kopplas till det aktuella projektets utformning under punkt 8 - "Krav".*

# 2. Definitioner och förkortningar

## *2.1.* **Prescriptive approach**

Antalet begränsningar. En mer prescriptive approach har inte samma valfrihet, färre öppna val.

## *2.2.* **Skikt**

Ett skikt är en grov indelning av en applikation på fysisk nivå. Ett skikt ansvarar för en grupp av funktioner.

# 3. Projektets syfte

## *3.1.* Projektets bakgrund

KTH vill utforma en produkt som kan underlätta bevattning av växter i deras lokaler. En förstudie visade växters naturliga del i det svenska hemmet. Plantagen är nordens ledande kedja för försäljning av växter. De är därför en indikator kring trender och försäljning i Sverige. Mellan år 2016 och 2017 ökade försäljningen med 2%.[1] Med en ökad försäljning ökar även behovet till skötsel och FlowerPowers tänkta kundkrets.
Tid. Vi är alltid i behov av mer tid. Tid att tvätta, tid att laga mat, tid att vattna blommor. Marknaden att optimera/effektivisera den tid vi spenderar till varje syssla har blivit en marknad i sig. Familjer får livsmedel hemkörda till dörren, husdjur får tillfälliga ägare när deras husse och matte reser bort. Flower Power är ytterligare ett naturligt steg i den riktningen. Behöver man jobba över, stanna kvar på affärsresan ytterligare några dagar så ska man ändå inte behöva mötas av en vissen växt när man kliver innanför farstun. Det blir extra svårt när man bor själv och inte kan dela upp ansvaret. Det är ett problem eftersom ensamboende utgör 38% av alla hushåll[2]. Flower Powers applikation skickar påminnelser och sköter bevattningen åt en.

## *3.2.* Mål

Produkten ska senare kunna implementeras i alla hem/företagslokaler. En sensor ska kopplas till vald växt, sensorn är i sin tur kopplad till en databas där användaren ska se värden för samtliga växter i hemmet. Ett enkelt paket ska kunna beställas för att på egen hand kunna sättas upp i det egna hemmet. För utförligare implementeringsdokumentation se punkt 6.1.

## *3.3.* Effekt

Den långsiktiga effekten av FlowerPower är en modul med låg kostnad och behov av skötsel som underlättar bevattningen i svenska hem. Det kommer i sin tur leda till en ökad livskvalitet för den tänkta användaren då FlowerPower avlastar från den dagliga arbetsbördan. Ökad livskvalitet minskar stress och den övergripande folkhälsan, vilket är i statligt intresse.

# 4. Kund och andra intressenter

## *4.1.* Lista över kund och intressenter

**KTH** – Första intressent som har påbörjat projektet i form av en kravformulering
**Statlig intressent** – I och med främjandet av folkhälsa kan en möjlig intressent vara statliga företag, i form av subventioner utan vinstintresse.

# 5. Användare

*5.1.* Användargrupper

Den tänkta målgruppen är bred och innefattar både privatpersoner (stora familjer samt ensamstående) och företag av varierande storlek. Det är viktigt att utvärdera varje användares förkunskaper. Ett företag kan till exempel ha en teknikansvarig som har betydligt lättare för att implementera modulen än vad en privatperson har. Det är viktigt att ha i åtanke vid uppdelningen av tasks till varje use case.

# 6. Avgränsningar

*6.1.* Implementation

Implementationen innefattar en Arduino Uno med tillhörande sensor, en ESP8266 wifi-modul samt en raspberry pi.

- ESP8266 kopplar upp sig till ett lokalt nätverk och kommunicerar genom serial till arduinon.
- Raspberry Pi har en databas med alla värden och en tråd för varje planta, tråden skickar en HTTP request som plockas upp av ESP8266
- Requesten skickas vidare till Arduino. Arduinon väntar på requesten i en loop-funktion
- Svaret blir en string i ett text/html svar som skickas till ESP, vidare till raspberry.
- Raspberry Pi får en string som den kan hantera och skicka vidare till en SQL-databas samt till front end.

För ytterligare dokumentation, se teknisk rapport för implementation av wifi samt arkitekturdokumentation.

# 7. Fakta och förutsättningar

# 8. Krav

*8.1.* Introduktion av krav

Projektets krav- och kundansvarig arbetar utifrån en agil arbetsmetod enligt Scrum. Projektet har kantats av sen kontakt med kund och allmänt vaga kravformuleringar, därför har scrum hjälpt till med sin flexibla anpassning till förändringar. En vattenfallsmetod hade låst projektet på en väg, vilket kräver en större flexibilitet i den initiala planeringen. KanBan hade även kunnat fungera i projektet, samtidigt var det viktigt att ha en tydligare struktur av roller och uppgiftsfördelning, en mer preskriptiv approach då gruppmedlemmarna har en relativt kort introduktion i metodlära och behövde guidas i studiesyfte. Det var bra att välja en "batch" med tasks till varje sprint samt ha en rollfördelning i gruppen för att få igång arbetet snabbt, något som inte ingår i KanBans metod utan snarare Scrum (KanBan and Scrum Info Q sid.11).

Det har under projektets gång varit viktigt att undvika alltför tekniska cases/stories, det hör hemma i tasks som man arbetar fram för teamet. På så sätt kan vår kund vara med i avvägningar kring tradeoffs (Scrum and XP from the trenches, sid. 47). Se bilaga 3 för exempel vid uppdelning av tasks samt punkt "punkt" för ytterligare förklaring. Observera även att kraven är uppdelade i funktionella samt icke-funktionella krav. Alla krav är inte alltid riktade till användarens direkta interaktion med produkten utan även "de restriktioner och krav som finns på en systemtjänst" (Sommerville). Trots att användaren inte ser kraven direkt så kommer de ändå vara viktiga för hur kunden uppfattar produkten. Därför är det minst lika viktigt att de icke-funktionella kraven uppmärksammas i det fortsatta arbetet (se punkt 8.4).

*8.2.* User Stories

- Som användare vill jag gå in på hemsidan när jag är på semster och se fuktvärden för att få en snabb överblick
- Som användare vill jag få en notis vid ett visst fuktvärde för att slippa hålla koll på det själv, helst till min mail
- Som användare vill jag kunna ändra fuktkänslighet för att anpassa bevattning till olika växter då jag har kaktusar som inte kräver lika mycket vatten
- Som användare vill jag kunna starta bevattning för vald blomma för ökad kontroll över bevattningsprocessen
- Som användare vill jag kunna nödstoppa bevattningen för ökad kontroll

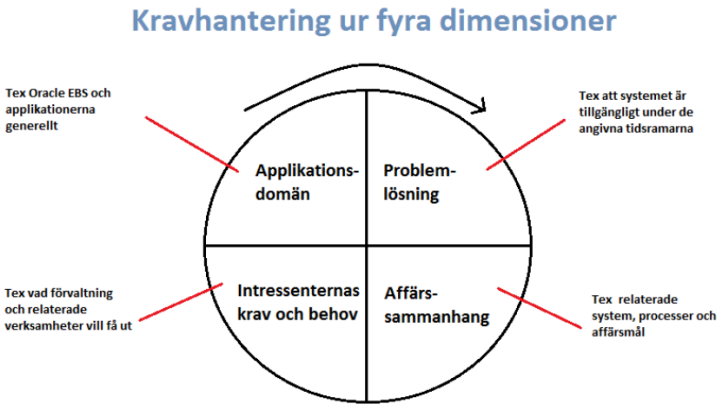
Såg listan ut såhär från början? Svaret är nej, den har redigerats löpande under projektets gång. Som Kinberg föreslår (Scrum and XP from the trenches, sid. 41) fanns det en checklist för "definition of ready" och "definition of done" där hemsidan inte platsade i vår checklist "definition of done". Det kan liknas vid avvägningen man gör mellan stories till use case (Jacobson 2015). Det

får dock inte blandas ihop med stories som Jacobson pratar om. Story att gå in på hemsidan ansågs till exempel inte var redo att appliceras som use case eftersom den var för övergripande och var en del av vattna blommorna/se över fuktvärdens main flow. Den kunde istället läggas in i use case:et se fuktvärden utan att bli ett eget case, vilket blev andra sprintens huvudmål. Hemsidan blev istället en task för det arbetande teamet. Allt för att möta kundens förväntningar. Projektgruppen arbetar både Top Down samt  Bottom Up, men till en början Bottom Up (Use Case 2.0 sid. 17). Två stories togs fram för att undvika att blommor vissnade och definierades sen som UC. Därefter definierades ett main flow och alternative flow. Vissa alternative flows blev ett eget UC, men ett sekundärt sådant. Det visades genom en tydlig rangordning. I punkt 8.3 presenteras de i den nämnda rangordningen.

*8.3.* Use Cases av funktionella krav.
Prioritering i samrådan med kund togs fram utifrån kundens stories och rangordnades på följande sätt. Kraven delades upp i nödvändiga krav och "förbättringar" som i samrådan med kund fick finslipa produkten i mån om tid. Uppdelningen var ett grepp på metoden att dela upp ett primärt UC:s alternative flow till sekundära UC.

1. Översikt av fuktvärden
2. Få notis vid fuktvärde
3. Ändra sensorkänslighet
4. Starta bevattning
5. Nödstoppa bevattning
6. Kunna se över värden på språng (mobil)

Det är viktigt att fortfarande koppla varje story till sitt case eftersom en ändring kan komma att ändra prioritering och struktur av kraven. Den iterativa kravhanteringen kan leda till många hinder som illustrerat i bilden nedanför.




Problemlösning är hur produkten löser kunden krav på en produkt, men för att nå dit krävs även en krav hantering kring applikationsdomänen, t.ex. systemkrav. De här kraven kommer oftast upp under projektets gång. Därför är det extra viktigt att kopplingen mellan Case och Stories dokumenteras i en kravspecifikation. Bara för att man uppfyller kravet i en stund betyder inte att kravet är uppfyllt för kommande iterationer av produkten då förutsättningar ändras. En Bottom approach användes för att formulera första UC. Därefter skrevs main flow samt alternative flow till UC för att tillåta en Top Down approach. Krav som att stänga av vatten upptäcktes genom alternative flow och kunde formuleras som ett sekundärt UC. Det är viktigt att uppdatera specifikationen för att ha en översikt över kopplingen mellan UC och stories eftersom ändringar kan behöva en ny story eller till och med slice. Se punkt 8.5.3 för exempel och förtydigande.

*8.4.* Icke-funktionella krav

8.4.1. Användbarhet
- 8.4.1.1. Lättförståelig applikation ska eftersträvas genom att tydligt visa och guida användaren till respektive blomma, med pedagogiska färg- och typsnittsval
- 8.4.1.2. Data
  Värden ska endast presenteras som heltal.

8.4.2. Felhantering

8.4.2.1. Alla programmerares tester ska skrivas i pytest för ökad kompabilitet och för debug.
8.4.2.2. Väntan för exception vid wifi-connection mellan Arduino och Pi
Ett exception ska slängas fem sekunder efter klassen ArduinoConnections försök till server request mot arduinon/sensor. Det är för att underlätta den kommande felhanteringen i trådarna "plant" till huvudtråden. Vid ändring av tid, var tydlig med att dokumentera tid och motivering av ändring.

8.4.3. Säkerhet
8.4.3.1. Datatermer får inte användas av användaren
För att undvika cross site-scripting måste indata från front end hanteras.
8.4.3.2. Bara auktoriserade användare kan se sina växters värden.
Det här säkerställs genom användarnamn och lösenord för att ta del av de värden som är knutet till kontot.

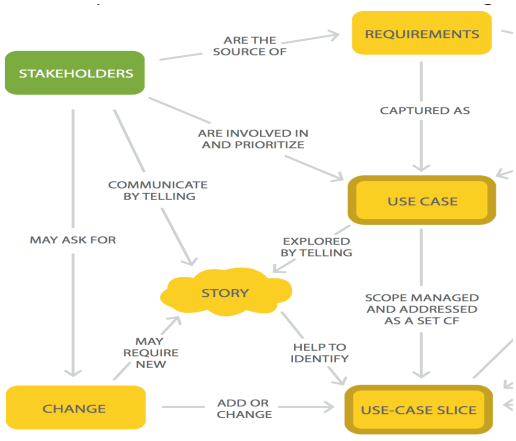
8.4.4. Användargränssnitt
8.4.4.1. Pedagogiskt gränssnitt
Pedagogiskt färg- och typsnittsval med val av språk. Extern style guide bifogas och dokumenterar eventuella ändringar.

## *8.5.* Förtydliganden och tillägg av krav

8.5.1. Kopplat till punkt 8.1 - "Alltför tekniska krav"
Under Sprint 2 var ett krav att kunna kommunicera mellan raspberry pi och arduino uno. Det är ett alltför tekniskt task som inte borde presenteras för kund, inte ens som slice. Den lades därför in som ett task under use case:et "se översiktlig vy av fuktvärden". Ett case som är utifrån användarens perspektiv och tydligt knutet till vår nämnda tidslinje. Uppdelningen sker enbart i ett task för arbetsgruppen att bry sig om. User Case ska vara av direkt värde för användaren.

8.5.2. Kopplat till punkt 8.2 - Val av stories och "Definition of done"
Beslutet att slå ihop hemsidan med översikten av fuktvärden togs bland annat på grund av resurserna. Kunskapen kring utveckling av hemsida samt tidsramen ansågs vara tillräcklig för att minimera risk att inte uppnå iterationens mål. Ponerar man andra förutsättningar hade hemsida-storyn kunnat vara en del av use case i form av en slice, där vi hårdkodade värden till vår hemsida.
Att föra in hemsidan som ett case, även om det är sekundärt, öppnar för ett missförstånd med kunden, eftersom det frångår de förväntningar och krav som de har på modulen. Hur ett Use Case ska uppnås är upp till teamet att arbeta fram.

8.5.3 Kopplat till 8.3 - Ändringar från kund med exempel
Ett problem, följt av en avvägning, skedde i sprint 3. Ett case hade ett task som skulle koppla samman en Telstick med en pump för att användaren skulle kunna starta en bevattning. På grund av hårdvaruförseningar var projektgruppen osäkra på om det var ett möjligt mål under sprinten. Use caset delades därför i samråd med kund upp i en slice för kommunikation med en enklare modul. Slicen var en tydlig tårtbit till målet kunden förväntade sig som kunde visas upp vid ett sprintdemo. Här hade kravspecifikationen en central roll när man behövde se över vilka Case som påverkas av ändringen.



Figuren ovanför (Jacobson 2005) visar just beroenden i kravprocessen. Vid ändring kan en ny story behöva skrivas samt en slice läggas till/ändras, vilket kommer att påverka relaterade use case. För att lokalisera detta caset krävs dokumentation. Naturligtvis är det ett större problem ju fler komponenter (slices) ett use case har delats upp i när ändringen av UC sker. (Jacobson 2005, 16-17).
Den slice som kom ur beslutet är ytterligare ett steg mot den fullständiga insynen i ett projekt som diskuterades i inledningen. Färre missförstånd med kund sparar på tidsresurser och arbetskraft.

# 9. Osäkerheter och risker

*9.1.* Beroenden

9.1.1. Vår front end är beroende av att hårdvaran kopplas rätt hos leverantör. Det är därför viktigt att felsökning är möjligt på varje nivå för att kunna lokalisera problemet. Tester skrivs i pytest som standard för projektet.

*9.2.* Stabilitet

9.2.1. Brandvägg och säkerhet hos det valda nätverket kan försvåra uppkopplingen hos ESP8266. Uppkopplingen tar för givet att den är auktoriserad efter lösenord och användarnamn har angivits.

*9.3.* Prestanda

9.3.1. Arduinon kan ha svårt att serva fler än en sensor, vilket kommer bli en flaskhals för projektet. Vid implementering av ytterligare sensorer, utfärda ett task att se över möjligheter till strömförsörjning samt tillägg av pins innan implementeringen blir en task i en kommande sprintiteration.

# 10. Användardokumentation och utbildning

*10.1.* Rollutbildning

Till varje roll ingår ett flertal seminarium och kunskapstester för att underlätta samarbetet i projektgruppen

# 11. Tidsuppskattning

*11.1.* Tidsuppdelning i sprintmoment

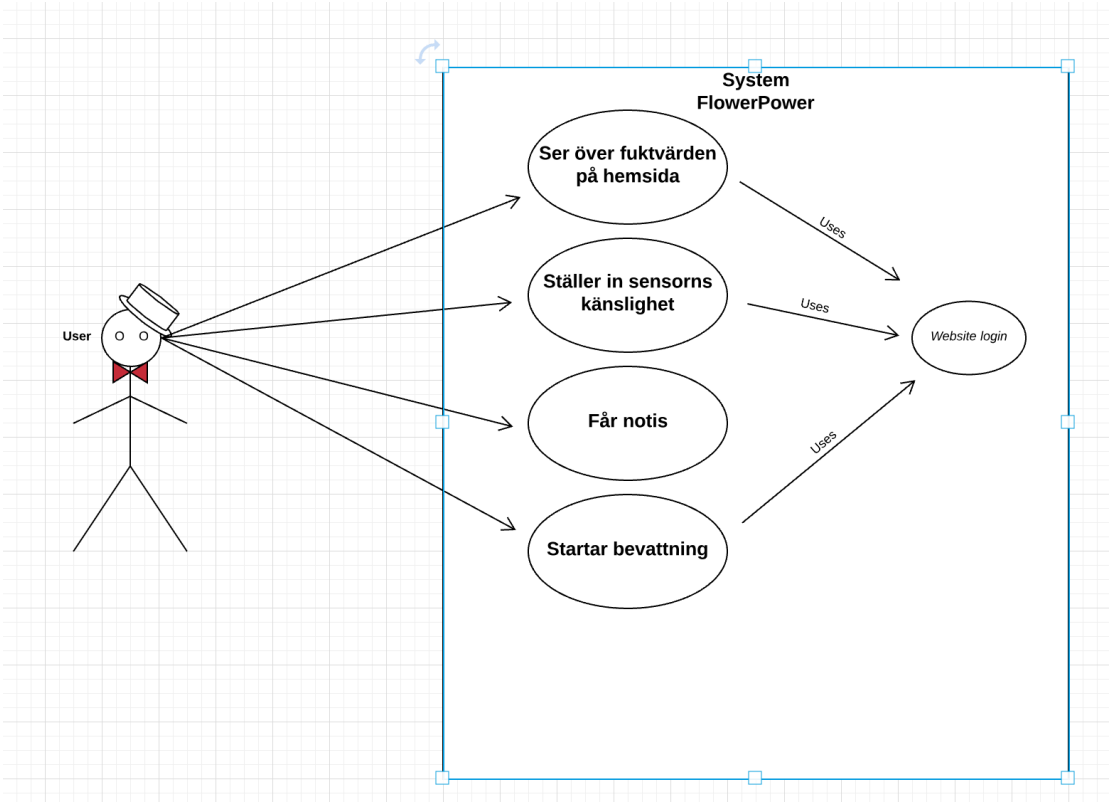
Projektets tidsuppskattning löper under en högskoletermin. Enligt Scrum skedde sen uppdelningen i iterationer. Första och andra sprinten hade en burndown på 20 poäng. Medan tredje hade en uppskattad tid på 30 poäng. Varje poäng är 4 timmar. Anledningen till en ökad tidsuppskattning berodde på att arbetskraften behövde delas upp på två parallella projekt (kursrapport).

# Referenser

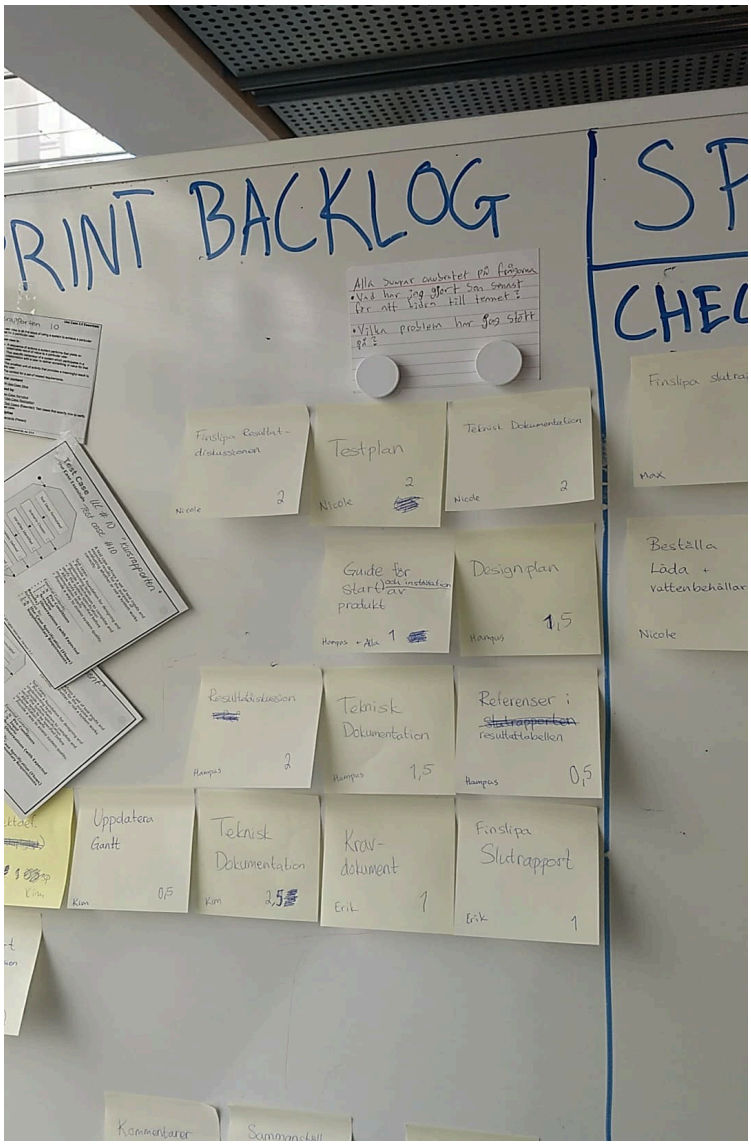
**[1]** https://www.ratos.se/globalassets/global/05_investor-relations/delarsrapporter/2017_q1/delarsrapport-q1-sve.pdf

**[2]** https://www.scb.se/sv_/Hitta-statistik/Artiklar/Atta-av-tio-delar-hushall-med-nagon/

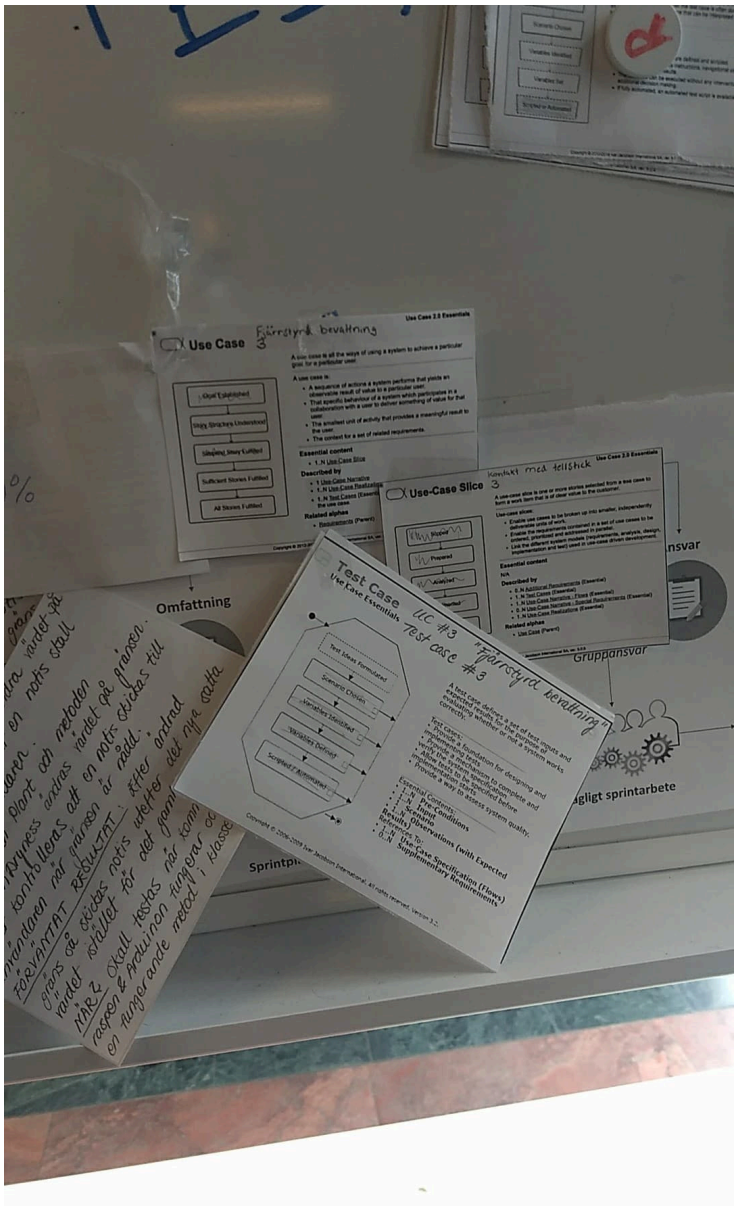
# Bilaga 1


System
FlowerPower
Ser över fuktvärden på hemsida
Ställer in sensorns känslighet
Får notis
Startar bevattning
Uses
Uses
Uses
Website login
User

**Bilaga 2**


RINT BACKLOG
SP
CHEC
Alla svarar osv kring på frågorna
• Vad har jag gjort sen senast för att bidra till teamet?
• Vilka problem har jag stött på?
Finslipa Resultat-diskussionen
2
Nicole
Testplan
2
Nicole
Teknisk Dokumentation
2
Nicole
Finslipa slutrapport
max
Beställa Låda + vattenbehållare
Nicole
Guide för start() av produkt
Hampus + Alla
1
Designplan
1,5
Hampus
Resultatdiskussion
2
Hampus
Teknisk Dokumentation
1,5
Hampus
Referenser i resultattabellen
0,5
Hampus
Uppdatera Gantt
0,5
Kim
Teknisk Dokumentation
2,5
Kim
Krav-dokument
1
Erik
Finslipa Slutrapport
1
Erik
Kommentarer

**Bilaga 3**


Use Case 3 Förstyrd beställning
Use-Case Slice 3 Kortet med tellstick
Test Case UC #3 "förstyrd beställning"
Test case #3
Omfattning
Gruppansvar
Sprintpl
agligt sprintarbete