**Document de analiză a cerințelor clientului**

**Sera Smart – Echipa 17**

**Membrii echipei: Reznicencu Sergiu, Dudau Vlad, Mindrescu Albert, Bogdan Staicu, Iulian Gal, Sorin Neculae**

*Scopul acestui document este colectarea și comunicarea întregii echipe care sunt cerințele pe care le aveți de îndeplinit. Important de reținut că documentul nu este perfect secvențial. Scrieți ce credeți la un punct, iar după ce rezolvați și un al doilea punct, puteți reveni pentru a șlefui punctele anterioare. Este important să urmăriți o coerență globală a documentului, în urma acestor reveniri. Porniți de la acest template, dar nu păstrați aceste instrucțiuni în documentul final.*

## Scopul aplicației:

*Identificați care este scopul principal al aplicației pe care o veți dezvolta. Pentru construirea scopului, veți începe de la cerințele de proiect din Curs I, dar îmbinat cu specificul aplicației pe care vă propuneți să îl faceți.*

*Aplicatia are ca scop modificarea serei traditionale prin automatizarea acesteia, folosindu-se de date de natura variabila pe care le preia din mediul inconjurator (lumina, temperatura, umiditate etc.). Aceasta aplicatie poate fi rulata si parametrii pot fi modificati de catre utilizator remote prin intermediul telefonului/desktopului si va oferi un feedback in timp real a conditiilor din sera.*

## Obiectivele aplicației:

*Definirea unui număr mic de obiective care acoperă cele mai importante perspective ale aplicației. Puteți începe de la cum ar arăta un proiect minim viabil (eventual pornind de la cerințele din curs). Ce valoare aduce clientului? Ce particularități de dezvoltare trebuie avute în vedere? Ce metrici de evaluare sunt relevante?*

*În general scopul și obiectivele se construiesc împreună; scopul fiind o privire de ansamblu a proiectului (și deci a obiectivelor) iar obiectivele încearcă să clarifice care sunt livrabilele, care împreună definesc scopul.*

* Sera va avea ca scop maximizarea cantitatii de recolta din interiorul acesteia de-a lungul timpului.
* Aceasta va economisi resursele neregenerabile folosite si timpul persoanei care o detine/opereaza.
* Aplicatia ar trebui sa fie usor de folosit si foarte intuitiva*.*

## Grupul țintă

*Cui îi adresați această aplicație? Care este profilul utilizatorului a cărui nevoie căutați să o satisfaceți? În special trebuie să surprindeți felurile diferite în care utilizatori diferiți folosesc același feature (de exemplu, cum folosește un gamer numpad-ul, și cum îl folosește un contabil) Folosiți User Stories.*

Aplicatia se adreseaza agricultorilor, dar si amatorilor pasionati de legumicultura, ce cauta sa isi automatizeze propriile culturi*.*

* Ca agricultor, as vrea sa pot sa accesez o serie de statistici relevante despre sera, pentru a optimiza calitatea viitoarei recolte.
* Ca agricultor, as vrea sa pot modifica parametrii din interior, precum umiditatea solului, temperatura aerului, cantitatea de lumina, pentru a influenta recolta.
* Ca utilizator, mi-ar fi foarte util ca aplicatia sa fie foarte usor de folosit.
* Ca legumicultor amator, as dori sa pot sa ma descurc cu sera intr-un mod cat mai necostisitor.

## Colectarea cerințelor

*Care sunt cerințele pe care userii din story-urile de mai sus le-ar cere? Care sunt cerințele de sistem care apar ca o consecință a cerințelor userului (performanța în anumiți parametrii; utilizarea anumitor resurse externe; utilizarea unui mod specific de dezvoltare ș.a.m.d.). Această zonă este bine completată dacă are un număr cât mai mare de cerințe (relevante), chiar dacă sunt prea multe, sau depășesc un pic dimensiunea proiectului pe care îl avem în vedere. Aici trebuie să înțelegem tot ce* ***s-ar putea*** *face.*

- prezenta unui spatiu de stocare ce retine in timp datele senzorilor si compileaza statistici pe baza lor.

- folosirea unui numar minim de senzori de temperatura, umiditate, lumina.

- accesarea unui api extern pentru preluarea datelor metereologice.

- citirea unui senzor de temperatura.

- citirea unui senzor de umiditate.

- citirea unui senzor de lumina.

- citirea unui senzor de miscare.

- performanta sistemului nu este critica.

- conexiunea la internet.

- crearea unei interfete minimaliste.

- adaptarea aplicatiei atat pe mobil cat si pe desktop.

## Interpretarea și prioritizarea cerințelor

*Dintre cerințele de mai sus vom interpreta și prioritiza cerințele.*

*1. Label-uiți cerințele funcționale / non-funcționale. Cerințele funcționale sunt cele care îndeplinesc o nevoie care a reieșit dintr-un user story, și răspund la întrebarea ce trebuie aplicația să facă. Cerințele nonfuncționale sunt cele care descriu calitățile de sistem, și răspund întrebărilor de tipul cum trebuie să fie un anumit feature sau aplicația cu totul? Acest label-ing e suficient să îl faceți în documentul de analiză, nu e necesar să îl cărați pe mai departe.*

* prezenta unui spatiu de stocare ce retine in timp datele senzorilor si compileaza statistici pe baza lor. Functional
* folosirea unui numar minim de senzori de temperatura, umiditate, lumina. Nonfunctional
* accesarea unui api extern pentru preluarea datelor metereologice. Functional
* citirea unui sensor de temperatura. Functional
* citirea unui sensor de umiditate. Functional
* citirea unui sensor de lumina. Functional
* citirea unui sensor de miscare. Functional
* performanta sistemului nu este critica. Nonfunctional
* conexiunea la internet. Functional
* crearea unei interfete minimaliste. Nonfunctional
* adapatarea aplicatiei atat pe mobil cat si pe desktop. Nonfunctional

*2. Gruparea cerințelor*

*Identificați criterii de gruparea cerințelor care ulterior credeți că vă vor ajuta la dezvoltare.*

*- Folosiți criterii pentru a grupa cerințele într-un mod care vi se pare util – după zona de tehnologie (BE, DevOps ș.a.) după eventualele module ale app (comunicare, procesare, stocarea datelor ș.a.), după feature-uri. Menționați aceste categorii și în documentul de analiză.*

STOCAREA DATELOR

* prezenta unui spatiu de stocare ce retine in timp datele senzorilor si compileaza statistici pe baza lor.

PERFORMANTA

* folosirea unui numar minim de senzori de temperatura, umiditate, lumina.
* crearea unei interfete minimaliste.
* performanta sistemului nu este critica.

FUNCTIONABILITATE

* adapatarea aplicatiei atat pe mobil cat si pe desktop.
* conexiunea la internet.

APIs

* accesarea unui api extern pentru preluarea datelor metereologice.

CITIREA SI PRELUCRAREA DATELOR

* citirea unui sensor de temperatura.
* citirea unui sensor de umiditate.
* citirea unui sensor de lumina.
* citirea unui sensor de miscare.

*3. La acest moment puteți să creați un repo de GitHub pentru proiectul vostru și să puneți cerințele într-un back-log de issue-uri în GitHub Issues. Folosiți grupările de la 2 pentru a crea label-uri relevante pentru fiecare issue.*

*Resources:*[*The Product Backlog: A Step-by-step Guide*](https://www.toptal.com/product-managers/agile/product-backlog-step-by-step-guide) *În general sunt bune articolele de la toptal pe software engineering.*

*4. Play planning poker. Planning poker e un joc prin care toți membrii echipei evaluează prioritatea și dificultatea taskurilor în raport cu abilitățile individuale. Media acestor evaluări generează nivelul de prioritate și dificultate final al cerinței.*

*Recomand aplicația de aici: scrumpoker.online (pare că are și o integrare cu GitHub, dar nu am testat-o)*

*Pentru fiecare issue, veți juca câte două runde de planning poker. Trebuie să fiți într-un call. După ce alegeți issue-ul, fiecare user se autentifică în aplicație, și simultan ar trebui să votați dificultatea acelui issue. Faceți media și notați-o. Discutați dacă considerați că este cazul, și feel free să schimbați rezultatul dacă vi se pare că nu e potrivit.*   
*Repetați pentru a identifica prioritatea taskului (adică cât de valoros este pentru aplicația finală). Notați și acest rezultat la fiecare task.*

Poker planning dupa dificultate:

1. prezenta unui spatiu de stocare ce retine in timp datele senzorilor si compileaza statistici pe baza lor. Functional 3 3 3 4 = 3.25

2.folosirea unui numar minim de senzori de temperatura, umiditate, lumina. Nonfunctional 1 2 1 2 = 1.50

3. accesarea unui api extern pentru preluarea datelor metereologice. Functional 2 3 2 3 = 2.50

4.citirea unui sensor de temperatura. Functional 2 2 2 2 = 2

5. citirea unui sensor de umiditate. Functional 2 2 2 2 = 2

6. citirea unui sensor de lumina. Functional 2 2 2 2 = 2

7.citirea unui sensor de miscare. Functional 2 2 2 2 = 2

8. performanta sistemului nu este critica. Nonfunctional 1 1 1 2 = 1.25

9. conexiunea la internet. Functional 4 2 1 3 =2.50

10. crearea unei interfete minimaliste. Nonfunctional 3 3 3 4 =3.25

11. adapatarea aplicatiei atat pe mobil cat si pe desktop. Nonfunctional 5 5 5 4 = 4.75

Poker planning dupa prioritate:

1.prezenta unui spatiu de stocare ce retine in timp datele senzorilor si compileaza statistici pe baza lor. Functional 3 3 3 4 = 3.25; Prioritate= 5 5 5 5 = 5

2.folosirea unui numar minim de senzori de temperatura, umiditate, lumina. Nonfunctional 1 2 1 2 = 1.50; Prioritate= 1 1 3 5 = 2.5

3. accesarea unui api extern pentru preluarea datelor metereologice. Functional 2 3 2 3 = 2.50; Prioritate= 4 5 4 4 = 4.25

4.citirea unui sensor de temperatura. Functional 2 2 2 2 = 2; Prioritate= 3 4 4 3 = 3.50

5. citirea unui sensor de umiditate. Functional 2 2 2 2 = 2; Prioritate= 3 4 4 3 = 3.50

6. citirea unui sensor de lumina. Functional 2 2 2 2 = 2; Prioritate= 3 4 4 3 = 3.50

7.citirea unui sensor de miscare. Functional 2 2 2 2 = 2; Prioritate= 3 4 4 3 = 3.50

8. performanta sistemului nu este critica. Nonfunctional 1 1 1 2 = 1.25; Prioritate= 1 1 2 2 = 1.50

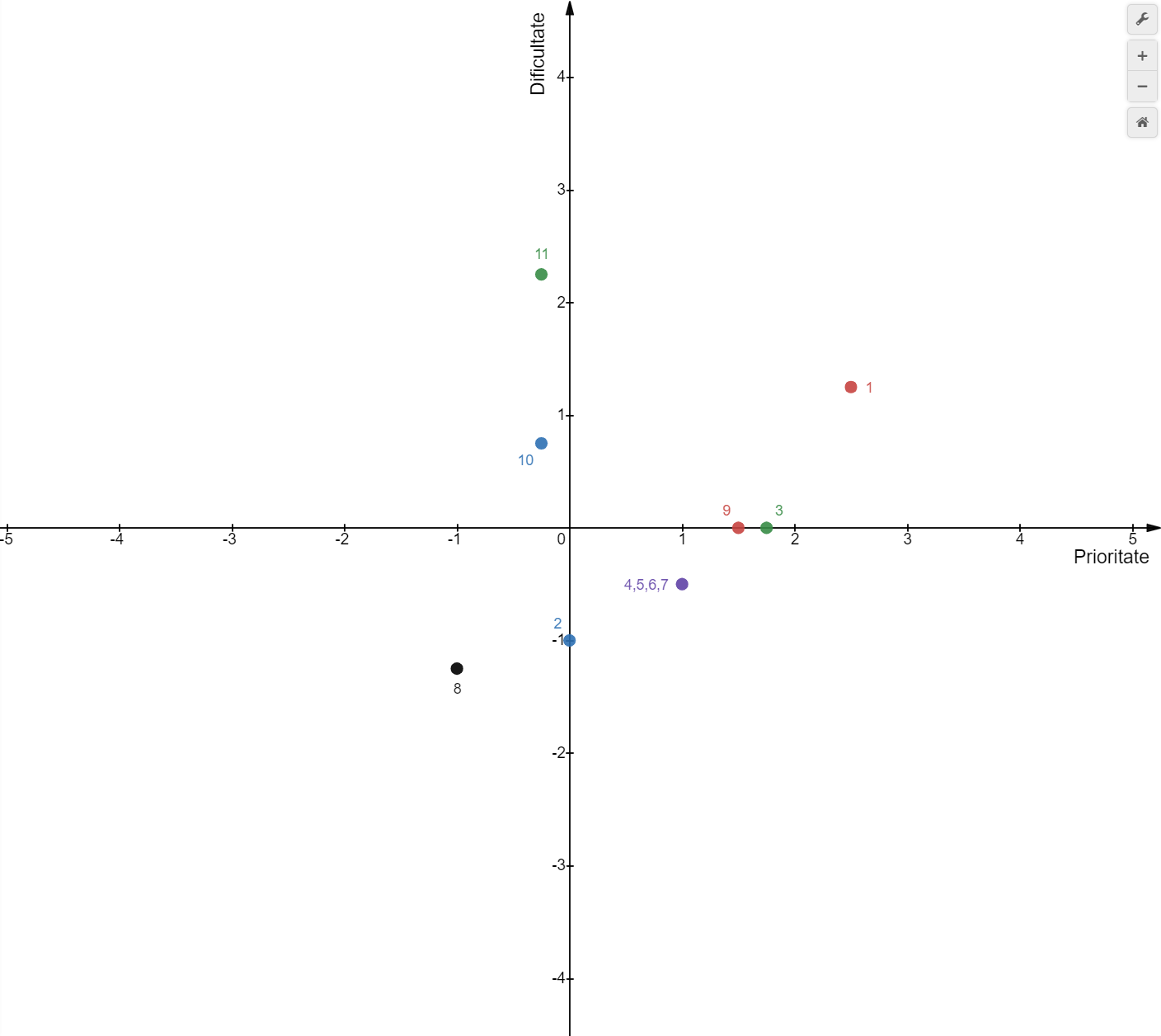
9. conexiunea la internet. Functional 4 2 1 3 =2.50; Prioritate= 3 5 5 3 = 4

10. crearea unei interfete minimaliste. Nonfunctional 3 3 3 4 =3.25; Prioritate= 3 2 2 2 = 2.25

11. adapatarea aplicatiei atat pe mobil cat si pe desktop. Nonfunctional 5 5 5 4 = 4.75; Prioritate= 1 2 3 3 = 2.25

*5. Plot the issues.*

*Realizați o axă, unde pe una dintre axe aveți dificultataea, iar pe cealaltă, prioritatea. Împărțiți axa în 4 cadrane (usor-valoros, dificil-valoros, usor-nevaloros, dificil-nevaloros). Astfel toate cerințele vor fi grupate în 4 categorii. Veți prioritiza cerințele usor-valoros, veți pune în backlog cele usor-nevaloroase și fidicil-valoroase, și în nice-to-have cele dificil nevaloros.*

  
  
*6. Check the features.*

*Verificați că adunate toate issue-urile prioritare, și din backlog, adunat constituie cele 5 feature-uri minim necesare pentru a îndeplini proiectul.   
  
7. Add technical issues.*

*Adăugați issue-uri doar în GitHub de care va trebui să vă ocupați care nu reies din cerințe (dev setup, deployment setup, etc.)*

## Alocarea rolurilor

*Fie alocați pentru fiecare cerință unul dintre membrii echipei care va realiza acea cerință, fie asociați membrii cu anumite label-uri, urmând ca respectivii membrii ai echipei să realizeze taskuri din labelul asociat.*

## Documentul de analiză va fi adăugat în GitHub-ul proiectului.