Requirement analysis

Matthijs Kotterink

# Versie beheer:

|  |  |  |
| --- | --- | --- |
| Versie nr. | Datum | Veranderingen |
| 1 | 14/11/2020 | <…> |
|  |  |  |
|  |  |  |

Inhoud

[Versie beheer: 2](#_Toc58855980)

[Inleiding 4](#_Toc58855981)

[Opstelling 4](#_Toc58855982)

[Requirements: 4](#_Toc58855983)

[Vergelijking met agile 5](#_Toc58855984)

[Requirements 5](#_Toc58855985)

[Use cases: 6](#_Toc58855986)

[Opstelling 6](#_Toc58855987)

[Aanname 6](#_Toc58855988)

[Omschrijving 6](#_Toc58855989)

[Uitzonderingen 6](#_Toc58855990)

[Resultaat 6](#_Toc58855991)

[Test Cases: 7](#_Toc58855992)

[Opstelling 7](#_Toc58855993)

[Invoer 7](#_Toc58855994)

[Verwachtte uitkomst 7](#_Toc58855995)

[TestMatrix 8](#_Toc58855996)

# Inleiding

Requirement analysis is essentieel in het maken of breken van de applicatie. In deze analyse beschrijf je de essentiele componenten van het project en bedenk je alvast wat er getest en gedaan moet worden in het project.

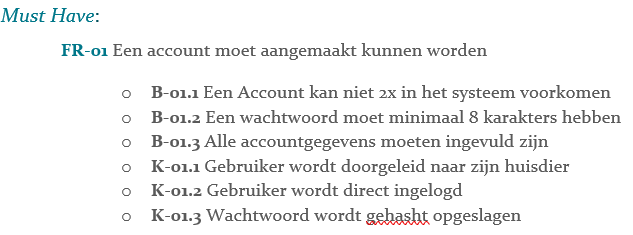
# Opstelling

## Requirements:

Bij het opstellen van de requirements begin je met het bedenken wat het project moet inhouden of wat de stakeholder aan het eind van het project verwacht, aan de hand van deze verwachtingen stel je de requirements. Deze komen te staan in de vorm van de MoSCoW methode. Deze staat voor:

1. **Must have:** Hier beschrijf je de allerbelangrijkste en essentiele onderdelen van je applicatie. Denk aan B.V. het inloggen in het systeem of het registreren van een nieuwe gebruiker.
2. **Should have:** Hier komen belangrijken aspecten alleen zijn deze niet zodanig belangrijk dat het project zou breken als het er niet in kwam.
3. **Could have:** Hier komen alleen de dingen die extra zijn mocht je team nog tijd hebben aan het eind van het project
4. **Won’t have:** Deze onderwerpen zijn wel interresant maar meer geschikt voor een soortgelijk project of een nieuwere versie.

Als je je requirements aan de hand van de MoSCoW methode hebt beschreven kun je per requirements de boudaries waar deze requirement aan moet voldoen, vervolgens beschrijf je de



In dit voorbeeld heb ik bovenaan beschreven met een ‘FR’, dit staat voor ‘Functional Requirement’ en is dus ook de requirement zelf.

Daaronder staat eerst een ‘B’, dit is voor ‘boundary’, hierbij beschrijf je eigenlijk wat de limieten moeten zijn van deze requirement. Een zo’n boundary is bijvoorbeeld dat er niet twee dezelfde account mogen voorkomen in het project. Dit is dus iets waar je expliciet op moet gaan letten in het project.

Hierna heb ik een aantal punten met de ‘K’ beschreven, dit staat voor ‘’ en zet je er eigenlijk neer om zo te kunnen zien wat er na deze requirement moet gebeuren bij het voorbeeld komt er bijvoorbeeld te staan dat het wachtwoordt van het nieuwe account gehast opgeslagen moet worden.

# Vergelijking met agile

Agile is een methode voor het oppakken voor een groepsroject, met deze methode kun je de planning beter op orde houden en een gestructureerde vorm geven aan ‘de workflow’.

De agile methodes(voornamelijk scrum) gebruiken allemaal een vorm van de basis van een analyse document.

# Requirements

De epics die je aan het begin van je stakeholder krijgt zijn eigenlijk de basis voor dit analysedocument. In sprint 0 is het belangrijk om alles op orde te krijgen zodat je goed van start kunt gaan met het project. Hierbij is het dus ideaal om de epics uit te werken in de vorm van het analyse document.

Om deze goed in het document te kunnen verwerken heb je de beschrijving nodig zowel als de prioriteit. In de meeste gevallen geeft de stakeholder de prioriteit zelf, als dit niet zo is moet je zelf gaan bedenken wat het belangrijkste is voor het project en op deze manier ook een kleine opzet maken voor de planning.

Als je de epics prioriteit hebt gegeven kun je dit vertalen naar de requirements in het analyse document, als je deze volgens de moscow methode hebt opgesteld (in volgorde van prioriteit) kun je hierbij de uitzonderingen beschrijven.

Met deze opzet heb je je epics verwerkt in het analyse document, hiermee kun je vervolgens de rest ook uitwerken voor een beter gestructureerd analyse document en een ook een direct gekoppeld document.

## Use cases:

Nadat je de requirements op hebt gesteld hebt kun je deze op een gemakkelijke manier uitwerken. Dit doe je doormiddel van use cases.

### Opstelling

Use cases worden opgesteld doormiddel van een tabel met een aantal kopjes, Eerst beschrijf je de naam van de requirement, hierbij mag je ook refereren naar de requirement zelf. Vervolgens geef je een korte omschrijving van de requirement en daaronder zet je de actors of eigenlijk de namen van de personen die er toe doen. Dit is het bovenste gedeelte van de use case en ook een beetje de inleiding, het is belangrijk dat je alle personen in de use case met actor aanspreekt en alleen bij de actors benoemt.

Nadat je deze drie kopjes Hebt gedaan kun je beginnen aan de de onderste vier die eigenlijk de requirement wat dieper beschrijven.

### Aanname

In de aanname beschijf je eigenlijk wat er van te voren moet gebeurt zijn, als je het registreren van een gebruiker je requirement is dan kun je bij aannames zetten dat de gebruiker geen account mag hebben voor het registreren, anders heeft hij deze functie namelijk niet nodig.

### Omschrijving

In de omschrijving leg je eigenlijk in detail uit wat er gebeurt moet worden in de requirement en ga je eigenlijk tussen het systeem en de actor over en weer.

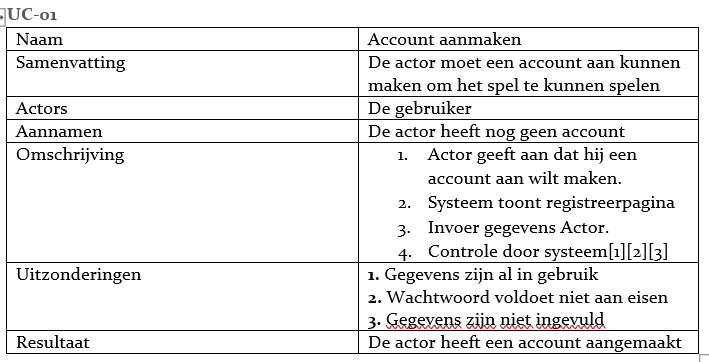
### Uitzonderingen

Bij de uitzonderingen beschrijf je wat er fout kan gaan (hier wordt ook naartoe gerefereerd bij de omschrijving). Bij het registreren als voorbeeld kunnen de gegevens al in gebruik zijn.

### Resultaat

Aan het einde zet je het resultaat neer nadat je de requirement zou hebben uitgevoerd.

Als je dezen stappen hebt voltooid zou je een resultaat moeten hebben wat lijkt op dit:



Nadat je een use case hebt gemaakt kun je doorgaan naar de test cases.

## Test Cases:

Als je klaar bent met de use cases opstellen kun je naar het laatste onderdeel van je document gaan, hierbij beschrijf je eigenlijk de basis van de tests die je gaat maken voor het testen van de applicatie. Je maakt per Use case hier altijd een test die goed gaat en een test die fout gaat.

### Opstelling

Test cases worden ook in een tabel opgesteld, hierin staan een aantal essentiele dingen die ervoor zorgen dat je je Unit tests goed kunt gaan uitvoeren. Allereerst beschrijf je de Test en Use case naam zodat je die dalijk kan gaan gebruiken in de Test matrix, vervolgens heb je de invoer en erna de verwachtte uitkomst.

### Invoer

Bij de invoer beschrijf je alle info die je gaat gebruiken om de test uit te kunnen voeren, denk bij het registreren bijvoorbeeld aan dingen zoals: Woonplaats, wachtwoord, gebruikersnaam, leeftijd. Eigenlijk alle dingen die je nodig hebt om de test succesvol uit te kunnen voeren.

### Verwachtte uitkomst

Als je een Unit test zou hebben voltooid komt daar natuurlijk een uitkomst uit, In dit laatste kopje voorspel je eigenlijk deze uitkomst. Het kan zijn dat een test faalt terwijl je dat niet voorspeld had. Dit is een makkelijke manier om hier alvast over na te gaan denken.

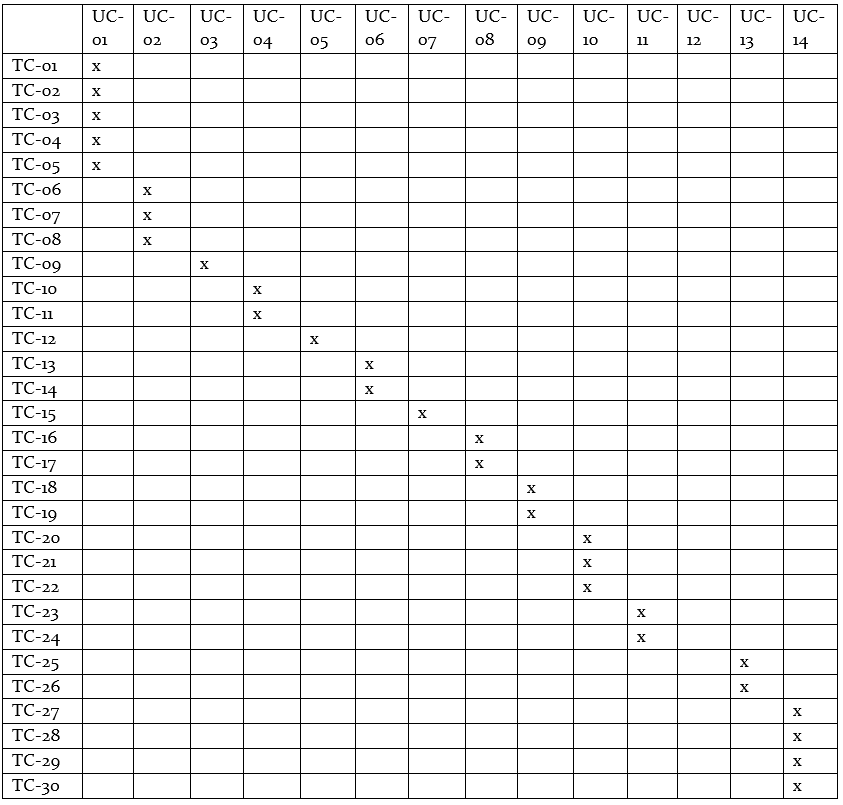


In het voorbeeld hierboven heb ik voor de eerst use case, het registreren dus. Beschreven welke gegevens iemand nodig gaat hebben en wat er moet gebeuren als er iets fout zou gaan.

## TestMatrix

Als je alle test cases hebt opgesteld is het belangrijk om in een oogopslag te kunnen zien of je alle use cases en alle boundaries hebt geraakt in het project. Hier komt de Testmatrix erg goed van pas.

In de TestMatrix laat je zien welke use case aan welke testcase is gebonden, hieronder een voorbeeld.



Hier is een voorbeeld met zon testmatrix en je kunt dan ontzettend gemakkelijk zien dat use case 12 geen tetst heeft, ideaal dus.

# Agile methodes

Deze vorm van documentatie valt perfect binnen scrum en Agile omdat je een aantal onderdelen van agile bijna letterlijk kan vertalen naar componenten uit dit researchverslag. Hieronder zal ik een aantal voorbeelden geven en hoe je dit zou kunnen doen om je werk optimaal te kunnen voorbereiden.

## Epics

Als het goed is heb je aan het begin van het project een aantal epics gekregen. Hiermee geeft de stakeholder meestal aan wat hij/zij verwacht te krijgen aan het eind van het project. De meeste epics kun je omzetten naar requirements.

Als je begint met een waarde geven aan elke epic doormiddel van bijvoorbeeld story points dan kun je deze hierna gemakkelijk converteren dat deze in de MoSCoW methode past.

Hierna kun je gaan nadenken over wat er mis kan gaan en wat de uitzonderingen kunnen zijn bij elke epic(of requirement). Hierdoor creer je voor jezelf of voor de groep een beeld wat welke prioriteit heeft en wat makkelijk / moeilijk kan zijn.

Door dit om te zetten heb je de basis gelegd voor het maken van de requirement analysis. En heb je deze ook geschreven vanuit de epics van de stakeholder, waardoor dit document ook duidelijker door de stakeholder te lezen is.