1-11-2018

Software Requirements Specification Odinido

Case study OOAD OOSE

Klaas van der Linden en Luca Hogeweide

HAN OOSE-A

1. Inhoud

[2 Inleiding 3](#_Toc528780212)

[2.1 Opdracht omschrijving 3](#_Toc528780213)

[2.2 Actoren en Characteristieken 3](#_Toc528780214)

[2.3 Operating Environment 3](#_Toc528780215)

[2.4 Design and Implementation Constraints 3](#_Toc528780216)

[2.5 Product Functionaliteiten 4](#_Toc528780217)

[*3* *Domein Model* 6](#_Toc528780218)

[Glossary 6](#_Toc528780219)

[4 Use-case Descriptions 9](#_Toc528780220)

[4.1 Registreren 9](#_Toc528780221)

[4.1.1 Fully-dressed use case description 9](#_Toc528780222)

[4.1.2 System Sequence Diagram (optional) 11](#_Toc528780223)

[4.2 Aanmaken Kennistoets 11](#_Toc528780224)

[4.2.1 Fully-dressed use case description 11](#_Toc528780225)

[4.2.2 System Sequence Diagram 12](#_Toc528780226)

[4.3 Starten kennistoets 12](#_Toc528780227)

[4.3.1 Fully-dressed use case description 12](#_Toc528780228)

[4.3.2 System Sequence Diagram 13](#_Toc528780229)

[4.4 Genereren Scoreoverzicht 13](#_Toc528780230)

[4.4.1 Fully-dressed use case description 13](#_Toc528780231)

[4.4.2 System Sequence Diagram 14](#_Toc528780232)

[4.5 Uitvoeren kennistoets 14](#_Toc528780233)

[4.5.1 Fully-dressed use case description 14](#_Toc528780234)

[4.5.2 System Sequence Diagram 15](#_Toc528780235)

[4.6 Uitvoeren Teamgevecht 15](#_Toc528780236)

[4.6.1 Fully-dressed use case description 15](#_Toc528780237)

[4.6.2 System Sequence Diagram 16](#_Toc528780238)

[5 Other functional requirements (optional) 16](#_Toc528780239)

[6 Non-functional Requirements 16](#_Toc528780240)

[6.1 Usability 16](#_Toc528780241)

[6.2 Reliability 16](#_Toc528780242)

[6.3 Supportability 16](#_Toc528780243)

[6.4 + 17](#_Toc528780244)

# Inleiding

## Opdracht omschrijving

Dit document is geschreven voor de opdracht Odinido. Deze opdracht is gegeven voor het vak OOAD, gegeven door Marco Engelbart.

De opgegeven opdracht is om “de analyse en het ontwerp voor het volledige systeem”(BRON TOEVOEGEN) te maken. De resultaten hiervan worden een SRS en SDD gezet met argumentatie voor onze keuzes.

Naast de SRS en SDD documenten moet er een kleine applicatie gemaakt worden die het uitvoeren van de kennistoets simuleert.

## Actoren en Characteristieken

Er zijn twee actoren: de docent en een student.

De docent is een actor van het beheer gedeelte van het systeem. In het beheer gedeelte zit het aanmaken van het de toetsen en het starten hiervan. In verhouding tot het systeem is de leraar de persoon die een account aanmaakt bij Odinido.

De student is een actor van het uitvoer gedeelte van het systeem. In het uitvoer gedeelte van het systeem zit het uitvoeren van de kennistoets en het uitvoeren van het teamgevecht. De student zal de gemaakte kennistoetsen vanuit het systeem gaan uitvoeren voor de docent en na de toets weer weggaan.

## Operating Environment

Het ontwerp voor de implementatie gericht op een implementatie een Java. De applicatie zal dus in het Java Runtime Environment draaien. Daarnaast moet het mogelijk zijn dat het systeem kan samenwerken met een extern betaalsysteem zodat er betalingen kunnen worden verricht.

## Design and Implementation Constraints

Aan het design en de implementatie van de applicatie zijn een aantal gegeven (BRON TOEVOEGEN) richtlijnen verbonden.

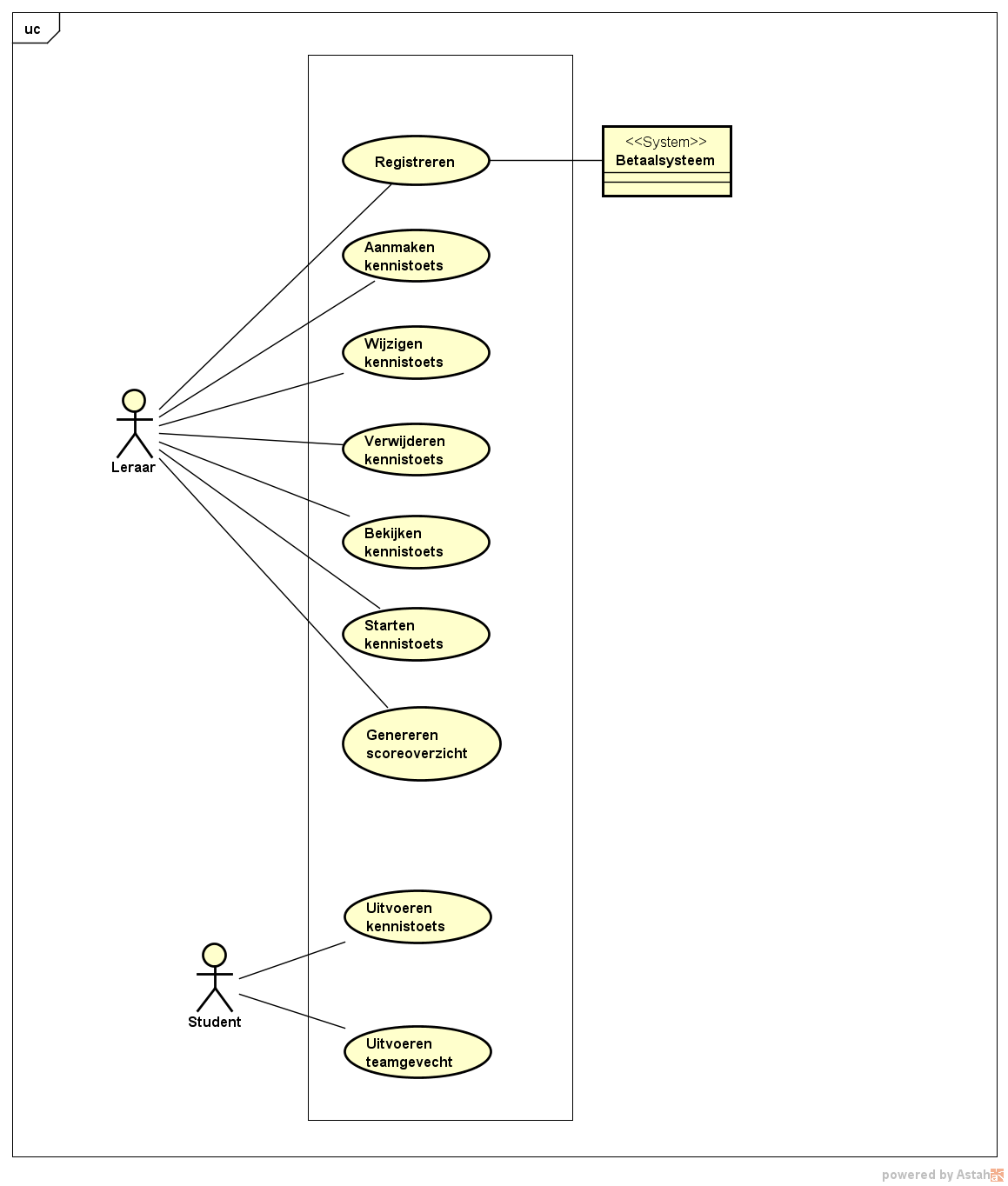
De richtlijnen voor het design zijn:

* De analyse is objectgeoriënteerd
* Het ontwerp is in UML volgens de richtlijnen van Larman

De richtlijnen van de implementatie zijn:

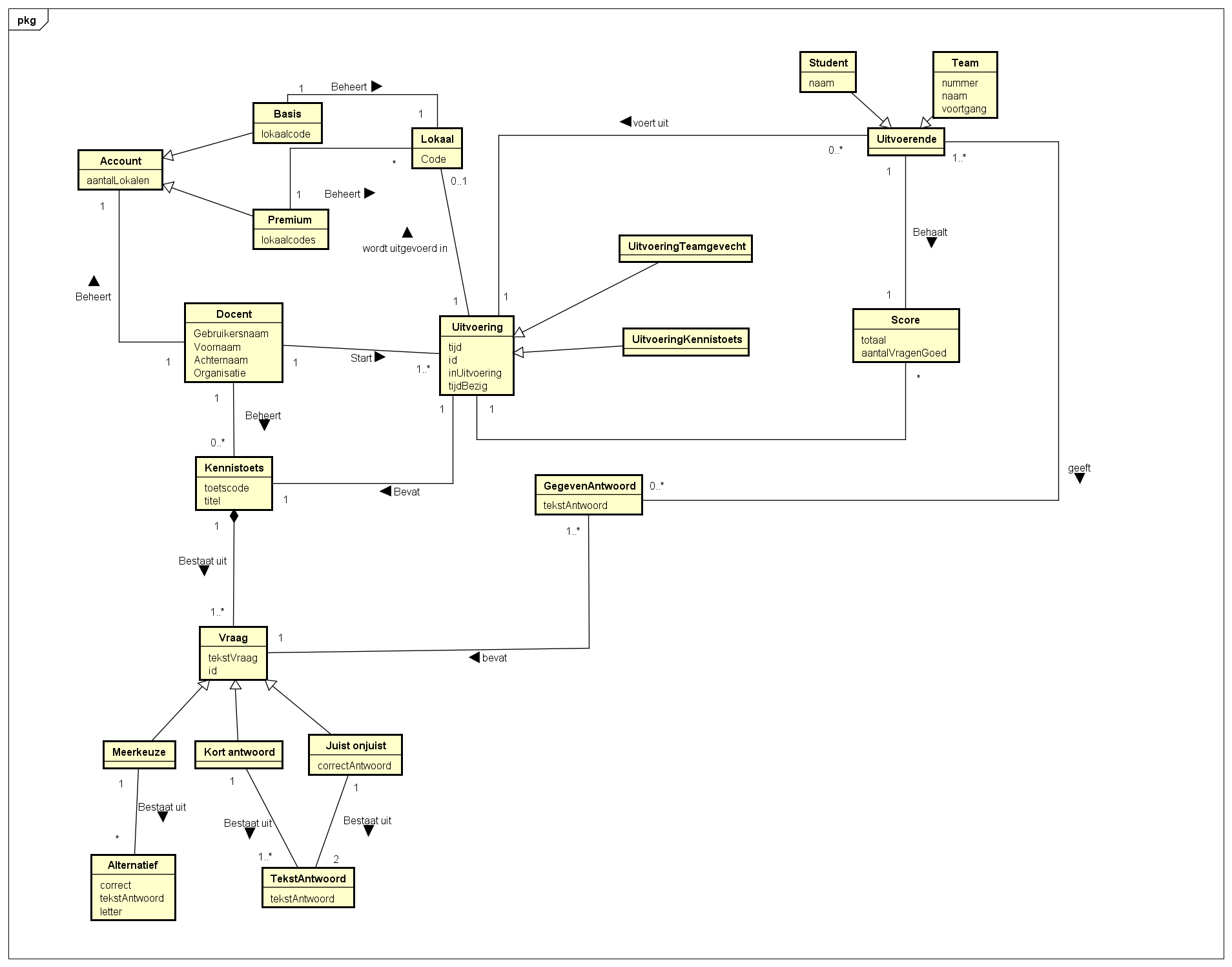
* De programmeertaal is Java

## Product Functionaliteiten



|  |  |
| --- | --- |
| **Use case** | **Brief description** |
| Registreren | Als een docent gebruik wil maken van Odinido moet hij zich eerst registreren. De docent vult zijn/haar naam, wachtwoord en werklocatie (schoolnaam) in. Het account kan het gratis basisaccount zijn of een betaald premium account.  Het bedrag van een premiumaccount is €49,99 per jaar. De betaalmethodes hierbij zijn een betaalservice of een creditcard.  Met een premiumaccount kan de docent meerdere lokalen beheren en een team gevecht aanmaken. |
| Aanmaken kennistoets | Een kennistoets wordt aangemaakt door een docent. Hierbij vult de docent een toetstitel en en genereert het systeem een toetscode. Een kennistoets bestaat uit juist of onjuist vragen of kort-antwoord vragen. De docent kiest het type vraag om vervolgens de vraag in te vullen. Bij de vraag kan de docent kiezen welk(e) antwoord(en) juist is/zijn. Het systeem voegt de vraag toe aan de kennistoets. |
| Wijzigen kennistoets | Een docent kan een kennistoets wijzigen. Hierbij kan de toetstitel worden aangepast en vragen worden gewijzigd, toegevoegd en verwijdert. |
| Verwijderen kennistoets | Een docent kan een kennistoets verwijdert. Hierbij verwijdert het systeem de kennistoets uit het Odinido systeem. |
| Bekijken kennistoets | Een docent kan een kennistoets bekijken. Hierbij toont het systeem de kennistoets die de docent heeft geselecteerd. |
| Starten kennistoets | De docent kiest een kennistoets om vervolgens deze in een lokaal te starten. De docent stelt hierbij een tijd in die aan de uitvoering van de kennistoets wordt verbonden. Het systeem start de kennistoets toont vervolgens de lokaalcode zodat studenten aan het lokaal kunnen deelnemen. |
| Genereren scoreoverzicht | De docent kies een uitvoering van een kennistoets en besluit om hiervoor een score te genereren. Het systeem genereert de score voor iedere student van de uitvoering en toont deze. |
| Uitvoeren kennistoets | Bij het uitvoeren van de kennistoets maakt de docent een lokaal aan waar de studenten zich kunnen aanmelden met hun naam. De studenten kunnen vervolgens de vragen een voor een beantwoorden. Het systeem berekent het aantal punten op basis van het aantal goed beantwoorde vragen en de tijd die de student nodig heeft om de kennistoets te doorlopen. Als alle studenten klaar zijn of de tijd om is, kan de docent een overzicht met score per student genereren. Dit gebeurt in de use case score genereren. |
| Uitvoeren teamgevecht | Als een docent een premium account heeft, kan hij een teamgevecht starten. De docent stelt een tijdsduur in voor de kennistoets. De studenten kunnen vervolgens zich aanmelden bij de kennistoets. Per team meldt één iemand zich aan als aanvoerder. Het systeem geeft het team een teamnummer. De rest van het team kan daarna deelnemen aan hun team. Als de teams zijn aangemaakt kan de docent de quiz starten zodat de teams de vragen kunnen beantwoorden. Het systeem toont de voortgang van elke groep en berekent het aantal goede vragen per groep. Op basis van het aantal vragen wordt de winnaar getoond, tenzij het aantal goede vragen gelijk is, dan is de bestede tijd doorslaggevend. |

# *Domein Model*



## Glossary

|  |  |
| --- | --- |
| **Klassenaam** | **Betekenis** |
| Docent | De docent is een domein waarin alle informatie van een docent staat. Dit domein bevat een lijst van kennistoetsen die de docent heeft aangemaakt. |
| Kennistoets | Een kennistoets is een toets die de uitvoerende kan uitvoeren. Een kennistoets hoort bij een docent en wordt toegevoegd aan een uitvoering wanneer deze uitgevoerd gaat worden. Een kennistoets bestaat uit vragen. |
| Vraag | Een vraag is de vraag waarop de uitvoerende antwoord moet gaan geven. Deze vraag zit in een kennistoets. Een vraag kan een Meerkeuze, Kort antwoord of Juist Onjuist vraag zijn |
| Meerkeuze | Een Meerkeuze is een ABCD vraag waarbij 1 van deze letters het goede antwoord is. Een Meerkeuze heeft meerdere objecten van Alternatief. |
| Alternatief | Een Alternatief is een A,B,C enz. voor een meerkeuze vraag. Een alternatief kan goed of fout zijn en heeft een bepaalde tekst die getoond wordt aan de uitvoerende |
| Kort antwoord | Een Kort antwoord is een vraag waarbij de gebruiker een kort antwoord moet invoeren in een tekstveld. Een kort antwoord vraag heeft een lijst met TekstAntwoorden die goedgekeurd worden. |
| Juist onjuist | Een Juist onjuist is een vraag waarbij de gebruiker juist of onjuist invoert. Een Juist onjuist vraag heeft een lijst met TekstAntwoord namelijk juist en onjuist. Ook heeft de vraag een correctAntwoord attribuut waarin zich het correct antwoord bevindt. |
| TekstAntwoord | Een TekstAntwoord is de tekst die getoond wordt aan de gebruiker via de User interface. In het geval van een kort antwoord heeft het een ingegeven tekst en in het geval van een Juist onjuist is het juist, onjuist of een variatie hierop. |
| GegevenAntwoord | Een GegevenAntwoord is een antwoord die gegeven is door de uitvoerende. Het GegevenAntwoord is een antwoord op een Vraag en bevat daarom een vraag. |
| Account | Een Account is een abstracte domein die bij een Docent hoort. Een docent bezit dit account. Het account heeft een attribuut met aantal lokalen, aangezien dit aantal verschilt per account type. |
| Basis | Basis is de gratis versie van de Odinido applicatie. Dit account type bevat 1 lokaal |
| Premium | Premium is de premium versie van de Odinido applicatie. Dit account type bevat 8 lokalen |
| Lokaal | Een Lokaal is de plek waar een uitvoering zich afspeelt. Een lokaal heeft een code waarmee die zich kan identificeren |
| Uitvoering | Een Uitvoering is het domein waar een kennistoets wordt uitgevoerd. In de Uitvoering zit een tijd, de maximale duur van een kennistoets, een id, om zichzelf te kunnen identificeren, een inUitvoering, een ja/nee om te kijken of het al begonnen is of niet, en een tijdBezig, hoelang is het de uitvoering bezig.  Een Uitvoering bevat ook een lijst met scores van de uitvoerende.  Een uitvoering wordt uitgevoerd door een uitvoerende en wordt beheerd door een Docent |
| UitvoeringTeamgevecht | Een UitvoeringTeamgevecht is een versie van een Uitvoering. De UitvoeringTeamgevecht gaat over een Teamgevecht waarbij een team zich aanmeld en er een scorebord wordt weergeven tijdens de uitvoering |
| UitvoeringKennistoets | Een UitvoeringKennistoets is een versie van Uitvoering. De UitvoeringKennistoets gaat over het individueel uitvoeren van een kennistoets. |
| Uitvoerende | De klasse Uitvoerende is een klasse die de uitvoering gaat uitvoeren. De actor student zal zich na het aanmelden hier bevinden. De Uitvoerende geeft antwoord aan een Uitvoering via het GegevenAntwoord |
| Student | Een Student is een Uitvoerende. De Student heeft alleen een naam |
| Team | Een Team is een Uitvoerende. Een team heeft een id om zich mee te identificeren, een naam om te tonen en een voortgang om aan te geven |
| Score | Een Uitvoerende heeft een score behaald die aangeeft hoeveel goede antwoorden er gegeven zijn |

# Use-case Descriptions

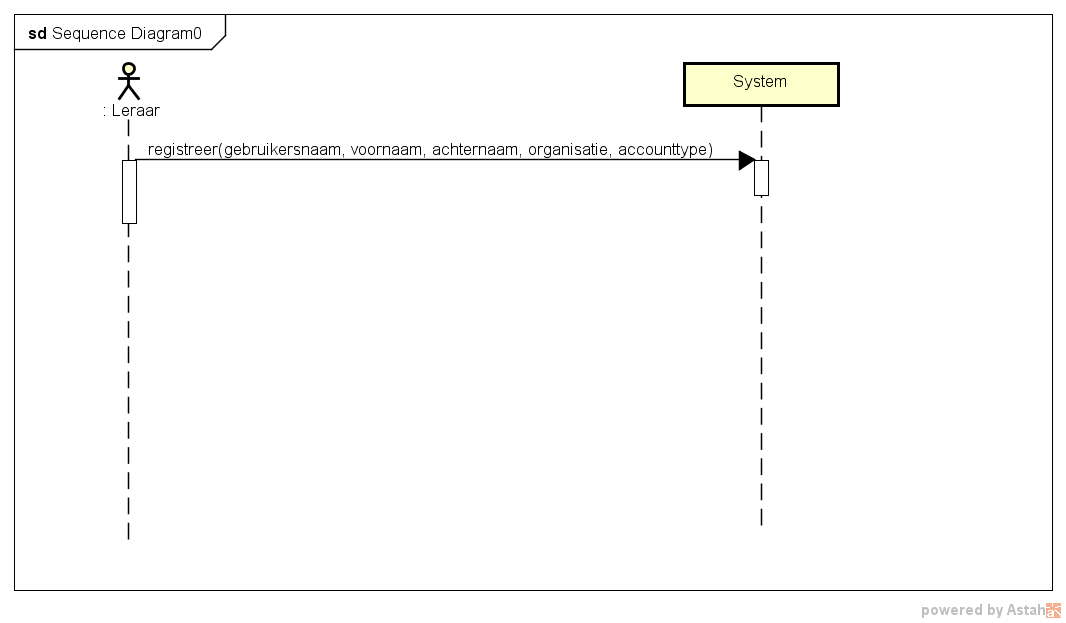
Hieronder volgende SSD’s en Fully-dressed use case beschrijvingen van de use cases. Van de use cases zijn wijzigen, verwijderen en bekijken kennistoets niet uitgewerkt gezien de eenvoudigheid en vanzelfsprekendheid van de use cases.

## Registreren

### Fully-dressed use case description

|  |  |
| --- | --- |
| **Primary actor:** Docent | |
| **Stakeholders and Interests:**   * Bank * Odinido | |
| **Brief description:**  Als een docent gebruik wil maken van Odinido moet hij zich eerst registreren. De docent vult zijn/haar naam, wachtwoord en werklocatie (schoolnaam) in. Het account kan het gratis basisaccount zijn of een betaald premium account.  Het bedrag van een premiumaccount is €49,99 per jaar. De betaalmethodes hierbij zijn een betaalservice of een creditcard.  Met een premiumaccount kan de docent meerdere lokalen beheren en een team gevecht aanmaken. | |
| **Preconditions:**   * Een docent werkt bij een school/organisatie | |
| **Postconditions (Success Guarantee):**   * Er is een account aangemaakt * Het systeem toont dat de betaling is geslaagd | |
| **Main Success Scenario (Basic Flow):** | |
| **Actor Action** | **System Responsibility** |
| 1) De docent geeft aan dat hij wil registreren | 2) De website toont de registratiepagina |
| 3) De docent vult zijn/haar persoonlijke gegevens in |  |
| 4) De docent geeft aan bij welke school/organistatie hij/zij werkt |  |
| 5) De docent kiest het accounttype premium |  |
| 6) De docent kiest de betaalmethode online betaalservice | 7) het systeem verwijst browser naar de online betaalservice |
| 8) de docent betaald | 8) het systeem verwerkt de betaling |
|  | 9) Het systeem toont de ‘geslaagd’ pagina |
| **Extensions (Alternative Flow):** | |
| 5a) De docent kiest het accounttype basis   1. Het systeem verwerkt de accountcreatie 2. Ga verder bij stap 9 |  |
|  |  |
| 6b) De docent kiest de betaalmethode creditcard   1. De docent voert het creditcardnummer in 2. Het systeem checkt de creditcard met de credicard validator 3. Ga verder bij stap 8 |  |
|  |  |
|  | 8c) De betaling kan niet worden verwerkt   1. Het systeem toont “Kon betaling niet verwerken, probeer later opnieuw” |

### System Sequence Diagram

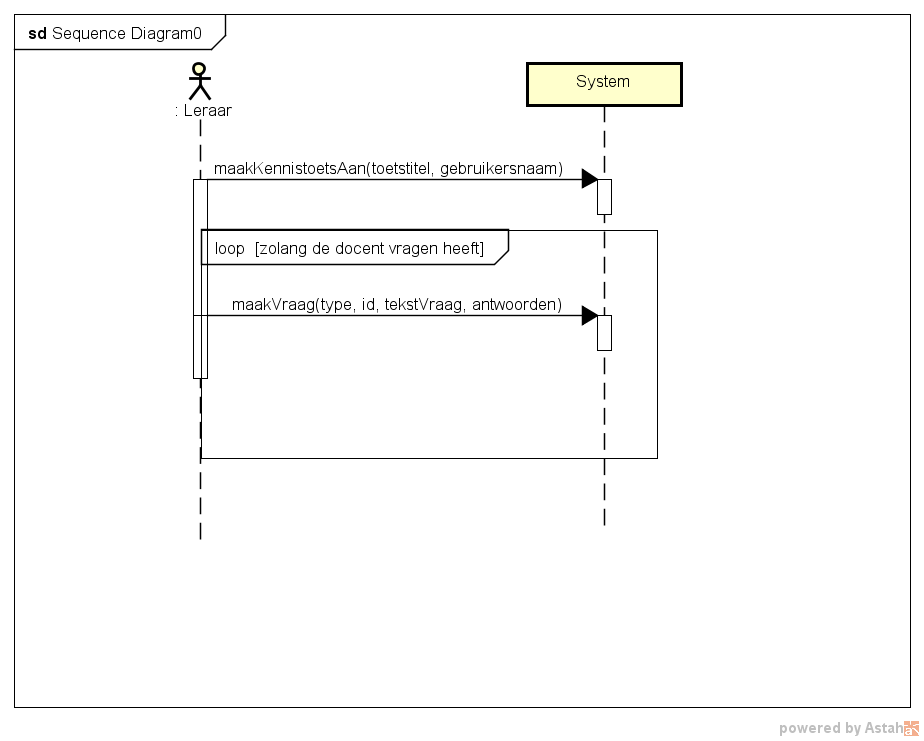


## Aanmaken Kennistoets

### Fully-dressed use case description

|  |  |
| --- | --- |
| **Primary actor:** Docent | |
| **Stakeholders and Interests:** Docent, Student | |
| **Brief description:**  Een kennistoets wordt aangemaakt door een docent. Hierbij vult de docent een toetstitel en en genereert het systeem een toetscode. Een kennistoets bestaat uit juist of onjuist vragen of kort-antwoord vragen. De docent kiest het type vraag om vervolgens de vraag in te vullen. Bij de vraag kan de docent kiezen welk(e) antwoord(en) juist is/zijn. Het systeem voegt de vraag toe aan de kennistoets. | |
| **Preconditions:**   * De docent heeft een account bij Odinido en is hiermee ingelogd | |
| **Postconditions (Success Guarantee):**   * Het systeem heeft de kennistoets toegevoegd aan het account van de docent | |
| **Main Success Scenario (Basic Flow):** | |
| **Actor Action** | **System Responsibility** |
| 1. De docent kiest om een nieuwe kennistoets aan te maken | 1. Het systeem vraagt om een toetstitel in te vullen |
| 1. De docent vult een toetstitel in |  |
| 1. De docent kiest er voor om een vraag toe te voegen | 1. Het systeem toont een overzicht met type vragen |
| 1. De docent kiest als type vraag de meerkeuzevraag en vult de vraag in |  |
| 1. De docent vult de antwoorden bij de vraag in |  |
| 1. De docent selecteer het juiste antwoord bij de vraag |  |
| Stappen 4 tot en met 8 worden herhaald tot de docent klaar is met vragen toevoegen |  |
| 1. De docent kiest er voor om de toets toe te voegen | 1. Het systeem genereert een toetscode en voegt de toets toe aan het account van de docent |
|  |  |
| **Extensions (Alternative Flow):** | |
| 6A De docent kiest als type vraag de juist onjuist vraag en vult de vraag in   1. De docent selecteer het juiste antwoord bij de vraag 2. Ga terug naar stap 4 indien alle vragen nog niet zijn ingevuld, ga anders verder bij stap 9 |  |
|  |  |
| 6B De docent kiest als type vraag de kort-antwoord vraag   1. De docent vult in welke antwoorden juist zijn 2. Ga terug naar stap 4 indien alle vragen nog niet zijn ingevuld, ga anders verder bij stap 9 |  |

### System Sequence Diagram

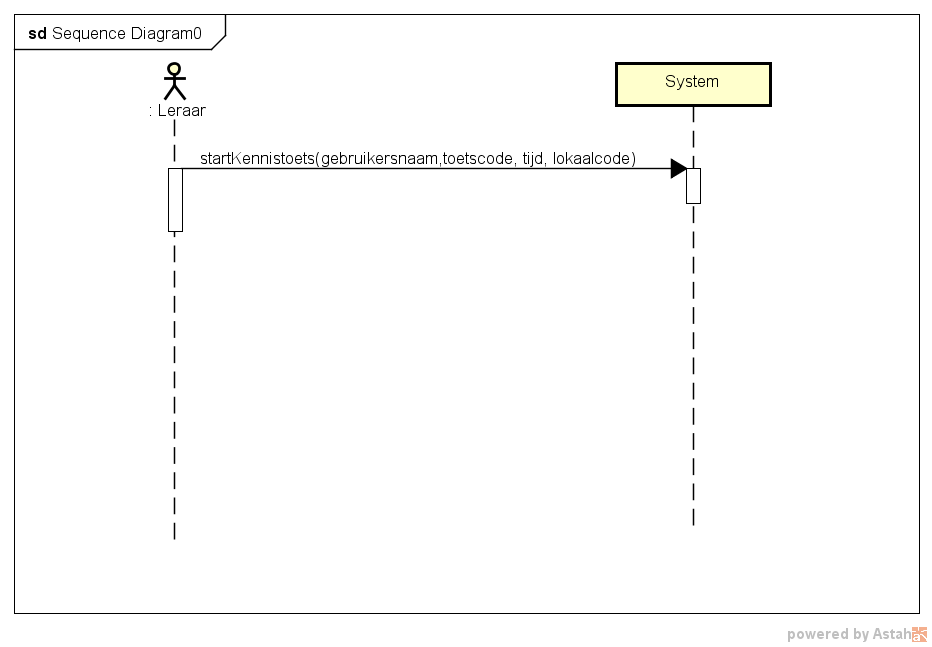


## Starten kennistoets

### Fully-dressed use case description

|  |  |
| --- | --- |
| **Primary actor:** Docent | |
| **Stakeholders and Interests:** Docent, Student | |
| **Brief description:**  De docent kiest een kennistoets om vervolgens deze in een lokaal te starten. De docent stelt hierbij een tijd in die aan de uitvoering van de kennistoets wordt verbonden. Het systeem start de kennistoets toont vervolgens de lokaalcode zodat studenten aan het lokaal kunnen deelnemen. | |
| **Preconditions:**   * De docent heeft een account bij Odinido. * De docent heeft minimaal een kennistoets op zijn account aangemaakt * Er is een nog niet gebruikte lokaalcode aanwezig | |
| **Postconditions (Success Guarantee):**   * De kennistoets is gestart in een lokaal met bijbehorende lokaalcode | |
| **Main Success Scenario (Basic Flow):** | |
| **Actor Action** | **System Responsibility** |
| 1. De docent kiest ervoor om een kennistoets te starten | 1. Het systeem toont een overzicht met kennistoetsen die de docent heeft aangemaakt. |
| 1. De docent kiest een kennistoets | 1. Het systeem maakt een uitvoering aan en voegt de kennistoets eraan toe |
| 1. De docent voert een tijd in | 1. Het systeem stelt de tijd in bij de uitvoering |
| 1. De docent kiest om de kennistoets te starten | 1. Het systeem start de toets in een lokaal en toont de lokaalcode (Basis) |
| **Extensions (Alternative Flow):** | |
|  | 8A Het systeem toont een overzicht met lokalen (Premium)   1. De docent kiest een lokaal om de toets in te starten 2. Het systeem toont een lokaal code   Stappen 1 + 2 worden herhaald zolang de docent lokalen wil openen |
|  |  |
|  | 8B Het systeem geeft een melding dat het maximum aantal lokalen al gestart is   1. Het starten van een kennistoets wordt geannuleerd. |

### 4.3.2 System Sequence Diagram

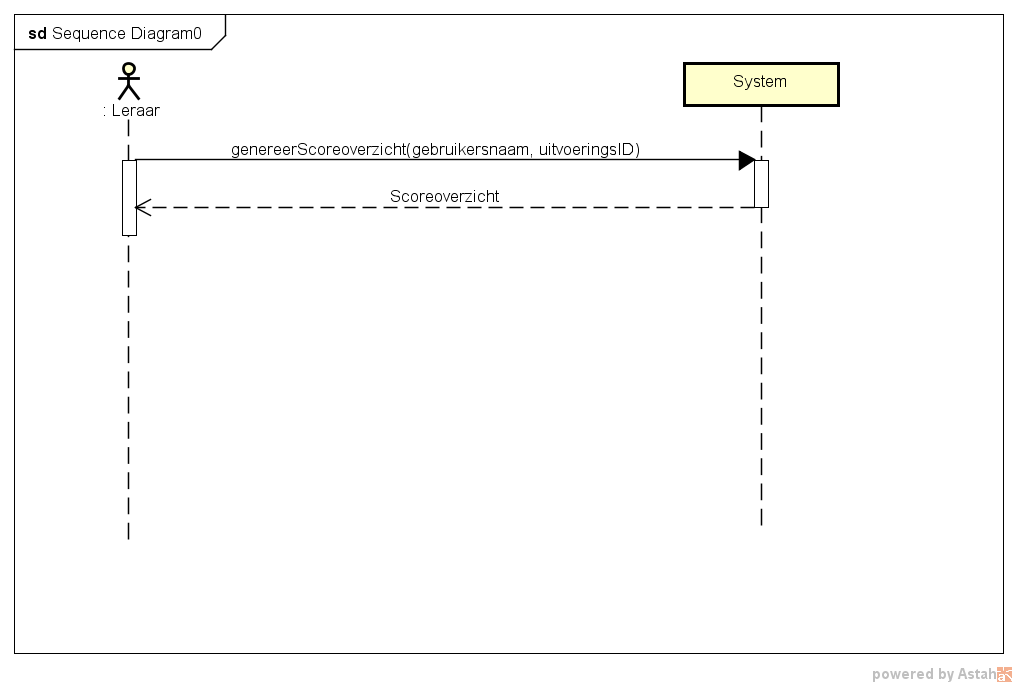


## Genereren Scoreoverzicht

### Fully-dressed use case description

|  |  |
| --- | --- |
| **Primary actor:** Docent | |
| **Stakeholders and Interests:** Student | |
| **Brief description:**  De docent kies een uitvoering van een kennistoets en besluit om hiervoor een score te genereren. Het systeem genereert de score voor iedere student van de uitvoering en toont deze. | |
| **Preconditions:**   * De uitvoering van de kennistoets in het lokaal is afgerond | |
| **Postconditions (Success Guarantee):**   * Het systeem toont de gegenereerde score | |
| **Main Success Scenario (Basic Flow):** | |
| **Actor Action** | **System Responsibility** |
| 1. De docent kiest ervoor om de score van de uitgevoerde kennistoets te genereren | 1. Het systeem genereert de score van alle studenten |
|  | 1. Het systeem toont de score van alle studenten |
| **Extensions (Alternative Flow):** | |
|  |  |
|  |  |

### System Sequence Diagram

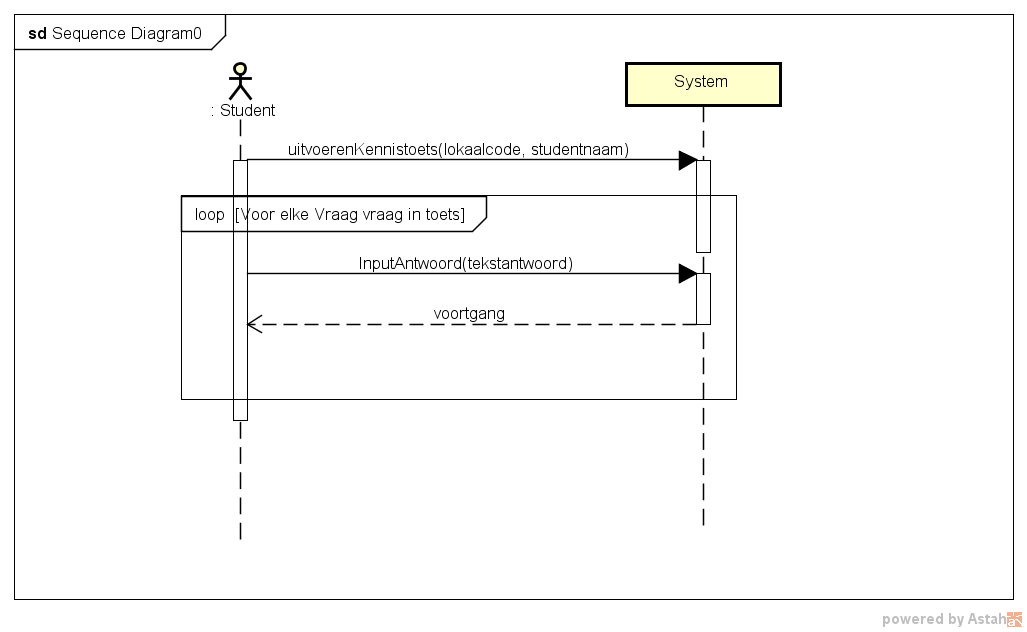


## Uitvoeren kennistoets

### Fully-dressed use case description

|  |  |
| --- | --- |
| **Primary actor:** Student | |
| **Stakeholders and Interests:** Docent | |
| **Brief description:**  Bij het uitvoeren van de kennistoets maakt de docent een lokaal aan waar de studenten zich kunnen aanmelden met hun naam. De studenten kunnen vervolgens de vragen een voor een beantwoorden. Het systeem berekent het aantal punten op basis van het aantal goed beantwoorde vragen en de tijd die de student nodig heeft om de kennistoets te doorlopen. Als alle studenten klaar zijn of de tijd om is, kan de docent een overzicht met score per student genereren. Dit gebeurt in de use case score genereren. | |
| **Preconditions:**   * De docent heeft een kennistoets gestart in een lokaal met bijbehorende lokaalcode | |
| **Postconditions (Success Guarantee):**   * Het systeem heeft geregistreerd dat de student klaar is met het uitvoeren van de toets | |
| **Main Success Scenario (Basic Flow):** | |
| **Actor Action** | **System Responsibility** |
| 1. De student meldt zich aan bij het lokaal door zijn naam in te vullen | 1. Het systeem vraagt om de toets te starten |
| 1. De student bevestigt dat hij de toets wil starten | 1. Het systeem toont een vraag |
| 1. De student beantwoordt de vraag | 1. Het system slaat het gegeven antwoord op |
| 1. De student navigeert naar de volgende vraag |  |
| Stappen 5, 6 en 7 worden herhaald totdat de student alle vragen heeft beantwoord |  |
|  | 1. Het systeem toont dat alle vragen zijn beantwoord |
| 1. De student geeft aan dat hij klaar is met het uitvoeren van de kennistoets | 1. Het systeem registreert dat de student klaar is met het uitvoeren van de toets |
|  |  |
| **Extensions (Alternative Flow):** | |
| 5A De student navigeert een vraag verder   1. Ga verder bij stap 4 |  |
|  |  |
| 5B De student navigeert een vraag terug   1. Ga verder bij stap 4 |  |
|  |  |
| 7A De student navigeert een vraag terug   1. Ga verder bij stap 4 |  |
|  |  |
|  | 8A Het systeem toont dat de tijd om is   1. Ga verder bij stap 9 |
|  |  |
| 9B De student navigeert terug naar de vorige vraag   1. Ga verder bij stap 4 |  |

### System Sequence Diagram

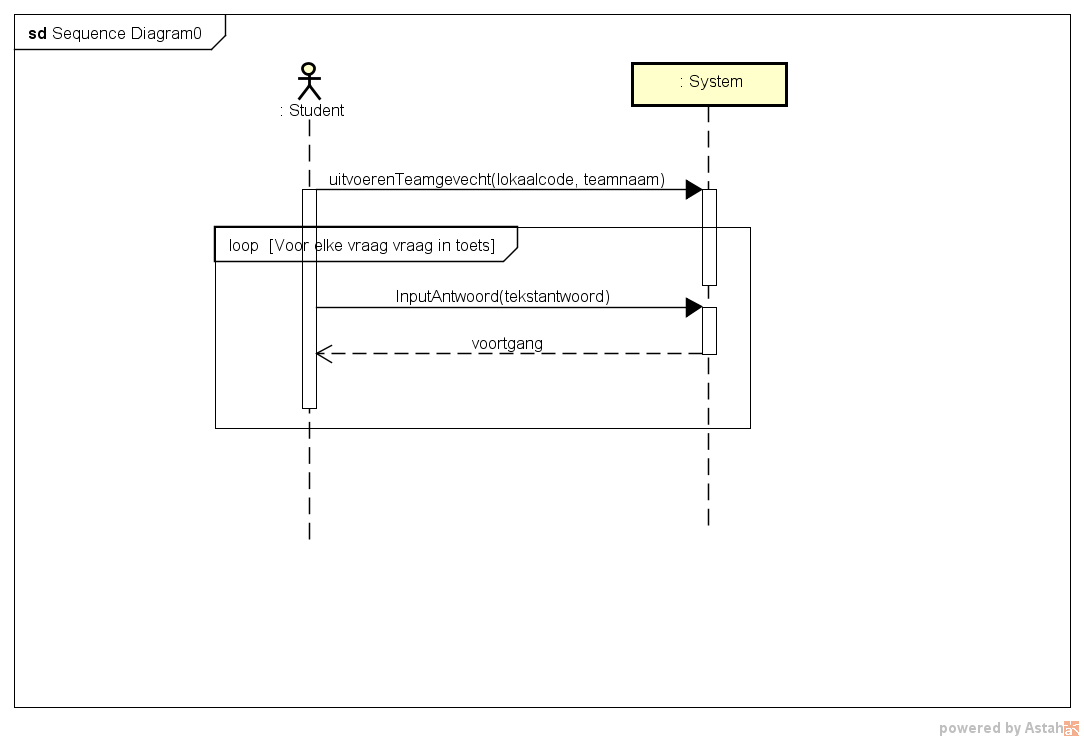


## 4.6 Uitvoeren Teamgevecht

### 4.6.1 Fully-dressed use case description

|  |  |
| --- | --- |
| **Primary actor:** Student | |
| **Stakeholders and Interests:**   * Docent | |
| **Brief description:**  Als een docent een premium account heeft, kan hij een teamgevecht starten. De docent stelt een tijdsduur in voor de kennistoets. De studenten kunnen vervolgens zich aanmelden bij de kennistoets. Per team meldt één iemand zich aan als aanvoerder. Het systeem geeft het team een teamnummer. De rest van het team kan daarna deelnemen aan hun team. Als de teams zijn aangemaakt kan de docent de quiz starten zodat de teams de vragen kunnen beantwoorden. Het systeem toont de voortgang van elke groep en berekent het aantal goede vragen per groep. Op basis van het aantal vragen wordt de winnaar getoond, tenzij het aantal goede vragen gelijk is, dan is de bestede tijd doorslaggevend. | |
| **Preconditions:**   * De docent heeft een kennistoets aangemaakt * De docent heeft een premiumaccount * Er is 1 teamleider per team gekozen | |
| **Postconditions (Success Guarantee):**   * Het Systeem heeft geregistreerd dat het team klaar is met het uitvoeren van het teamgevecht. | |
| **Main Success Scenario (Basic Flow):** | |
| **Actor Action** | **System Responsibility** |
| 1) De docent geeft aan dat hij/zij een kennistoets wil starten | 2) Het systeem toont de gemaakte kennistoetsen |
| 3) De docent kiest een kennistoets |  |
| 4) De docent stelt een tijd in | 5) Het systeem toont het aanmeld scherm met de logincode |
| 6) de gekozen teamleider logt in met de logincode | 7) Het systeem geeft een teamnummer aan de teamleider |
| 8) De gekozen teamleider vult de vraag in | 9) Het systeem toont de huidige stand van punten |
| Voer stap 8 en 9 uit tot alle vragen zijn beantwoord |  |
|  | 10) Het systeem registreert dat het team klaar is met het uitvoeren van het teamgevecht |
|  |  |
| **Extensions (Alternative Flow):** | |
| 8b) de gekozen teamleider vult niet alle vragen binnen de tijd in   1. Ga verder bij vraag 10 |  |

### 4.6.2 System Sequence Diagram



# Other functional requirements

Hier staan alle functionele eisen die zijn verkregen door FURPS te gebruiken. Onder de functionele eisen valt alles wat onder de F van FURPS valt.

|  |  |
| --- | --- |
| Code | Description |
| FR1 | Er moeten verschillende soorten vragen aangemaakt kunnen worden |
| FR2 | Er zijn twee typen accounts (Basis en Premium) |
| FR3 | Er kunnen tot 50 leerlingen deelnemen aan een lokaal |

# Non-functional Requirements

Hier staan alle niet functionele eisen. Dit zijn alle overige letters van FURPS naast de F.

## Usability

|  |  |
| --- | --- |
| Code | Description |
| NFR1 | Studenten moeten eenvoudig door de vragen kunnen navigeren. |

## Reliability

|  |  |
| --- | --- |
| Code | Description |
| NFR3 | Creditcards moeten gevalideerd worden door een creditcard validator |

## Supportability

|  |  |
| --- | --- |
| Code | Description |
| NFR4 | In de toekomst moet het mogelijk zijn om het systeem in andere talen (bijv. Engels, Duits, Frans en Spaans) op de markt te brengen |
| NFR5 | In de toekomst moet het mogelijk zijn om gemakkelijk over te stappen naar een andere punten systematiek |

## +

|  |  |
| --- | --- |
| Code | Description |
| NFR6 | Het systeem moeten kunnen communiceren met het betaalsysteem |
| NFR7 | Het systeem moet werken op alle gangbare devices1 |