Software Requirements Specification

Provo

Bosman, Niels // 646983
El Kaddouri, Iliass // 658025

Versie: 1.0 (12/03/2022)

Inhoudsopgave

1 Introduction

1.1 Overall description

Een algemene beschrijving van de te ontwikkelen opdracht staat beschreven in de Case Study OOAD: Provo.

Het doel van dit project is om kennistoetsen voor studenten digitaal te laten plaatsvinden.

De voordelen van het digitaal laten plaatsvinden van kennistoetsen zijn groot. Om te beginnen is het erg goed voor het milieu[1], omdat er erg veel papier en inkt bespaard wordt bij zowel het afnemen als het nakijken van de toets. Bovendien scheelt het de docent veel tijd, omdat hij/zij minder tijd bezig is met administratieve werkzaamheden, zoals maar niet afzonderlijk, het nakijken van de kennistoetsen.

1.2 User Classes and Characteristics

In dit systeem zijn er twee actoren, dit is een student en een docent. Studenten kunnen in dit systeem kennistoetsen uitvoeren en inleveren. Tijdens het uitvoeren van een kennisquiz kan de student navigeren tussen de vragen en eerder beantwoorde vragen nog wijzigen.

Het doel van de docent is om er voor te zorgen dat er kennistoetsen aangemaakt worden en gestart worden. Hierbij heeft de docent de mogelijkheid om tussen een veelvoud verschillende vraagtypes te kiezen. Nadat de quiz afgelopen is kan de docent een overzicht genereren om in te zien hoe de studenten hebben gepresteerd. Om deze acties uit te kunnen voeren moet de docent zich eerst registreren. De docent kan in het systeem ook profielgegevens en eventueel een het accounttype wijzigen.

Naast deze twee actoren is er ook nog één supporting actor, de betaling provider. Via de betalingsprovider kan een docent een betaling doen om het accounttype naar Premium te wijzigen.

1.3 Operating Environment

De Provo applicatie bestaat uit drie onderdelen, een client in de vorm van een webapplicatie, een server in de vorm van een RESTful[2] web API[3] welke de client voorziet van data en een SQL database om persistentie aan het project toe te voegen. De laatste twee onderdelen draaien op dezelfde server.

De client is ontwikkelt in de de facto talen voor webapplicatie clients: HTML, CSS en JavaScript, specifieker NodeJS. Verder draait deze client op een standaard static webserver

[1] https://www.docusign.nl/blog/vijf-redenen-waarom-papierloos-werken-goed-is-voor-het-milieu
[2] https://en.wikipedia.org/wiki/Representational_state_transfer
[3] https://en.wikipedia.org/wiki/Web_API

De API is geschreven in Java 17 (LTS) in een RESTful[2] structuur en communiceert met de client via het HTTP protocol.

Het laatste onderdeel, de persistentie laag, van de Provo webapplicatie bestaat uit een MySQL Docker container[4], welke op dezelfde server, naast de web API zal draaien. De web API en de database zullen via het TCP protocol communiceren.

1.4 Design and Implementation Constraints

In dit project wordt er gebruik gemaakt van vanilla Java zonder enig gebruik van een externe frameworks en/of libraries, dit is een restrictie van de opdrachtgever. Hierdoor kost het de developer erg veel tijd om features te implementeren en de code te onderhouden. Deze limitatie is ook nadelig voor de onderhoudbaarheid van de code.

De gehele applicatie zal door de developers zelf moeten worden onderhouden, wat ook meer tijd zal kosten dan wanneer er gebruik wordt gemaakt van externe frameworks/libraries. Ook is dit nadelig voor de security van de applicatie. Het kost namelijk een stuk meer moeite om zelf security patches uit te brengen in tegenstelling tot een framework die dit van tijd tot tijd oplost en released.

1.5 Product Functions

1.5.1 Use cases in brief format

1.5.1.1 Aanmelden lokaal

Een student meldt zich aan bij een lokaal door zijn/haar studentnummer op te geven.

1.5.1.2 Uitvoeren kennistoets

Een student beantwoordt de vragen van de kennistoets een voor een. Hierbij is het mogelijk om door de vragen heen te navigeren en terug te gaan naar een eerder bekeken vraag.

1.5.1.6 Aanmaken kennistoets

Een docent maakt een kennistoets aan. Hierbij kiest de docent 6 vragen uit een lijst. De lijst bestaat uit meerkeuzevragen, kort-antwoord-vragen en juist/onjuist-vragen.

1.5.1.7 Start kennistoets

Een docent geeft in het systeem aan dat er een kennistoets gestart moet worden. Het systeem geeft aan wanneer deze begonnen is.

[4] https://hub.docker.com/_/mysql/

1.5.1.8 Genereren overzicht

Een docent merkt op dat de tijd van de kennistoets is verstreken en/of alle studenten klaar zijn met de toets doordat het systeem dit aangeeft. De docent geeft aan dat er een overzicht gegenereerd kan worden.

1.5.1.9 Registreren account

Een docent registreert zich in de applicatie door een e-mailadres, voornaam, achternaam, wachtwoord en de school/organisatie in te voeren.

1.5.1.10 Beheren account

Een docent geeft aan het systeem aan dat de profielgegevens aangepast moeten worden. Hierin heeft de docent de optie om het e-mailadres, voornaam, achternaam, wachtwoord, school/organisatie en het accounttype te wijzigen.

1.5.1.11 Verwerken betaling

Een docent kan een betaling doen aan een betalingsprovider voor het upgraden naar een Premium account.

1.5.1.12 Aanpassen profiel gegevens

Een docent heeft de mogelijkheid om zijn profielgegevens te wijzigen.

1.5.1.13 Beheren lokalen

Een docent met een premium account heeft de mogelijkheid om lokalen toe te voegen en te verwijderen. Het maximum aantal lokalen is gelimiteerd tot 8.

1.5.2 Use case diagram

Figuur 1: Use case diagram Provo applicatie.

2 Domain model

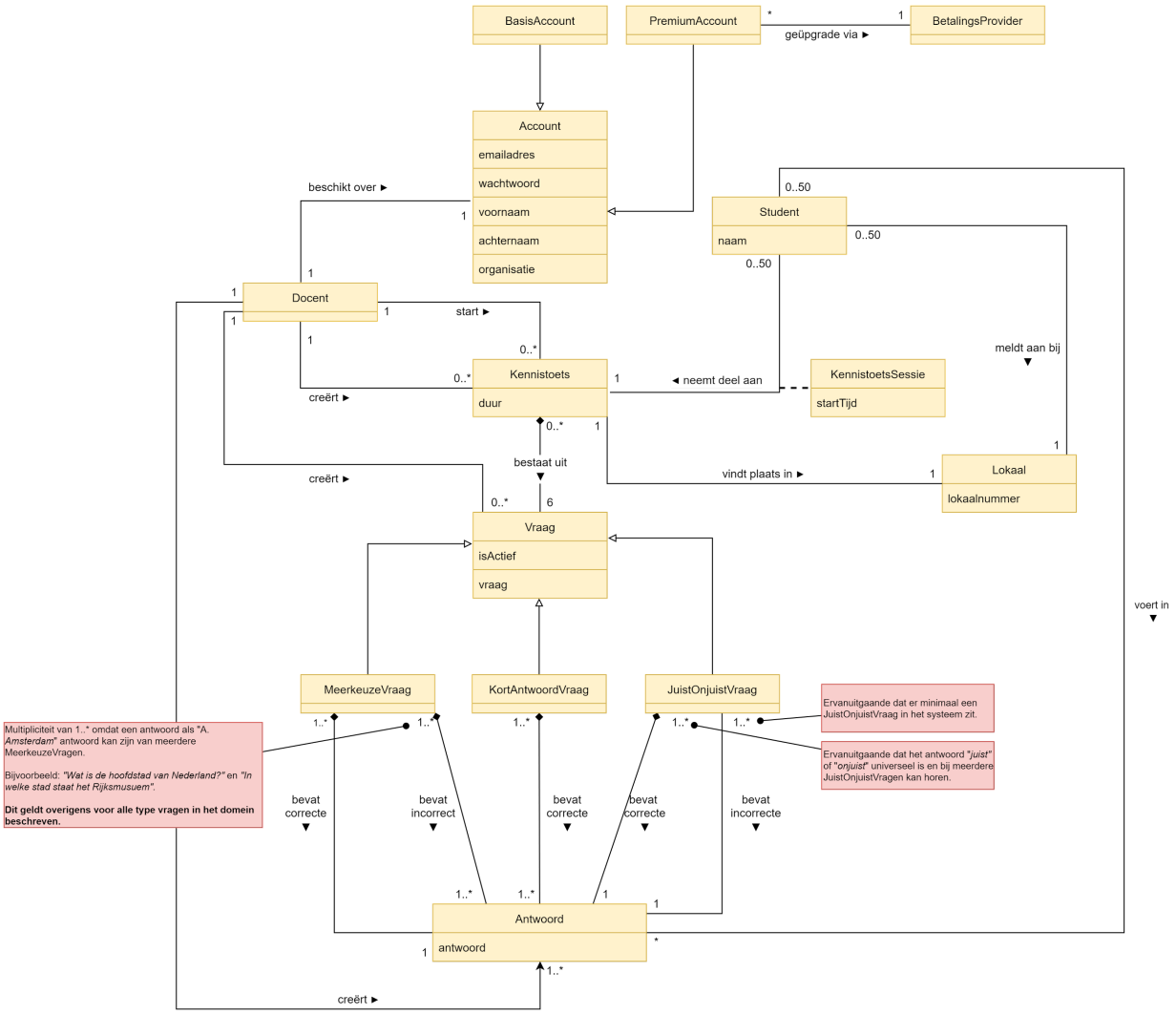
2.1 Glossary

De volgende woordenlijst is tot stand gekomen door noun-phrase-identification toe te passen op de ontvangen Casus Provo.

Zelfstandig naamwoord	Beschrijving
Docent	De maker van een kennistoets. Ook kan de docent in het systeem een kennistoets starten.
Kennistoets	Een toets die de docent kan starten en waar studenten aan deel kunnen nemen.
Vraag	Een onderdeel van een kennistoets, dat beantwoord kan worden door een student.
Meerkeuzevraag	Een specificatie van een vraag waarbij er meerdere keuzes zijn.
Juist/onjuist-vraag	Een specificatie van een vraag waarbij er twee keuzes zijn; “juist en onjuist”.
Kort-antwoord-vraag	Een specificatie van een vraag waarbij er een open antwoord ingevuld moet worden.
Antwoord	De oplossing van een vraag dat de student invult.
School/organisatie	De plek waar de docent werknemend is.
Account	De manier van een docent om zich aan het systeem te identificeren.
Lokaal	Een fysieke plek waar studenten een kennistoets kunnen maken.
Student	Een leerling die een kennistoets kan maken.
Betalingsprovider	De provider die er voor zorgt dat er betalingen in het systeem plaats kunnen vinden.
Accounttype	Een specificatie van het account. Hier is de keuze “Basis” of “Premium”, waarbij je voor premium moet betalen.
Profielgegevens	De gegevens van het account van een docent. Bestaande uit een e-mailadres, wachtwoord, voor- en achternaam.
Punten	Een score die een gebruiker toegewezen krijgt op basis van de prestaties in een kennistoets.
Overzicht	Een lijst met de resultaten van een kennistoets.

2.2 Model

Zie rode vakken voor toelichting.



Figuur 2: Domein model Provo applicatie.

3 Use-case Descriptions

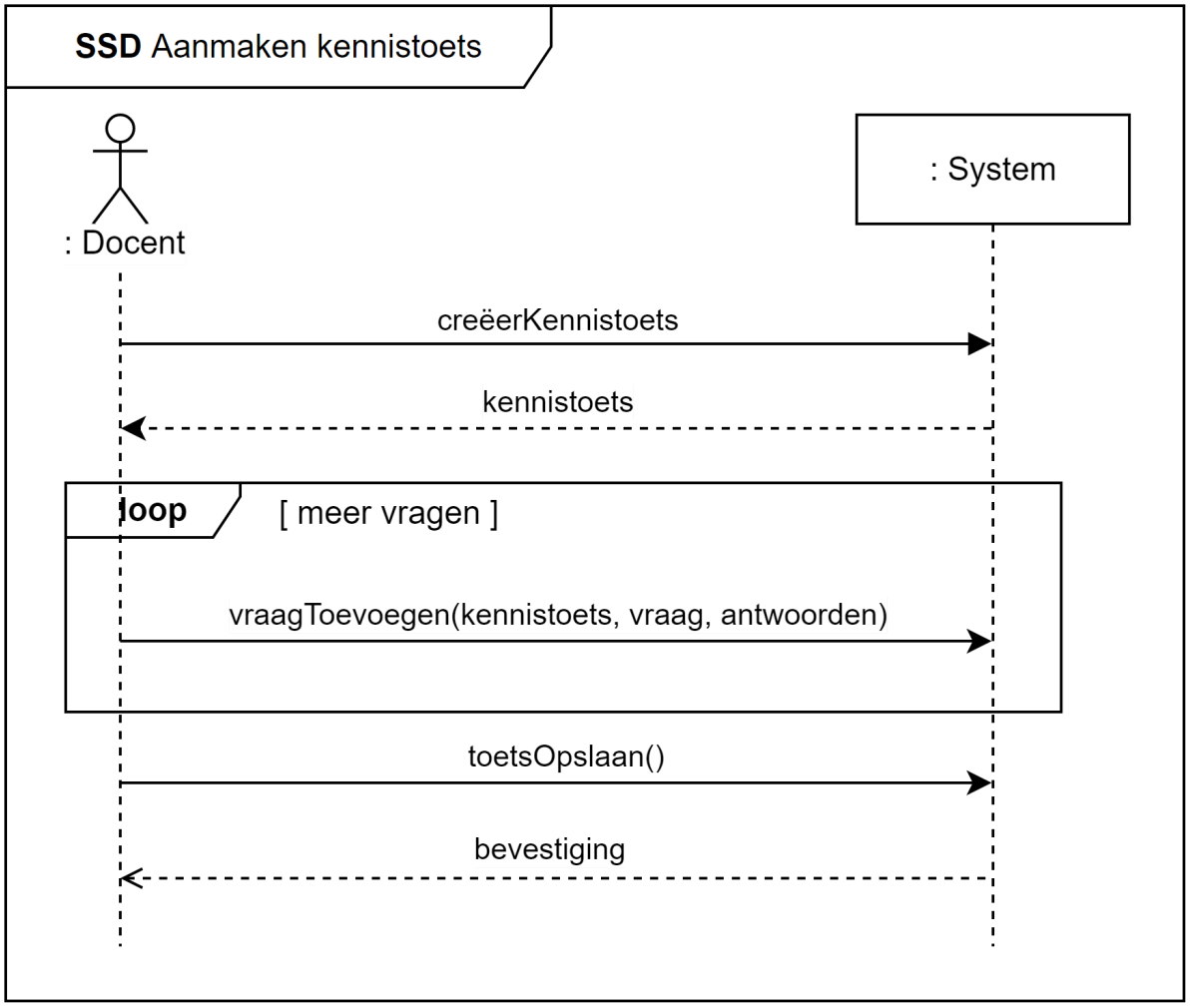
3.1 Aanmaken kennistoets

3.1.1 Fully-dressed use case description

Use case	Aanmaken kennistoets
Actor	Docent
User Goal	Het maken van een kennistoets.
Stakeholders	Student
Preconditions	1. De docent heeft zich geregistreerd en heeft een account. 2. De docent is ingelogt op zijn account.
Postconditions	1. Er is een kennistoets aan het systeem toegevoegd.
Main success Scenario	
Actor	**Systeem**
1. De docent laat het systeem weten een kennistoets te willen maken.	
	2. Het systeem geeft de mogelijkheid om vragen en bijbehorende antwoorden in te voeren.
3. De docent kiest een vraagtype en voert de vraag en bijbehoren(de) antwoord(en) in.	
Herhaling van stap 2 en 3 totdat de docent geen vragen meer in wilt vullen.	
4. De docent geeft aan klaar te zijn met invoeren.	
	5. Het systeem slaat de opgestelde kennistoets op en laat dit aan de docent weten.
Alternative Scenario 1: Een docent geeft geen vragen ingevoerd	
Bevat stap 1 t/m 2 van het main success scenario.	
4. De docent geeft aan klaar te zijn met	

invoeren zonder vragen te hebben ingevuld.	
	5. Het systeem geeft aan dat er geen vragen zijn ingevoerd.
	Het systeem gaat verder met stap 2 van main success scenario.

3.1.2 System Sequence Diagram



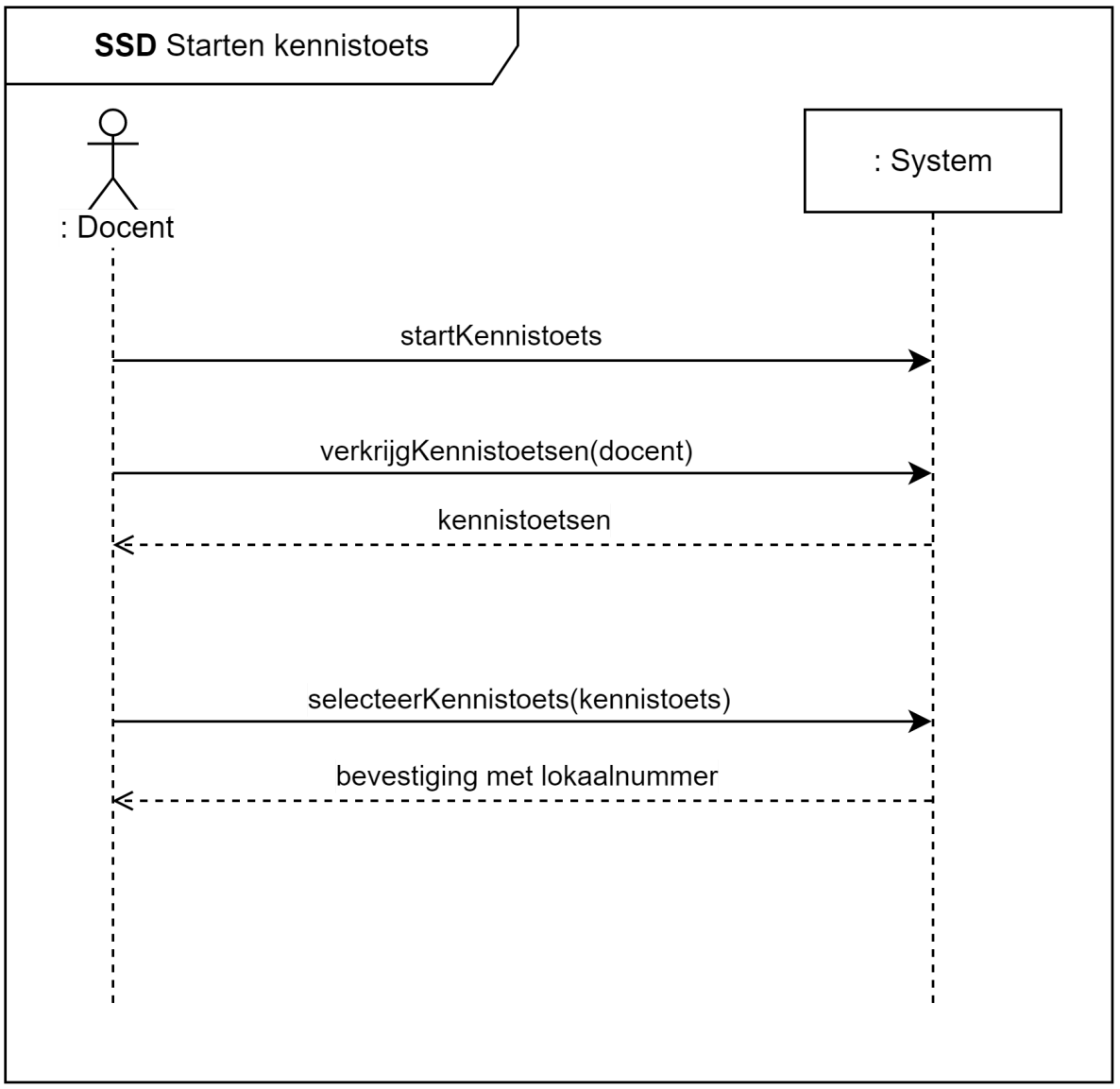
Figuur 3: System Sequence Diagram voor use case *Aanmaken kennistoets*.

3.2 Starten kennistoets

3.2.1 Fully-dressed use case description

Use case	Starten kennistoets
Actor	Docent
User Goal	Het beginnen van een kennistoets.
Stakeholders	Student
Preconditions	1. Er is tenminste één kennistoets beschikbaar om te kiezen.
Postconditions	1. Er is een sessie voor een kennistoets gestart waar studenten zich voor kunnen aanmelden.
Main success Scenario	
Actor	**Systeem**
1. De docent laat het systeem weten een kennistoets te willen starten.	
	2. Het systeem geeft de docent de mogelijkheid om een kennistoets te selecteren.
3. De docent selecteert een kennistoets en laat het systeem weten dat de kennistoets definitief kan beginnen.	
	4. Het systeem start de kennistoets sessie en toont de docent het lokaalnummer.
Alternative Scenario 1: Docent kiest zelf lokal(en)	
Bevat stap 1 van het main success scenario	
	2. Het systeem detecteert dat de docent een premium account heeft en geeft de docent de mogelijkheid lokal(en) te selecteren en de kennistoets te selecteren.
3. De docent selecteert een of meerdere lokalen, kiest een kennistoets en laat het systeem weten dat de kennistoets definitief kan beginnen.	
	Het systeem gaat verder met stap 4 van het main success scenario

3.2.2 System Sequence Diagram



Figuur 4: System Sequence Diagram voor use case *Starten kennistoets*.

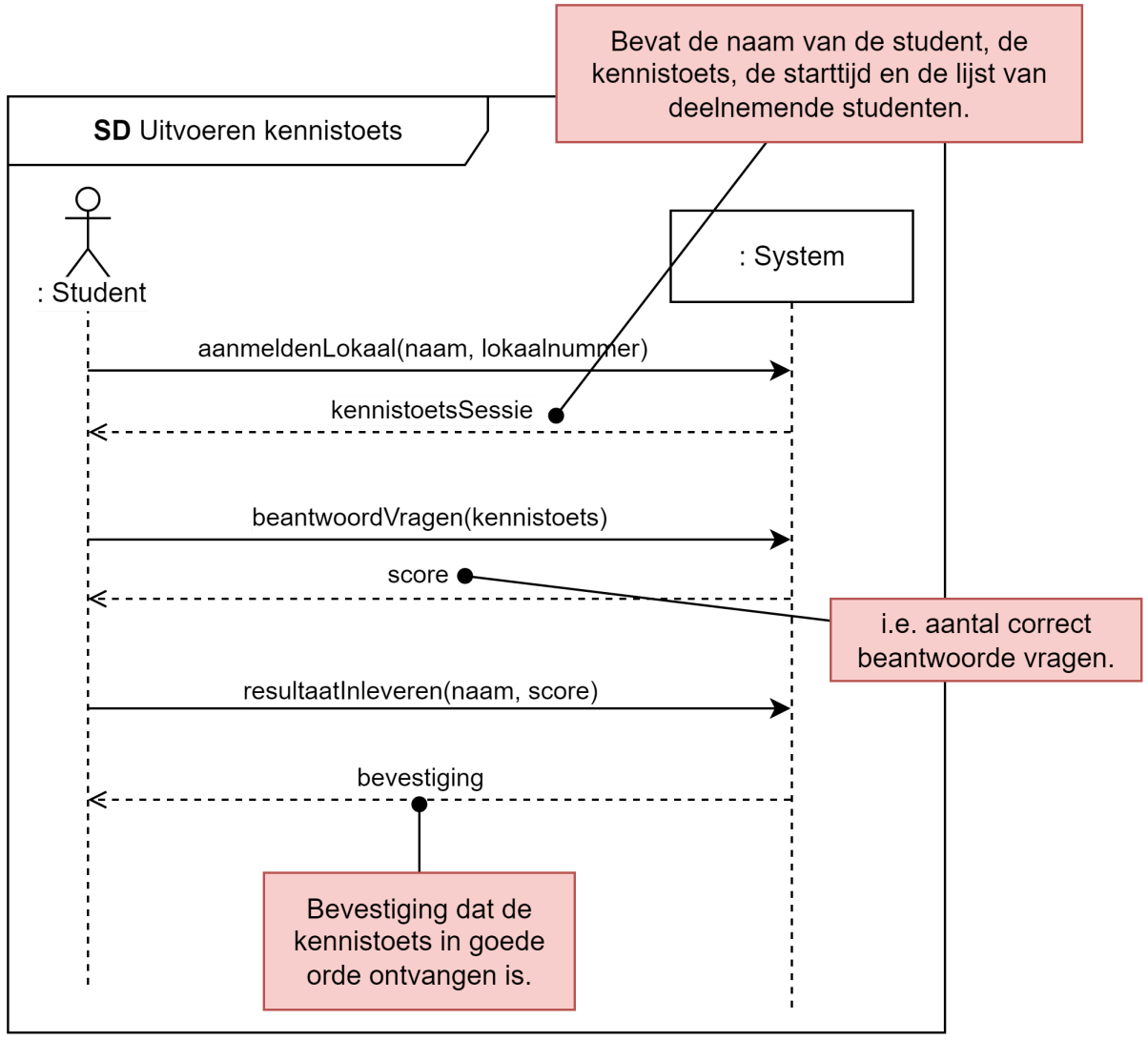
3.3 Uitvoeren kennistoets

3.3.1 Fully-dressed use case description

Use case	Uitvoeren kennistoets
Actor	Student
User Goal	Het uitvoeren van een kennistoets in een lokaal
Stakeholders	Docent
Preconditions	1. De student beschikt over een geldig lokaalnummer.
Postconditions	1. Het systeem heeft de antwoorden van de kennistoets opgeslagen.
Main success Scenario	
Actor	**Systeem**
1. De student laat het systeem weten aan de kennistoets mee te willen doen door het lokaalnummer en zijn/haar/het naam in te voeren.	
	2. Het systeem registreert dat de student aan de toets meedoet.
	3. Het systeem toont alle vragen van de kennistoets.
4. De student beantwoord de vragen die door het systeem getoond worden.	
5. De student geeft het systeem te kennen de toets in te willen leveren	
	6. Het systeem slaat de toets op en laat dit de student weten.
Alternative Scenario 1: Aanpassen antwoord	
Bevat stap 1 t/m 3 of 1 t/m 4 van het main success scenario.	
6. De student navigeert naar een vraag in de toets waarvan het antwoord aangepast moet worden	
7. De student voert een nieuw antwoord in	

bij de vraag en dient deze in.	
	Het systeem gaat verder met stap 4 van het main success scenario.
Alternative Scenario 2: Geen tijd meer over	
Bevat stap 1 t/m 5 van het main success scenario	
	6. Het systeem geeft aan de student aan dat de tijd voor de kennistoets op is en dat de student niet meer verder kan met de kennistoets.
Het systeem gaat verder met stap 6 van het main success scenario.	

3.3.2 System Sequence Diagram



Figuur 5: System Sequence Diagram voor use case *Uitvoeren kennistoets*.

3.4 Genereren overzicht

3.4.1 Fully-dressed use case description

Use case	Generen overzicht
Actor	Docent
User Goal	Het genereren van een overzicht, met per deelnemer de (totaal)score.
Stakeholders	Studenten
Preconditions	1. Alle studenten hebben de kennistoets afgerond zie 3.3.
Postconditions	1. Er is een overzicht gegenereerd.
Main success Scenario	
Actor	**Systeem**
1. De docent geeft aan het systeem aan een overzicht te willen genereren van de zojuist afgeronde toets.	
	2. Het systeem berekent de scores per student en toont een overzicht hiervan.

Deze fully dressed use case bevat geen system sequence diagram omdat deze maar bestaat uit één systeem actie.

3.5 Registreren account (Bonus)

3.5.1 Fully-dressed use case description

Use case	Registreren account
Actor	Docent
User Goal	Het registreren van een account.
Stakeholders	
Preconditions	2. De docent maakt deel uit van een school/organisatie
Postconditions	2. Het systeem heeft het account van de docent opgeslagen.

Main success Scenario	
Actor	**Systeem**
3. De docent laat aan het systeem weten te willen registreren	
	4. Het systeem toont een registratieformulier met de volgende velden: e-mailadres, wachtwoord, voor- en achternaam, organisatie naam.
5. De docent vult de alle velden in.	
	6. Het systeem slaat het account op en geeft dit te kennen aan de docent.
Alternative Scenario 1: Docent vult niet alle velden in.	
Bevat stap 1 t/m 2 van het main success criteria	
3. De docent vult niet alle velden in.	
	4. Het systeem geeft aan dat niet alle velden zijn ingevuld.
	5. Het systeem stuurt de docent terug naar stap 2 van het main success scenario.

4 Other functional requirements

4.1 Functionality

Code	Description
FR1	Een docent kan met een premium account kan tot 8 verschillende kennistoetsen aanmaken en (parallel) uitvoeren.
FR2	Bij een Basis account krijgt een docent een lokaal met een lokaal nummer toegewezen waarin hij een kennistoets kan uitvoeren.

5 Non-functional requirements

5.1 Usability

Code	Description
NFR1	Een student kan op een eenvoudige manier door de vragen te navigeren om terug te gaan naar een eerder bekeken vraag.

5.2 Performance

Code	Description
NFR2	Het systeem moet over voldoende resources beschikken om de maximale capaciteit van ten minste 1000 docenten met een Premium account te faciliteren. I.e. 8000 sessies met elk 50 leerlingen. *Notitie: Deze requirement hebben wij geredeneerd uit eigen ervaring en met informatie wat er in de casus te vinden. Dit staat **nergens** in de casus gedefinieerd.*

5.3 Supportability

Code	Description
NFR3	Er is de mogelijkheid om puntentelling flexibel te maken zodat er makkelijk overgestapt kan worden op een andere systematiek.
NFR4	Er is de mogelijkheid om in de toekomst versies in andere talen op de markt te brengen.
NFR5	Het platform dient beschikbaar te komen op de meest gangbare devices.