|  |
| --- |
| System Requirements Specification Provo\_  Case Study OOSE-OOAS |

Namen: Marnix Wildeman 658705

Max Jansen 584700

Klas: ITA-OOSE-B

Datum: 21-10-2021

Docent: Joost Kraaijeveld

Inhoudsopgave

[1 Introductie 3](#_Toc85721602)

[1.1 Opdracht Omschrijving 3](#_Toc85721603)

[1.2 Actoren en de Kenmerken 3](#_Toc85721604)

[1.3 Werkomgeving 3](#_Toc85721605)

[1.4 Ontwerp- en implementatiebeperkingen 4](#_Toc85721606)

[1.5 Product Functionaliteiten 4](#_Toc85721607)

[2 Domeinmodel 7](#_Toc85721608)

[2.1 Domeinmodel 7](#_Toc85721609)

[2.2 Glossarium 7](#_Toc85721610)

[3 Use-case omschrijvingen 9](#_Toc85721611)

[3.1 Registreren 9](#_Toc85721612)

[3.1.1 Fully-dressed use-case omschrijving 9](#_Toc85721613)

[3.1.2 System Sequence Diagram 10](#_Toc85721614)

[3.2 Starten kennistoets 11](#_Toc85721615)

[3.2.1 Fully-dressed use-case omschrijving 11](#_Toc85721616)

[3.2.2 System Sequence Diagram 12](#_Toc85721617)

[3.3 Uitvoeren kennistoets 13](#_Toc85721618)

[3.3.1 Fully-dressed use-case omschrijving 13](#_Toc85721619)

[3.3.2 System Sequence Diagram 14](#_Toc85721620)

[3.4 Genereren totaalscore 15](#_Toc85721621)

[3.4.1 Fully-dressed use-case omschrijving 15](#_Toc85721622)

[3.4.2 System Sequence Diagram 15](#_Toc85721623)

[4 Functionele eisen 16](#_Toc85721624)

[5 niet-functionele eisen 16](#_Toc85721625)

[5.1 Usability 16](#_Toc85721626)

[5.2 Supportability 16](#_Toc85721627)

[6 Bronnenlijst 17](#_Toc85721628)

# Introductie

## Opdracht Omschrijving

Educom wil een nieuw online leerplatform ontwikkelen genaamd Provo. Met Provo kan een docent kennistoetsen aanmaken en deze laten uitvoeren door de leerlingen. Voor deze opdracht zullen wij een objectgeoriënteerde analyse en -ontwerp in UML opleveren. Deze zullen wij documenteren in SRS- en SDD-documenten.

Bovenop de SRS- en SDD-documenten leveren wij een Java-applicatie op die het maken van een kennistoets simuleert. Deze zal niet volledig werken hoe het uiteindelijk zal werken. Je kunt in deze versie niet door de vragen heen lopen, maar alleen van voren naar achteren werken.

De foto’s in dit document zijn relatief klein gehouden om ruimte te besparen. Dit heeft in sommige gevallen de leesbaarheid verminderd. Zodat er niet ingezoomd hoeft te worden tijdens het lezen heeft elke foto een cijfer, dit cijfer correspondeert met een foto in het “Foto’s” mapje.

## Actoren en de Kenmerken

De twee geïdentificeerde actoren zijn:

* De docent.
* De leerling

De docent houdt zich vooral bezig in het beheer gedeelte van Provo. Een beheerder kan kennistoetsen aanmaken, aanpassen en verwijderen. Ook het starten van kennistoetsen en de totaalscores laten genereren is een taak van de beheerder. Om dit alles te kunnen doen moet een docent een account aanmaken bij Provo.

De leerling is daarentegen voornamelijk aan het werk in het uitvoer gedeelte van het systeem. Hij/zij kan de door de docent gestarte kennistoetsen uitvoeren. De leerling logt in bij het systeem met zijn/haar naam en na het uitvoeren van de toets zal de leerling weer uit het systeem gaan. De gegevens worden niet opgeslagen.

## Werkomgeving

De applicatie die wij maken zal een Java-applicatie zijn. De werkomgeving zal dus draaien in Java Runtime Enviroment. Het is de bedoeling dat Provo op de meest gangbare devices moet werken, ieder met zijn eigen besturingssystemen.

Ook is het de bedoeling dat er een link komt met een betalingsprovider zodat betalingen verricht kunnen worden.

## Ontwerp- en implementatiebeperkingen

Het ontwerp en de implementatie hebben een aantal randvoorwaarden en richtlijnen aan zich verbonden. Deze richtlijnen komen rechtsreeks uit de casus “Case Study OOAD: Provo”.

De richtlijnen met betrekking tot het ontwerp:

* De analyse moet object georiënteerd zijn.
* Het ontwerp in UML moet volgens de richtlijnen van Larman.

De richtlijnen met betrekking tot de implementatie:

* Het platform dient beschikbaar te komen op de meest gangbare devices.
* De implementatie moet in Java.

## Product Functionaliteiten

Diagram

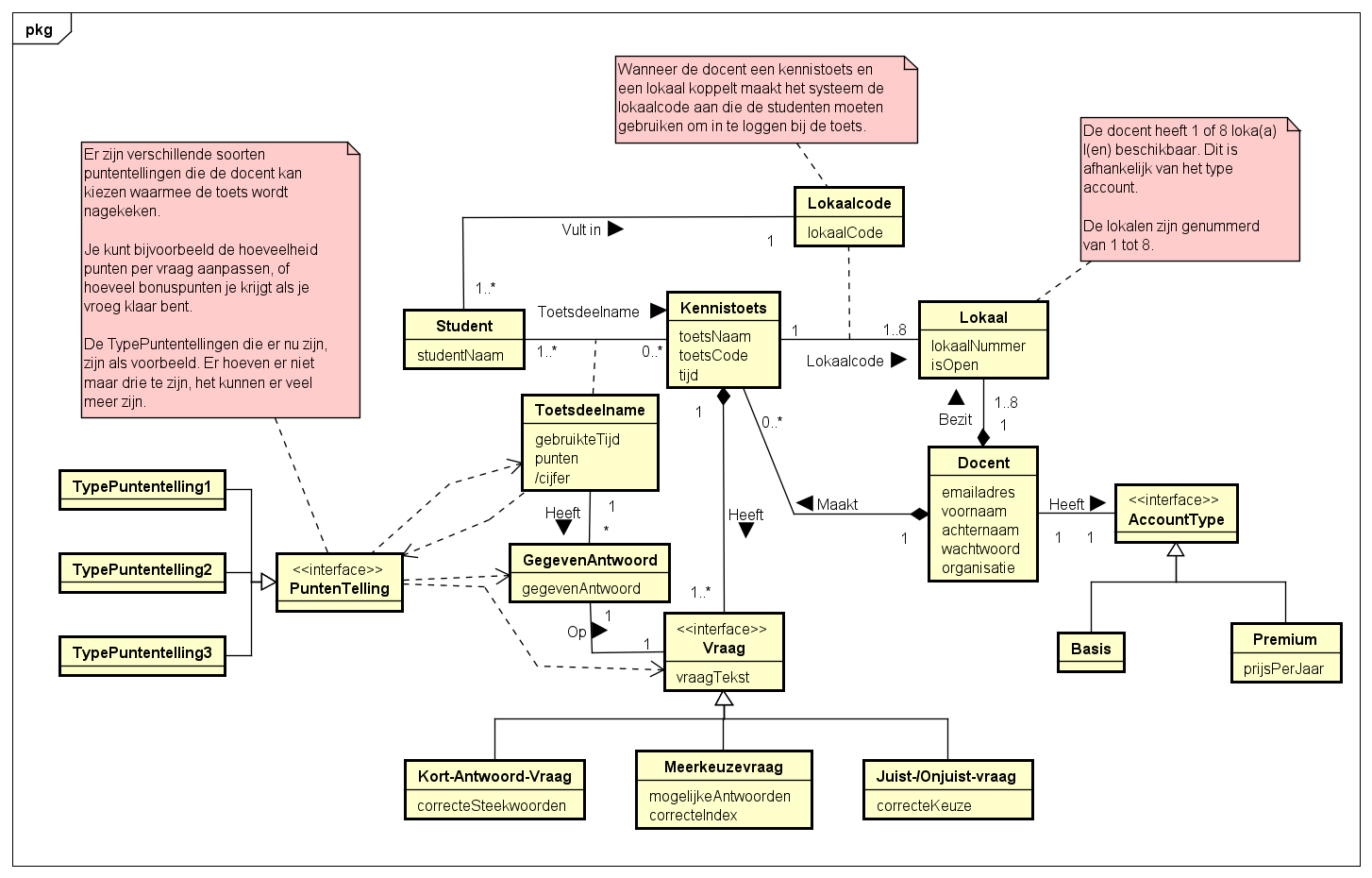
Description automatically generated

1

|  |  |
| --- | --- |
| Use Case | Brief Description |
| Registreren | Wanneer een docent gebruik wil maken van Provo moet hij/zij zich eerst registreren. Hierbij moet hij/zij zijn/haar emailadres, naam, wachtwoord en school/organisatie invoeren. Daarnaast kan hij/zij kiezen tussen een basis en een premium account.  Een premium account kost €49,99/jaar en geeft de docent toegang tot acht lokalen. Dit in tegenstelling tot het enkele lokaal van het basisaccount. |
| Account aanpassen | Een docent moet zijn/haar account kunnen aanpassen. Hierbij kan hij zijn/haar gegevens veranderen, maar ook het accounttype kan aangepast worden.  Als een docent van een premium account teruggaat naar een basisaccount kan hij/zij de rest van het betaalde jaar gebruik blijven maken van de premium services. |
| Account verwijderen | Als een docent besluit geen account meer bij Provo te willen hebben moet hij/zij zijn/haar account kunnen verwijderen. De gegevens zullen dan allemaal uit de database gehaald moeten worden. |
| Kennistoets aanmaken | Een kennistoets wordt door de docent zelf aangemaakt. Een kennistoets kan drie soorten vragen bevatten. Namelijk: meerkeuzevragen, juist/onjuist-vragen en kort-antwoord-vragen.  De docent moet een naam geven aan de toets, en het systeem zal een unieke toetscode genereren. |
| Kennistoets aanpassen | Een docent kan zijn/haar eigen kennistoetsen aanpassen. Hierbij kan de toetstitel veranderd worden, en vragen kunnen toegevoegd, gewijzigd en verwijderd worden. |
| Kennistoets verwijderen | Een docent kan zijn/haar eigen kennistoetsen verwijderen. De toetsen worden dan volledig uit het Provo systeem gehaald, en de toetscode komt beschikbaar voor andere toetsen. |
| Kennistoets starten | Een docent kan zijn/haar kennistoetsen openzetten in een lokaal zodat zijn/haar studenten deze kunnen uitvoeren.  Hierbij moet de docent de toetstitel, de lokaalcode(s) en tijd maximumtijd meegeven. Provo zal dan het lokaal openzetten en de lokaalcode laten zien waarmee studenten kunnen inloggen. |
| Genereren totaaloverzicht | Wanneer alle studenten klaar zijn met de toets kan de docent het totaaloverzicht laten zien. Hierbij telt het systeem de punten en bonuspunten van een student op en koppelt deze aan de studentnaam. Daarna worden ze in het overzicht geplaatst. Als alle studenten zijn geweest laat het systeem het overzicht zien. |
| Uitvoeren kennistoets | De studenten kunnen een meedoen aan de kennistoets door naar Provo te gaan en de lokaalcode in te voeren. Daarna moeten ze een naam voor zichzelf invoeren.  De student krijgt de vragen te zien in de volgorde waarin zo door de docent zijn aangemaakt. De student kan echter wel door de vragen heen scrollen zodat hij/zij niet vast komt te zitten op een vraag waar ze het antwoord niet op weten. Ook kunnen ze terug zodat ze antwoorden kunnen aanpassen.  Wanneer een student snel klaar is met de toets krijgt hij/zij bonuspunten die later bij de score worden opgeteld. |

# Domeinmodel

## Domeinmodel



2

## Glossarium

|  |  |
| --- | --- |
| Klassenaam | Betekenis |
| Docent | Het concept docent houdt alle gegevens van de docent |
| AccountType | Een docent heeft de keuze uit twee typen accounts. Namelijk basis en premium. |
| Basis | Het accounttype basis is gratis en komt met één lokaal waar de docent kennistoetsen in kan uitvoeren. |
| Premium | Het accounttype premium kost €49,99 per jaar en komt met acht lokalen waar de docent kennistoetsen in kan uitvoeren. De prijs per jaar wordt hier opgeslagen |
| Student | De student is de persoon die de kennistoets uitvoert. Om deel te kunnen nemen moet hij zijn/haar eigen naam invullen. Die naam wordt in dit concept opgeslagen |
| Kennistoets | Een kennistoets wordt aangemaakt door de docent en heeft zowel een naam als een code aan zich gekoppeld. Ook moet de docent een maximumtijd toevoegen aan de toets. |
| Lokaal | In een lokaal worden de kennistoetsen uitgevoerd. Een lokaal heeft een nummer van één tot acht en kan toegankelijk of gesloten zijn voor leerlingen. Het lokaalnummer en de status met betrekking tot de toegankelijkheid (isOpen) worden hier opgeslagen. |
| LokaalCode | Wanneer een kennistoets aan een lokaal wordt gekoppeld wordt er een lokaalcode aangemaakt. Deze code is nodig voor de studenten om zich aan te kunnen melden bij de toets. Standaard is de lokaalcode leeg, pas wanneer de lokaalcode is aangemaakt staat die in dit concept. |
| Toetsdeelname | Wanneer een leerling zich aanmeldt bij een kennistoets via de lokaalcode wordt er een Toetsdeelname aangemaakt. Hierin wordt de gebruikte tijd, de punten en het uiteindelijke cijfer opgeslagen. |
| GegevenAntwoord | Wanneer een student antwoord geeft op een vraag wordt die hier opgeslagen. |
| Vraag | De vragen die de docent maakt voor de oefentoets worden hier opgeslagen. De vraagTekst is de tekst van de vraag zelf. |
| Kort-Antwoord-Vraag | Het eerste type vraag dat in een kennistoets kan zitten is de Kort-Antwoord-Vraag. Hierbij moet de docent aangeven welke steekwoorden goed zijn. De juiste steekwoorden worden opgeslagen. |
| Meerkeuzevraag | Het tweede type vraag is de meerkeuzevraag. Hierbij geeft de docent een aantal mogelijke antwoorden waarbij er één goed is. Deze vragen en de index van de array van het juiste antwoord worden opgeslagen in dit concept. |
| Juist-/Onjuist-Vraag | Het derde type vraag dat de docent kan maken is een juist-/onjuist-vraag. Hierbij geeft de docent aan of het juist of onjuist is. Die keuze wordt opgeslagen. |
| Puntentelling | De manier van het tellen van punten kan veranderd worden. Hierom is het concept puntentelling een interface. |
| TypePuntentelling1 -2 en-3 | Dit zijn voorbeelden van hoe puntentellingen in dit model zitten. Je kunt makkelijk een andere manier selecteren. |

# Use-case omschrijvingen

Hier worden een aantal use-cases uitgelegd door middel van een fully-dressed omschrijving en een system sequence diagram. De use-cases die omschreven worden zijn: Registreren, Genereren van totaalscores en het Starten- en uitvoeren van een kennistoets.

## Registreren

### Fully-dressed use-case omschrijving

|  |  |
| --- | --- |
| **ID:** UC-1 | |
| **Titel:** Registreren | |
| **Pimary Actor:** Docent | |
| **Stakeholders:** Docent | |
| **Precondities:**   * De docent heeft geen account. | |
| **Postcondities:**   * De docent heeft een account. * Het account is opgeslagen in de database. * Er is een lokaal aan het account gekoppeld. | |
| **Main Succes Scenario (Basis account)** | |
| **Actor Action:** | **System Action:** |
| 1. De docent wil registreren |  |
|  | 1. Het systeem toont het registratiescherm |
| 1. De docent voert in: emailadres, voornaam, achternaam, wachtwoord en welke school/organisatie hij werkzaam is. |  |
| 1. De docent kiest voor account type. In dit geval voor basis. |  |
|  | 1. Het systeem wijst een lokaal toe aan het account. |
|  | 1. Het systeem slaat het account op in de database samen met het lokaal en de ingevulde gegevens. |
| **Alternative Flow(s):**  4A. De docent kiest voor account type. In dit geval voor Premium.  5A. Het systeem toont het scherm met betalingsproviders.  6A. De docent kiest een betalingsprovider uit.  7A. Het systeem gaat naar de juiste betalingsprovider.  8A. De docent betaald de prijs van het abonnement.  9A. Het systeem wijst de acht lokalen toe aan het account.  10A Ga verder bij 6. | |

### System Sequence Diagram

Diagram

Description automatically generated

3

## Starten kennistoets

### Fully-dressed use-case omschrijving

|  |  |
| --- | --- |
| **ID:** UC-2 | |
| **Titel:** Starten van een kennistoets | |
| **Pimary Actor:** Docent | |
| **Stakeholders:** Docent, Student | |
| **Precondities:**   * De docent heeft een account. * De docent is ingelogd. | |
| **Postcondities:**   * Kennistoets is gestart in gekozen lokaal. * Lokaal is geopend. * Lokaalcode is aangemaakt. * De studenten kunnen zich aanmelden. | |
| **Main Succes Scenario (Basis account)** | |
| **Actor Action:** | **System Action:** |
| 1. De docent besluit een kennistoets te starten. |  |
|  | 1. Het systeem toont de kennistoetsen die de docent gemaakt heeft. |
| 1. De docent kiest d.m.v. de toetscode of toetsnaam een kennistoets uit. |  |
|  | 1. Het systeem toont een overzicht met mogelijke toetslokalen. |
| 1. De docent kiest d.m.v. het cijfer gekoppeld aan het lokaal een lokaal uit. |  |
|  | 1. Het systeem maakt een lokaalcode aan. |
| 1. De docent voert een maximumtijd in. |  |
|  | 1. Het systeem stelt de tijd in bij de kennistoets. |
| 1. De docent kiest ervoor om het lokaal open te zetten voor studenten. |  |
|  | 1. Het systeem opent het lokaal. |
|  | 1. Het systeem laat de lokaalcode zien. |
| **Alternative Flow(s):**  **(Bij een premium account)**  4.1A. Het systeem laat een lijst zien van mogelijke lokalen.  5.1A. De docent kiest voor één lokaal.  6.1A. Ga verder bij 6.  4.2A. Het systeem laat een lijst zien van mogelijke lokalen.  5.2A. De docent kiest voor meerdere lokalen.  6.2A. Het systeem maakt meerdere lokaalcodes aan.  7.2A. Ga verder bij 7.  (**Wanneer de docent geen kennistoetsen heeft aangemaakt)**  2A. Het systeem geeft de melding dat de docent geen kennistoetsen heeft.  **(Wanneer de docent geen vrije lokalen meer heeft)**  4A. Het systeem geeft een melding dat er geen lokaal beschikbaar is. | |

### System Sequence Diagram

Table

Description automatically generated

4

## Uitvoeren kennistoets

### Fully-dressed use-case omschrijving

|  |  |
| --- | --- |
| **ID:** UC-3 | |
| **Titel:** Uitvoeren kennistoets | |
| **Pimary Actor:** Student | |
| **Stakeholders:** Student, Docent | |
| **Precondities:**   * Er is een lokaalcode aan gemaakt, met daarbij horend een toets. * Er bestaat een student. * De toets heeft vragen. * Lokaal is geopend voor studenten. | |
| **Postcondities:**   * Er is een toetsdeelname aangemaakt bij de student en kennistoets. * Bij de toetsdeelname zijn er gegeven antwoorden gemaakt en hoort bij de bijhorende vraag. | |
| **Main Succes Scenario (Basis account)** | |
| **Actor Action:** | **System Action:** |
| 1. De student vult naam in. |  |
|  | 1. Het systeem logt de student in. |
| 1. De student vult de lokaalcode in |  |
|  | 1. Het systeem kijkt of de lokaalcode klopt. |
|  | 1. Het systeem maakt een toetsdeelname aan. |
| 1. De student start de toets. |  |
| 1. De student wil naar een vraag. |  |
|  | 1. Het systeem toont de gegeven vraag. |
| 1. De student beantwoordt de vraag. |  |
|  | 1. Het systeem slaat het antwoord op. |
|  | Ga terug naar stap 7 tot alle vragen zijn beantwoord of de tijd om is. |
| 1. De student verzendt de toets. |  |
|  | 1. Het systeem slaat de gebruikte tijd op. |
|  | 1. Het systeem sluit toetsdeelname af. |
| **Alternative Flow(s):**  7A. De student geeft aan dat klaar te zijn met de toets.  8A. Ga verder bij 11. | |

### System Sequence Diagram

Diagram

Description automatically generated

5

## Genereren totaalscore

### Fully-dressed use-case omschrijving

|  |  |
| --- | --- |
| **ID:** UC-4 | |
| **Titel: G**enereren totaalscore | |
| **Pimary Actor:** Docent | |
| **Stakeholders:** Docent, Student | |
| **Precondities:**   * Alle studenten zijn klaar met de toets of de tijd voor de toets is verstreken. | |
| **Postcondities:**   * De totaalscores zijn berekend. * Een overzicht is gegenereerd en wordt weergegeven. | |
| **Main Succes Scenario** | |
| **Actor Action:** | **System Action:** |
| 1. De docent vraagt om het overzicht met de totaalscores. |  |
|  | 1. Het systeem telt de punten en bonuspunten op. |
|  | 1. Het systeem plaatst Studentnaam en totaalscore in overzicht |
|  | 1. Ga terug naar stap 2 tot alle studenten hun totaalscore hebben en in het overzicht staan |
|  | 1. Het systeem geeft het overzicht weer |
| **Alternative Flow(s):** | |

### System Sequence Diagram

Diagram

Description automatically generated

6

# Functionele eisen

|  |  |
| --- | --- |
| Code | Omschrijving |
| FR1 | Er moeten verschillende soorten vragen gemaakt kunnen worden. |
| FR2 | Er moeten twee soorten accounts aangemaakt kunnen worden. |
| FR3 | Hoe sneller een student klaar is, hoe meer punten toegedeeld zullen worden. |
| FR4 | Er kunnen maximaal vijftig studenten deelnemen aan een kennistoets in een lokaal. |
| FR5 | Een docent moet aan het einde van een kennistoets een overzicht met de totaalscores kunnen genereren. |

# niet-functionele eisen

## Usability

|  |  |
| --- | --- |
| Code | Omschrijving |
| NFR1 | Een student moet eenvoudig door de vragen kunnen navigeren. |
| NFR2 | Er moet gemakkelijk overgestapt kunnen worden op een andere puntentelling systematiek. |

## Supportability

|  |  |
| --- | --- |
| Code | Omschrijving |
| NFR3 | Er moeten andere talen toegevoegd kunnen worden. |
| NFR4 | Het moet beschikbaar komen op de meest gangbare devices |

# Bronnenlijst

*Case Study OOAD: Provo.* (sd). Opgehaald van https://onderwijsonline.han.nl/elearning/lessonfile/BylEz7MD/https://onderwijsonline.han.nl/elearning/lessonfile/BylEz7MD/eyJpdiI6IktuajJDTkhFMStWcXA2bFAydUlSOGc9PSIsInZhbHVlIjoiNlRCMENCeXAySHQvTHdCTlBCYWhxNkd1UTcvSTV4eFA0Yk1PZDRNQVYzRWhXL0NyTFE4NE9TcVVYT

