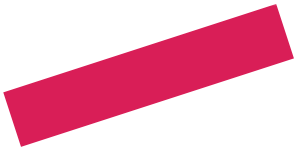
Software Requirements Specification

OOSE OOAD casus - Quebble

Sjoerd Scheffer
579392



HAN_UNIVERSITY
OF APPLIED SCIENCES

Docent: Marco Engelbart

2 juni 2021

Versie 2

Inhoudsopgave

Lijst van tabellen

Lijst van figuren

1 Inleiding

1.1 Algemene beschrijving

Het bedrijf Solid Games wil een quiz-applicatie Quebble ontwikkelen die beschikbaar wordt op de meest gangbare devices.

Een spel bestaat uit zowel open vragen als meerkeuzevragen, en de speler moet credits kopen om te kunnen spelen. De medewerkers van Solid Games zullen de quizzes beheren.

1.2 Gebruikersklassen en karakteristieken

Deze sectie beschrijft de aanwezige actoren en hun relatie tot het eindproduct.

- **Speler**: De eindgebruiker van het product. De speler kan het spel spelen door credits te gebruiken. De speler kan deze credits tegen vaste prijzen kopen in de applicatie.
- **Medewerker**: De medewerker die de quizzes en de bijbehorende vragen en antwoorden onderhoudt. De medewerker moet vragen aan het systeem kunnen toevoegen en verwijderen, en de sets aan vragen (quizzes) kunnen beheren.

1.3 Werkomgeving

Deze sectie beschrijft de omgeving waarin de applicatie functioneert.

De applicatie is aanvankelijk een console Java-applicatie voor desktop PC's.

Uiteindelijk draait de applicatie op PC's en mobiele telefoons. PC's en mobiele telefoons zijn namelijk de meest gangbare internetapparaten voor consumenten (StatCounter, 2021).

1.4 Ontwerp- en implementatiebeperkingen

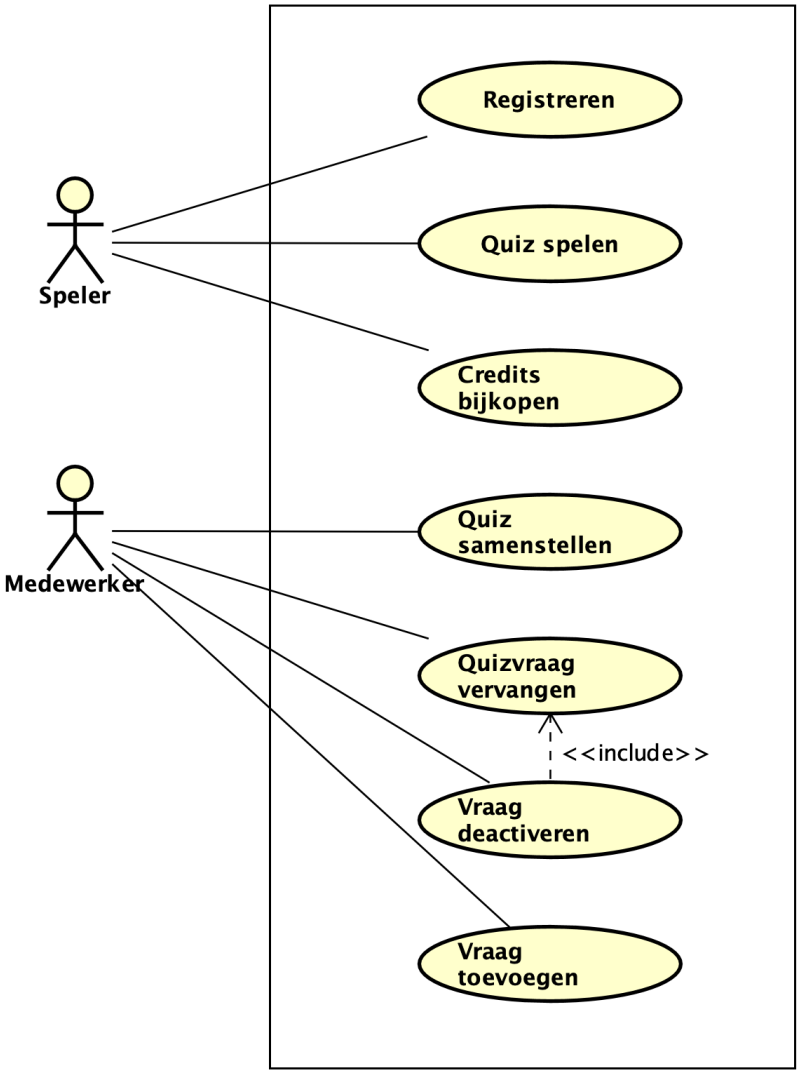
Deze sectie beschrijft de beperkingen van het systeem die de opties van ontwikkelaars kunnen begrenzen.

- De applicatie is in Java geschreven.
- De applicatie moet testbaar zijn doormiddel van unit tests.
- De applicatie moet aanvankelijk een console-applicatie zijn.
- De applicatie moet uiteindelijk als zowel desktop-applicatie als mobiele applicatie beschikbaar zijn.
- De applicatie gebruikt een in-memory object als mock voor een toekomstige database.

- De library waarmee de applicatie woorden controleert op correctheid moet vervangbaar zijn.
- De systematiek voor de puntentelling moet vervangbaar zijn.

1.5 Productfuncties

Deze sectie bevat het usecasediagram en de korte usecasebeschrijvingen.



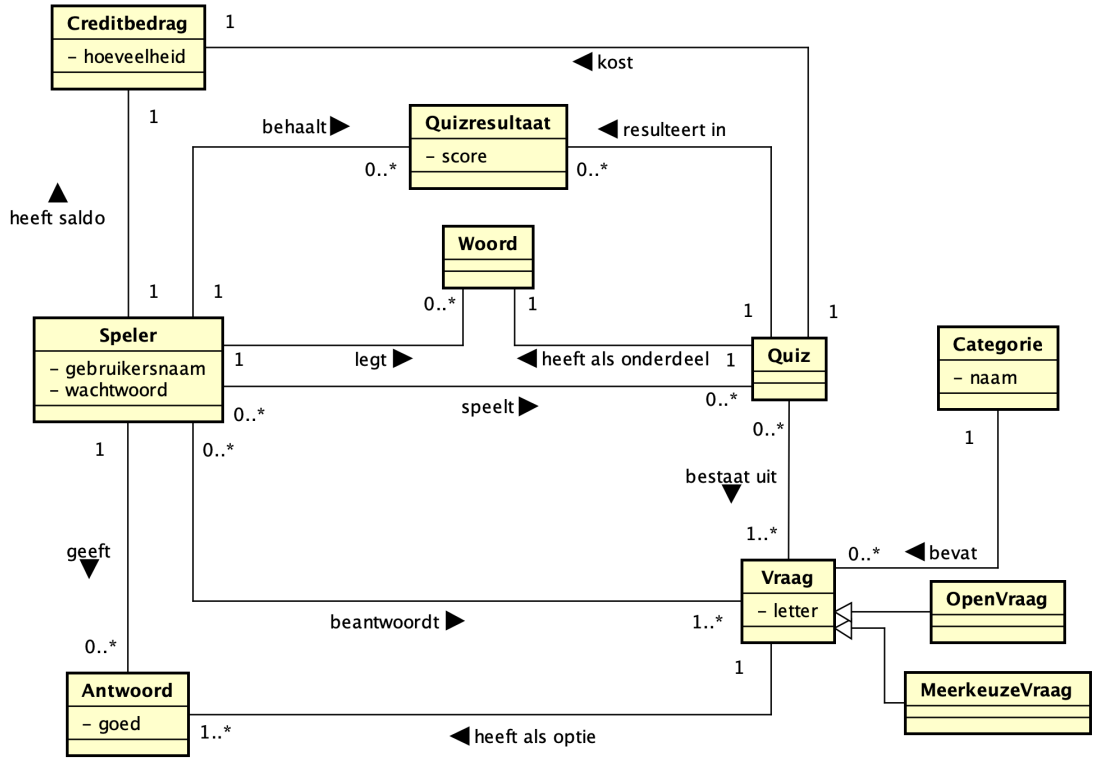
Figuur 1: Usecasediagram

De volgende korte beschrijven van de usecases uit figuur 1 zijn volgens de methodiek van Larman (2002) opgesteld (p. 46).

- **Registreren:** De speler voert een gebruikersnaam en wachtwoord in. Het systeem slaat de gegevens op, en wijst de speler 1000 credits toe.
- **Quiz spelen:** De speler start een quiz. Het systeem stelt vragen aan de speler. De speler beantwoordt een voor een de vragen. Het systeem geeft de speler een letter per goed antwoord. De speler maakt een woord van de letters. Het systeem controleert de antwoorden en het woord en kent de speler een score toe.
- **Credits bijkopen:** De speler kiest het aantal credits dat hij wil kopen. Het systeem geeft de bijbehorende prijs weer. De speler betaalt en het systeem kent de speler het gekozen aantal credits toe.
- **Quiz samenstellen:** De medewerker kiest 8 reeds opgeslagen actieve vragen. Het systeem slaat de vragen op in een speelbare quiz.
- **Quizvraag vervangen:** De medewerker kiest een reeds bestaande quiz. De medewerker kiest een vraag uit de quiz, en kiest vervolgens een vraag van buiten de quiz. Het systeem vervangt de geselecteerde quizvraag door de gekozen vraag van buiten de quiz.
- **Vraag toevoegen:** De medewerker vult een vraag en de bijbehorende categorie in. De medewerker geeft aan of het een meerkeuzevraag of een open vraag betreft. De medewerker vult de mogelijke antwoorden in. Het systeem slaat de vraag en de bijbehorende antwoorden op.
- **Vraag deactiveren:** De medewerker kiest een reeds opgeslagen active vraag. Het systeem deactiveert de vraag.

2 Domeinmodel

Dit hoofdstuk bevat het domeinmodel zoals beschreven door Larman (2002, p. 127). Dit model bevat alle relevante concepten om een quiz te kunnen spelen. Daarnaast bevat dit hoofdstuk zowel een woordenlijst als een onderbouwing voor de gemaakte keuzes.



Figuur 2: Domeinmodel

2.1 Woordenlijst

Deze sectie licht de gebruikte concepttermen uit het domeinmodel in figuur 2 toe:

- **Speler:** Speelt een quiz door vragen te beantwoorden. Beschikt over een creditbedrag als saldo om een quiz te betalen.
- **Creditbedrag:** Een hoeveelheid aan credits.
- **Quiz:** Een door de speler speelbare set aan vragen.
- **Quizresultaat:** Een door de speler behaald resultaat na het spelen van een quiz.
- **Categorie:** Een onderverdeling van vragen.

- **Woord:** Een woord dat de speler legt aan het einde van de quiz, gebruikmakend van de letters die hij door het correct beantwoorden van de voorafgaande vragen heeft verkregen.
- **Vraag:** Een vraag die de speler beantwoordt wanneer deze onderdeel is van een quiz.
  - **OpenVraag:** Een vraag die één of meerdere goede antwoorden heeft, maar waarvan deze antwoorden niet zichtbaar zijn voor de speler.
  - **MeerkeuzeVraag:** Een vraag die meerdere antwoorden bevat, waarvan er maar één juist is, en waarvan alle antwoorden zichtbaar zijn voor de speler.
- **Antwoord:** Een mogelijk antwoord dat de speler kan geven op een gestelde vraag.

2.2 Onderbouwing

Deze sectie onderbouwt de gemaakte keuzes van het domeinmodel in figuur 2.

Quizzes binnen Quebble worden gespeeld door de **speler**. Om aan een **quiz** te beginnen zijn moet de **speler** geregistreerd en ingelogd zijn, en over een bepaald saldo beschikken (zie precondities uit tabel 1). Daarom heeft de **speler** attributen voor de inloggegevens, en beschikt hij over een bepaald **creditbedrag** als saldo. Dit saldo en de inloggegevens zijn ook vereist bij de usecases *registreren* (zie sectie 3.1.2) en *credits bijkopen* (zie sectie 3.2.2). De **speler** beschikt nu over de benodigdheden om een **quiz** te kunnen spelen.

Wanneer een **speler** een **quiz** start, betaalt hij eerst het **creditbedrag** wat de **quiz** kost. Een **quiz** bestaat allereerst uit een set **vragen**. Deze vragen zijn ingedeeld in **categorieën**. De **speler** geeft een voor een **antwoord** op deze **vragen**.

Wanneer het een **open vraag** betreft, zijn er één of meerdere **antwoorden** goed. Deze laten zich echter niet zien aan de **speler**.

Wanneer het een **meerkeuzevraag** betreft, is er maar één goed **antwoord**. Deze laat zich, samen met een aantal foute **antwoorden**, wél zien aan de speler.

Voor elke **vraag** die de **speler** goed beantwoordt, krijgt de **speler** een letter. Deze letters gebruikt de **speler** aan het einde van de **quiz** om een **woord** te leggen. Wanneer de **speler** een correct woord heeft gemaakt van de letters, krijgt de speler van de **quiz** een **quizresultaat**. Hierin is de berekende score te zien.

3 Usecasebeschrijvingen

Dit hoofdstuk bevat uitgebreide uitwerkingen van de usecases.

De uitwerkingen bevatten fully-dressed usecasebeschrijvingen volgens de usecases.org methode (Larman, 2002, p. 50). Het succesvolle hoofdscenario is ingedeeld in twee kolommen als voorgesteld door Wirfs-Brock (1993), ten behoeve van de overzichtelijkheid.

De uitwerkingen bevatten ook systemsequencediagrammen volgens Larman (2002) die de succesvolle hoofdscenario's beschrijven (p. 110).

Eventuele activitydiagrammen zijn terug te vinden in de Software Design Description.

Alleen de drie belangrijkste usecases die nodig zijn om een quiz te spelen zijn hieronder uitgewerkt. Deze zijn geprioriteerd en verantwoord:

1. **Quiz spelen:** Deze usecase bevat het gehele spelverloop voor de speler.
2. **Registreren:** Deze usecase faciliteert precondities *"Speler is ingelogd"* en *"Speler is geregistreerd"* voor *Quiz spelen* (zie tabel 1).
3. **Credits bijkopen:** Deze usecase faciliteert de preconditie *"Speler beschikt over 40 of meer credits"* voor *Quiz spelen*.

De overige usecases zijn niet essentiëel voor het spelen van het spel. Binnen de context van de opdracht, faciliteert de gemockte dataset de preconditie *"Er is een speelbare quiz in het systeem aanwezig"* voor *Quiz spelen*.

3.1 Quiz spelen

3.1.1 Fully-dressed usecasebeschrijving

Tabel 1: Fully-dressed usecasebeschrijving van *Quiz spelen*

Primaire actor: Speler

Stakeholders en belangen:

- **Speler:** Wil een quiz spelen met zo min mogelijk eerder door hem gespeelde vragen. Wil aan het einde van de quiz zijn behaalde resultaten zien.
- **Bedrijf:** Wil een dermate leuke ervaring bieden aan de speler dat de speler naderhand meer credits blijft kopen.

Precondities:

- Speler is geregistreerd.
- Speler is ingelogd.
- Speler beschikt over 40 of meer credits.
- Er is een speelbare quiz in het systeem aanwezig.

Postcondities:

- Speler heeft alle quizvragen beantwoord.
- Speler heeft een woord gelegd.
- Speler heeft zijn behaalde score gezien.
- Speler beschikt over 40 minder credits dan voor het spelen.

Succesvol hoofdscenario:

Actoractie	**Systeemverantwoordelijkheid**
1. Speler start een nieuwe quiz.	
	2. Systeem schrijft 40 credits van de speler af.
	3. Systeem laat een korte uitleg van het spelverloop en de vraagstellingen zien.
	4. Systeem laat een nieuwe quizvraag zien.
5. Speler beantwoordt de quizvraag.	
Herhaal stap 4 en 5 totdat de speler alle quizvragen heeft beantwoord.	
	6. Systeem laat de behaalde letters zien.
7. Speler legt een woord.	
	8. Systeem laat de behaalde score zien.

Wordt vervolgd op de volgende pagina.

Alternatieve scenario's:

1a. Er is geen nog niet door de speler gespeelde quiz in het systeem aanwezig.
1. Systeem meldt dat er geen nog niet gespeelde quizzes zijn.
2. Speler kiest ervoor om een willekeurige, al gespeelde quiz te spelen.
    2a. Speler kiest ervoor om te stoppen.
        1. Stop.
3. Naar hoofdscenario stap 2.

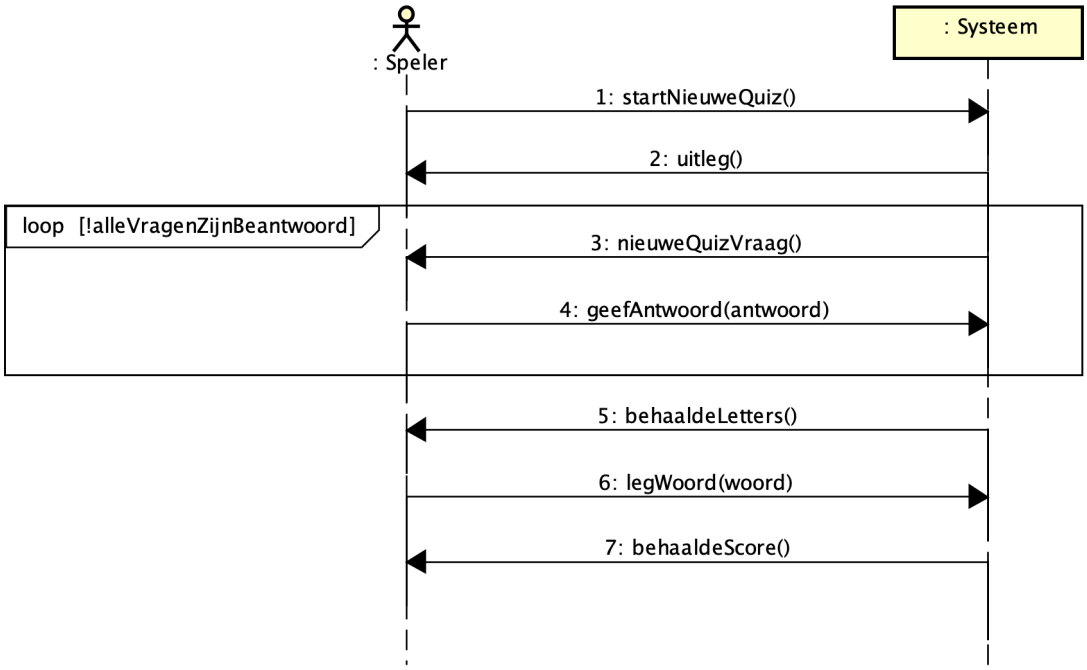
1b. Het systeem was gecrasht terwijl nog niet alle quizvragen waren beantwoord.
1. Speler hervat de nog openstaande quiz.
2. Naar hoofdscenario stap 4.

1c. Het systeem was gecrasht nadat alle quizvragen waren beantwoord, voordat er een woord was gelegd.
1. Speler hervat de nog openstaande quiz.
2. Naar hoofdscenario stap 6.

5a. Het gegeven antwoord is leeg, of het gegeven antwoord is geen keuzemogelijkheid.
1. Systeem laat een foutmelding zien.
2. Naar hoofdscenario stap 5.

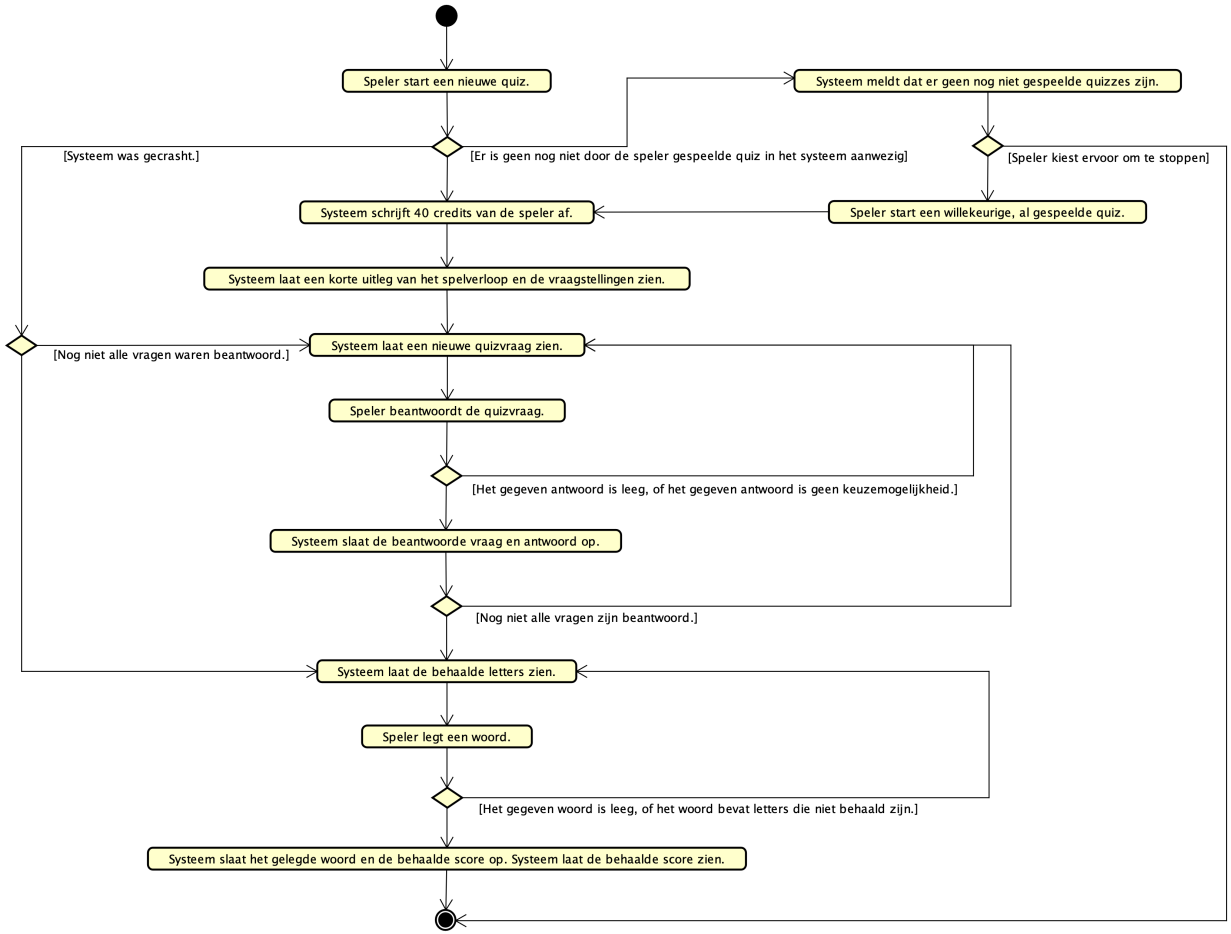
7a. Het gegeven woord is leeg, of het woord bevat letters die niet behaald zijn.
1. Systeem laat een foutmelding zien.
2. Naar hoofdscenario stap 7.

3.1.2 Systemsequencediagram



Figuur 3: Systemsequencediagram van *Quiz spelen*

3.1.3 Activitydiagram



Figuur 4: Activitydiagram van *Quiz spelen* (wordt verplaatst naar het SDD)

3.2 Registreren

3.2.1 Fully-dressed usecasebeschrijving

Tabel 2: Fully-dressed usecasebeschrijving van *Registreren*

Primaire actor: Speler

Stakeholders en belangen:

- **Speler:** Wil zo min mogelijk eerder gespeelde quizzes kunnen spelen. Wil na het registreren meteen kunnen beginnen met spelen.
- **Bedrijf:** Wil de speler doormiddel van 1000 gratis credits aanmoedigen om quizzes te spelen.

Precondities:

Postcondities:

- Speler heeft een account waarmee hij quizzes kan spelen.
- Speler beschikt over 1000 credits.

Succesvol hoofdscenario:

Actoractie	Systeemverantwoordelijkheid
1. Speler kiest ervoor om zich te registreren.	
2. Speler vult een gebruikersnaam en wachtwoord in.	
	3. Systeem registreert de speler en schrijft 1000 credits bij de speler bij.

Alternatieve scenario's:

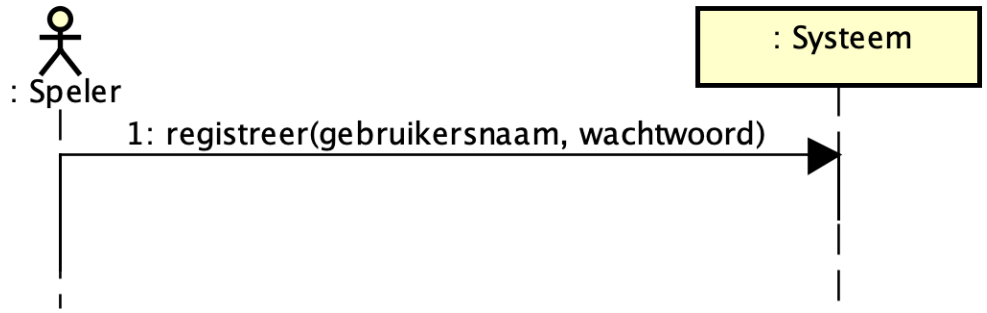
2a. De gebruikersnaam is leeg, voldoet niet aan de eisen voor een gebruikersnaam, of is al geregistreerd.

1. Systeem laat een foutmelding zien.
2. Naar hoofdscenario stap 2.

2b. Het wachtwoord is leeg, of voldoet niet aan de eisen voor een wachtwoord.

1. Systeem laat een foutmelding zien.
2. Naar hoofdscenario stap 2.

3.2.2 Systemsequencediagram



Figuur 5: Systemsequencediagram van *Registreren*

3.3 Credits bijkopen

3.3.1 Fully-dressed usecasebeschrijving

Tabel 3: Fully-dressed usecasebeschrijving van *Credits bijkopen*

Primaire actor: Speler

Stakeholders en belangen:

- **Speler:** Wil meer quizzes kunnen spelen dan voorheen mogelijk was.
- **Bedrijf:** Wil winst maken door de geldtransactie van de speler.

Precondities:

- Speler is geregistreerd.
- Speler is ingelogd.

Postcondities:

- Speler beschikt over het door de speler gekozen aantal credits meer dan voorheen.
- Speler kan de gekochte credits gebruiken om een quiz te spelen.
- Het bedrijf heeft de geldtransactie voor het gekozen aantal credits van de speler ontvangen.

Succesvol hoofdscenario:

Actoractie	**Systeemverantwoordelijkheid**
1. Speler geeft aan credits te willen kopen.	
	2. Systeem laat de mogelijke aantallen en de bijbehorende prijzen zien.
3. Speler kiest een aantal credits.	
4. Speler betaalt het voorgestelde bedrag.	
	5. Systeem schrijft het door de speler gekozen aantal credits bij de speler bij.

Alternatieve scenario's:

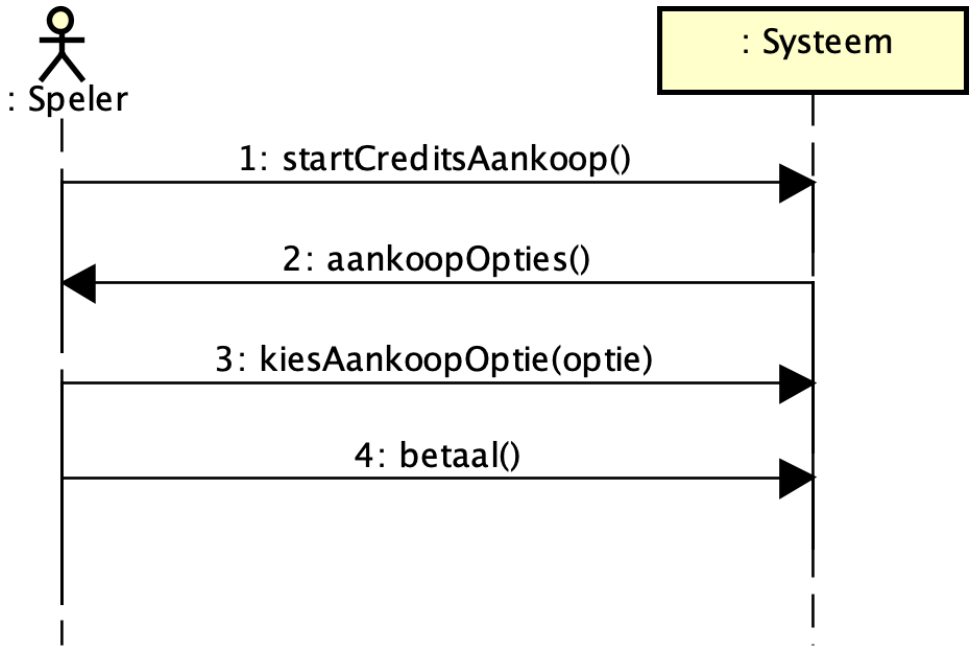
3a. Speler annuleert het bijkopen.

1. Stop.

4a. Betaling mislukt.

1. Systeem laat een foutmelding zien.
2. Naar hoofdscenario stap 2.

3.3.2 Systemsequencediagram



Figuur 6: Systemsequencediagram van *Credits bijkopen*

4 Andere functionele eisen

Deze sectie beschrijft functionele eisen die niet in het usecasediagram te vinden zijn. De eisen zijn ieder voorzien van een korte verantwoording.

FE1. De applicatie moet de wachtwoorden van spelers opslaan als Bcrypt hashes.
Dit verhoogt de beveiliging van gebruikersgegevens. Bcrypt is een van de meest veilige opties (Kamal, 2019).

5 Niet-functionele eisen

Deze sectie beschrijft de niet-functionele eisen volgens de FURPS+ methodiek (Grady, 1992).

FURPS op zichzelf is een van de simpelste modellen, en houdt voornamelijk rekening met gebruikerseisen, en minder met de ontwikkelaarseisen (Grady en Caswell, 1987, p. 159; AL-Badareen e.a., 2011). FURPS+ compenseert dit met extra categorieën (Eeles, 2005). FURPS+ is daarom een simpele doch alomvattende methodiek om eisen op te stellen.

De eisen zijn ieder voorzien van een korte verantwoording.

5.1 Usability

NFE1. De applicatie moet het saldo van de speler laten zien gelijk nadat hij zich geregistreerd heeft.
De speler weet direct dat hij kan beginnen met spelen zonder eerst extra credits te hoeven kopen.

NFE2. De applicatie moet het saldo van de speler laten zien voordat de speler een quiz start.
De speler kan op basis van het saldo een bewuste keuze maken om te spelen of niet.

NFE3. De applicatie moet de kosten van een quiz aan de speler laten zien voordat de speler een quiz start.
De speler kan op basis van de kosten een bewuste keuze maken om te spelen of niet.

NFE4. Wanneer mogelijk moet de applicatie altijd een nog niet gespeelde quiz kiezen wanneer de speler een quiz start.
De speler beleeft minder uitdaging bij al eerder gespeelde quizzes.

NFE5. De applicatie moet de speler vragen om door te gaan wanneer er geen nog niet gespeelde quiz aanwezig is.
De speler krijgt zelf de keuze of hij het waardevol vindt om een al eerder gespeelde quiz te spelen.

NFE6. De applicatie moet aan het begin van elke quiz melden dat de speler niet terug kan naar een vorige vraag.
De speler moet dit weten voor een eerlijk spelverloop.

NFE7. De applicatie moet aan het begin van elke quiz een tekst laten zien waarin de vraagstelling en de antwoordmogelijkheden van de meerkeuzevragen en de open vragen uitgelegd worden.
De speler moet dit weten voor een eerlijk spelverloop.

NFE8. De applicatie mag bij het controleren van antwoorden geen onderscheid maken tussen hoofdletters en kleine letters.
Dit beperkt het foutief afkeuren van goede antwoorden.

NFE9. De applicatie moet aan het einde van een quiz de behaalde score laten zien aan de speler.
De speler krijgt direct inzicht in zijn prestaties, en dit kan motiveren om door te spelen.

NFE10. Wanneer een speler kiest om credits te kopen, moet de applicatie de bijbehorende prijzen laten zien.
Dit voorkomt onverwachte kosten voor de speler.

NFE11. De applicatie moet de speler om bevestiging vragen bij elke door de speler geïnitieerde transactie van credits of geld.
Dit voorkomt onverwachte kosten voor de speler.

5.2 Reliability

NFE12. Het wijzigen van quizzes of vragen mag geen invloed hebben op quizzes die op dat moment gespeeld worden.
Dit voorkomt consistentiefouten bij quizzes die gespeeld worden (zoals dubbele vragen).

NFE13. Wanneer de applicatie is gecrasht tijdens een active quiz, moet de speler deze quiz na het herstarten van de applicatie kunnen hervatten.
Dit voorkomt dat spelers hun quiz niet kunnen afmaken, en dat ze onterecht credits kwijt zijn.

5.3 Performance

NFE14. De vertraging in de user interface nadat een speler een vraag beantwoordt mag niet langer duren dan 500 milliseconden.
Dit is een balans tussen een veilige marge voor ontwikkelaars, en een reactietijd (< 1 seconde) die gebruikers niet als onprettig zullen ervaren (Nielsen, 1993, p. 134).

5.4 Supportability

NFE15. De user interface, vragen en antwoorden moeten volledig vertaalbaar zijn naar andere talen.
Dit maakt het mogelijk om de applicatie uit te brengen in andere landen dan Nederland.

NFE16. Het moet mogelijk zijn om extra methodes toe te voegen voor het toekennen van scores.
De uiteindelijke puntentelling ligt volgens de opdrachtgever nog niet vast.

NFE17. Een externe library moet de door de speler ingevulde woorden kunnen controleren.
Hierdoor kan de applicatie gebruik maken van reeds bestaande systemen die door onder andere WordFeud en Scrabble worden gebruikt.

5.5 +

NFE18. De applicatie moet een Java-applicatie zijn.
Java-applicaties werken op Windows, macOS, Linux, Android en, met een omweg, op iOS (Oracle, g.d.; Google Developers, 2021; Multi-OS Engine, 2016).

Referenties

AL-Badareen, A., Selamat, M., A. Jabar, M., Din, J. & Turaev, S. (2011). Software Quality Models: A Comparative Study. *Communications in Computer and Information Science*, *179*, 46–55. https://doi.org/10.1007/978-3-642-22170-5_4

Eeles, P. (2005, november 15). *Capturing Architectural Requirements*. International Business Machines Corporation. Verkregen 12 november 2020, van https://web.archive.org/web/20201112020231/http://www.ibm.com/developerworks/rational/library/4706.html#N100A7

Google Developers. (2021, februari 23). *Application Fundamentals*. Verkregen 30 mei 2021, van https://developer.android.com/guide/components/fundamentals

Grady, R. B. (1992, februari 1). *Practical Software Metrics for Project Management and Process Improvement*. Prentice Hall.

Grady, R. B. & Caswell, D. L. (1987, mei 1). *Software Metrics: Establishing a Company-Wide Program* (1ste ed.). Pearson Education.

Kamal, P. (2019). Security of Password Hashing in Cloud. *Journal of Information Security*, *10*, 45–68. https://doi.org/10.4236/jis.2019.102003

Larman, C. (2002). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (2de ed.). Prentice Hall.

Multi-OS Engine. (2016). *Overview*. Verkregen 30 mei 2021, van https://doc.multi-os-engine.org/multi-os-engine/1_Overview/Overview.html

Nielsen, J. (1993). *Usability Engineering (Interactive Technologies)* (1ste ed.). Morgan Kaufmann.

Oracle. (g.d.). *Wat zijn de systeemvereisten voor Java?* Verkregen 30 mei 2021, van https://java.com/nl/download/help/sysreq.html

StatCounter. (2021, april 1). *Desktop vs Mobile vs Tablet vs Console Market Share Worldwide*. Verkregen 29 mei 2021, van https://gs.statcounter.com/platform-market-share#monthly-202004-202104

Wirfs-Brock, R. (1993). Designing Scenarios: Making the Case for a Use Case Framework. *Smalltalk Report*.