PROJECT portemonnee



Inhoudsopgave

[1. Inleiding 3](#_Toc509993590)

[2. Functioneel ontwerp 3](#_Toc509993591)

[2.1 Use-Case diagram 3](#_Toc509993592)

[2.2 Use-Case tabellen 3](#_Toc509993593)

[2.3 Menustructuur 3](#_Toc509993594)

[3. Technisch Ontwerp 3](#_Toc509993595)

[3.1 Klassendiagram 3](#_Toc509993596)

[3.2 Sequentiediagram 4](#_Toc509993597)

[4. Testrapport 4](#_Toc509993598)

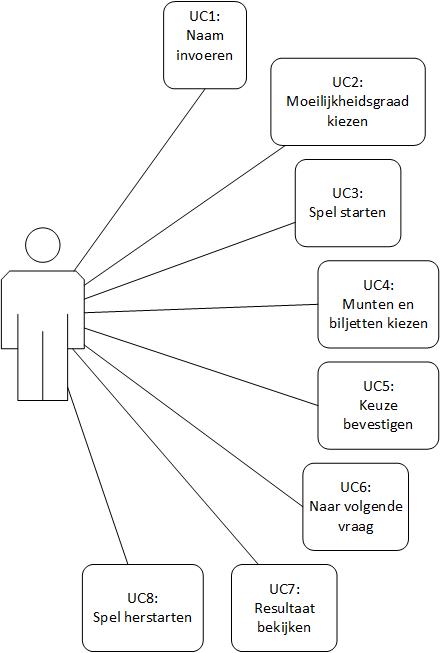
1. Inleiding

Om onze JavaScript vaardigheden te vergroten hebben wij het project Portemonnee gekregen. In dit project breiden we onze JavaScript ervaring uit. Ook leren we rekening houden met een specifieke (jonge) doelgroep.

2. Functioneel ontwerp

In het Functioneel Ontwerp (FO) staat een overzicht over de functionaliteiten waaraan de applicatie moet voldoen. Ook de gebruikersschermen, navigatiestructuur en uitvoerontwerp worden behandeld.

## 2.1 Use-Case diagram

In de use-case diagram laten wij alles zien wat een speler kunt doen m.b.v rondjes die aan de speler verbonden zijn.

## 2.2 Use-Case tabellen

In de use-case tabel beschrijven wij de use-cases en hoe ze geïmplementeerd worden.

|  |  |
| --- | --- |
|  |  |
| **Naam** | UC1: Naam invoeren |
| **Actor** | Speler |
| **Preconditie(s)** | - |
| **Actie** | De speler klikt op het vlakje “Vul hier je naam in” en vult hierzo zijn eigen naam in. |
| **Postconditie(s)** | De speler is nu gemachtigd om de moeilijkheidsgraad te kiezen. |
| **Uitzonderingen** | Als de speler geen naam invult, kan hij niet verder. |

|  |  |
| --- | --- |
|  |  |
| **Naam** | UC2: Moeilijkheidsgraad kiezen |
| **Actor** | Speler |
| **Preconditie(s)** | De speler moet zijn naam ingevoerd hebben. |
| **Actie** | De speler klikt op makkelijk of moeilijk en kiest zo doe moeilijkheidsgraad van het spel |
| **Postconditie(s)** | De speler kan nu het spel beginnen |
| **Uitzonderingen** | Als de speler niet kist, word automatisch makkelijk uitgevoerd. |

|  |  |
| --- | --- |
|  |  |
| **Naam** | UC3: Spel starten |
| **Actor** | Speler |
| **Preconditie(s)** | De speler moet zijn naam ingevuld hebben en moeilijkheidsgraad |
| **Actie** | De speler klikt op de start knop |
| **Postconditie(s)** | De speler is nu bij de eerste opdracht |
| **Uitzonderingen** | / |

|  |  |
| --- | --- |
|  |  |
| **Naam** | UC4: Munten en Biljetten kiezen |
| **Actor** | Speler |
| **Preconditie(s)** | De speler moet het spel gestart hebben |
| **Actie** | De speler klikt op de munten of biljetten en die komen op de counter te liggen |
| **Postconditie(s)** | Al heeft de speler genoeg geld neergelegd kan hij op bevestigen klikken |
| **Uitzonderingen** | / |

|  |  |
| --- | --- |
|  |  |
| **Naam** | UC5: Keuze bevestigen |
| **Actor** | Speler |
| **Preconditie(s)** | De speler moet geld op de counter hebben liggen. |
| **Actie** | De speler klikt op de bevestigen knop |
| **Postconditie(s)** | De speler is nu bij de eerste opdracht |
| **Uitzonderingen** | / |

## 

## 2.3 Menustructuur

In de menustructuur staat de menustructuur van de website en hoe er onderling gecommuniceerd word met de pagina’s.

3. Technisch Ontwerp

Het doel van het Technisch Ontwerp (TO) is het beschrijven hoe de functionele eisen zoals in het FO vastgelegd, geïmplementeerd zullen worden.

## 3.1 Klassendiagram

In het klassendiagram staan alle classes, methodes, functies en variablen die wij in de applicatie gaan gebruiken, en hoe die met elkaar communiceren.

## 3.2 Sequentiediagram

In de sequentiediagram maken wij gebruik van het MVC-Model (Model, View, Controller). Hierin staat hoe de 3 classes met elkaar gaan communiceren en met welke methodes.

4. Testrapport

In het testrapport testen wij alle use-cases en geven wij aan of er nog use-cases verkeerd werken of malfuncties hebben. (Hier gaan wij niet van uit!)