Functioneel Ontwerp

TypoTrainer v1.2.0

Project: TypoTrainer – Applicatie om snel en foutloos te leren typen

Sprint 2: UC3-UC4

Niveau: 2  
Versie: 1.2.0  
Datum: 05-12-2025

**Scrum Team**: Tim de Boer (1218918), Edward Popova (1208441), Kai Bloemers (1217499), Nicky Tse (117484), Rick Reimert (1214962)

Versiebeheer

|  |  |  |
| --- | --- | --- |
| Versie | Datum | Wijzigingen |
| 0.1.0 | 10-11-2025 | Initiële versie |
| 1.1.0 | 21-11-2025 | Sprint 1: UC1-UC2, dashboard, database koppelingen |
| 1.2.0 | 05-12-2025 | Sprint 2: UC3 – UC4 |
| 1.3.0 |  |  |
| 1.4.0 |  |  |
| 1.5.0 |  |  |

Distributie

|  |  |  |
| --- | --- | --- |
| Versie | Datum | Ontvangers |
| 1.0.0 | 10-11-2025 | Tim de Boer, Kai Bloemers, Nicky Tse, Rick Reimert |
| 1.1.0 | 21-11-2025 | Tim de Boer, Kai Bloemers, Nicky Tse, Rick Reimert |
| 1.2.0 | 05-12-2025 | Tim de Boer, Kai Bloemers, Nicky Tse, Rick Reimert |
| 1.3.0 |  |  |
| 1.4.0. |  |  |
| 1.5.0 |  |  |

Inhoud

[TypoTrainer v1.2.0 1](#_Toc584270300)

[1 Inleiding 3](#_Toc858623306)

[2 Domeinanalyse 4](#_Toc1070786328)

[2.1 Gewenste situatie 4](#_Toc1548616376)

[2.2 Domeinmodel 4](#_Toc27362736)

[2.3 Use case diagram 9](#_Toc49390985)

[5 Functionele en non-functionele requirements 13](#_Toc2073935573)

[6 Definition of Done 13](#_Toc138631633)

[7 Use cases (UC) 14](#_Toc372893562)

[7.1 UC1 Inloggen / (Sprint 1-extra): Dashboard laden 14](#_Toc630115080)

[7.2 UC2 Account aanmaken 14](#_Toc1128843528)

[7.3 UC3 Typo-oefening starten 15](#_Toc1001611617)

[7.4 UC4 Resultaten tonen 16](#_Toc1505976669)

[7.5 UC5 Adaptieve oefening genereren (Sprint 3) 17](#_Toc1444939119)

[Figuur 7 Wireframe Adaptieve oefening genereren 17](#_Toc2083114886)

[7.6 UC6 Voortgang monitoren (Sprint 3) 18](#_Toc326565336)

[8 Sprint 1 – Toegepaste Use Cases 18](#_Toc1415286559)

[9 Datastroom Sprint 2 (UC3 → UC4) 19](#_Toc572485123)

# Inleiding

Dit document beschrijft het Functioneel Ontwerp (FO) voor TypoTrainer, een leerapplicatie die gebruikers helpt om snel en foutloos te leren typen. Het Functioneel Ontwerp is gebaseerd op de gevalideerde requirements en beschrijft de functionele werking, de gebruikersscenario’s, het domeinmodel en de schermontwerpen.

Dit is versie 1.2.0 van het Functioneel Ontwerp voor TypoTrainer. Deze versie bevat alle uitbreidingen en wijzigingen voor Sprint 2 (UC3–UC5), waaronder:

- Uitwerking oefenlogica (UC3)

- Uitbreiding resultatenmodule (UC4)

- Detailuitwerking adaptieve generator (UC5)

Alle elementen zijn afgestemd op Requirementsanalyse v1.2.0.

1.2 Aanpassingen Sprint 2

- Nieuwe sectie toegevoegd: gedetailleerde logica voor real-time foutmarkering.

- Uitbreiding domeinmodel: precisering velden in Resultaat en Foutanalyse.

- Use case scenario’s voor UC3–UC4 uitgebreid met triggers, randvoorwaarden en foutafhandeling.

- UI-mockups geactualiseerd voor oefenmotor, resultaten en adaptieve aanbevelingen.

- Definitie van Done uitgebreid met dataconsistentie-eisen voor resultaten.

# Domeinanalyse

In dit hoofdstuk staat een beschrijving van het domein dat is voortgekomen uit een gesprek met de opdrachtgever. Het domein valt samen met de opgestelde requirements. Om het domein inzichtelijk te maken is gebruik gemaakt van een domeinmodel en een use case diagram.  
Het hoofdstuk begint met de huidige situatie, gevolgd door een beschrijving van de gewenste situatie. Daarna volgen de diagrammen met beschrijvingen van het domein.

In Sprint 2 ligt de focus op het implementeren van de oefenfunctionaliteiten, de resultatenverwerking en de adaptieve oefeninggenerator.

Daarom is in deze versie van het Functioneel Ontwerp aangegeven welke onderdelen van het domeinmodel binnen de scope van Sprint 2 vallen. Deze onderdelen zijn relevant voor UC3 (Typo-oefening starten) en UC4 (Resultaten tonen).

## Gewenste situatie

De gebruiker logt in op het systeem, kiest een oefening en typt de getoonde tekst. De app meet snelheid en nauwkeurigheid, markeert fouten en toont resultaten. Het systeem past zich automatisch aan op basis van de prestaties van de gebruiker (adaptief leren). Docenten kunnen voortgang van studenten bekijken.

## Domeinmodel

## Afbeelding met tekst, diagram, ontvangst, Lettertype Door AI gegenereerde inhoud is mogelijk onjuist.



**Toelichting op het Domeinmodel TypoTrainer**

Het domeinmodel beschrijft de belangrijkste gegevensstructuren (entiteiten) van het systeem TypoTrainer en hoe deze met elkaar samenhangen.

Het model vormt de logische basis voor de database- en softwarearchitectuur.

**Entiteiten binnen de scope van Sprint 2**

De volgende entiteiten worden in Sprint 2 (versie 1.3.0) functioneel uitgewerkt en (deels) geïmplementeerd:

* **Oefening** – wordt gebruikt om een oefening te kunnen selecteren en uitvoeren.
* **Oefentekst** – bevat de tekst die de gebruiker moet typen.
* **Resultaat** – slaat de prestaties van de gebruiker op na het afronden van een oefening.
* **Foutanalyse** – registreert fouten per oefening en vormt input voor adaptieve oefeningen.
* **AdaptieveOefeningGenerator** – analysemechanisme dat op basis van fouten nieuw oefenmateriaal aanbeveelt.

Deze entiteiten worden daarom in deze Sprint uitgebreid beschreven met extra velden zoals foutfrequenties, reactietijden en complexiteitsscores.

**Overige entiteiten**

Andere entiteiten in het domeinmodel, zoals Gebruiker, Student, Docent en Statistiek, vallen in Sprint 2 buiten de primaire implementatiescope. Ze worden wel weergegeven om het volledige gegevensdomein te tonen, maar worden in deze sprint niet functioneel uitgebreid of aangepast.

**Entiteiten Domein model**

**1.** **Gebruiker (leerling / student / docent)** *[valt buiten de scoop van Sprint 2]*

Hierin worden alle algemene gegevens over een gebruiker beheerd.

**Attributen:**

* GebruikerID (PK) – Unieke primaire sleutel (Primary Key) voor elke gebruiker.
* Naam – De volledige naam van de gebruiker.
* Email – Gebruikersadres voor login en communicatie.
* WachtwoordHash – Versleutelde versie van het wachtwoord (voor beveiliging).
* Rol – Geeft aan of iemand Student, Docent of Beheerder is.
* Aanmaakdatum – datum waarop het account is aangemaakt.

**Afkortingen:**

* PK = Primary Key
* FK = Foreign Key

**2. Student (subklasse van Gebruiker)** *[valt buiten de scoop van Sprint 2]*

Bevat specifieke gegevens over een student die oefeningen uitvoert.

**Attributen:**

* Leerdoelen – Persoonlijke doelen (zoals typesnelheid verbeteren).
* Typeniveau – Huidige vaardigheidsniveau van de student.
* TotaalOefeningen – Aantal uitgevoerde oefeningen.

**Relaties:**

* Wordt begeleid door één Docent (N:1).
* Voert meerdere Oefeningen uit via Resultaat (N:M).
* Heeft één Statistiek (1:1).
* Kan gekoppeld zijn aan één AdaptieveOefeningGenerator.

**3. Docent (subklasse van Gebruiker)** *[valt buiten de scoop van Sprint 2]*

Beheert studenten en monitort hun voortgang.

**Attributen:**

* School – De onderwijsinstelling waaraan de docent verbonden is.
* Vakgebied – Onderwijsvak waarin de docent actief is.

**Relaties:**

Kan meerdere Studenten begeleiden (1:N).

**4. Oefening**

Vertegenwoordigt een oefening die de student uitvoert.

**Attributen:**

* OefeningID (PK) – Uniek ID van de oefening.
* Titel – Naam van de oefening.
* Type – Aard van de oefening (bijv. tekst, spraak of beginner).
* Moeilijkheidsgraad – Niveau van de oefening (bijv. makkelijk, gemiddeld, moeilijk).
* Duur – Gemiddelde tijdsduur van de oefening.
* DatumToegevoegd – Wanneer de oefening aan de database is toegevoegd.
* Actief – Boolean (waar/niet waar) die aangeeft of de oefening beschikbaar is.

**Relaties:**

* Eén Oefening gebruikt één Oefentekst (1:N).
* Eén Student kan meerdere Oefeningen uitvoeren (via Resultaat).

**5. Oefentekst**

De tekst of spraakbron die tijdens de oefening wordt gebruikt.

**Attributen:**

* TekstID (PK) – Uniek ID van de oefentekst.
* Inhoud – De eigenlijke tekst of verwijzing naar een audiobestand.
* Categorie – Soort oefenmateriaal (letters, woorden, zinnen of thema’s).
* Taal – De taal van de oefentekst.
* Lengte – Lengte van de tekst (in woorden of karakters).

**Relatie:**

Eén Oefentekst kan worden gebruikt in meerdere Oefeningen.

**6. Resultaat**

Koppelt een Student aan een uitgevoerde Oefening en slaat de prestaties op.

Het Resultaat object slaat de volledige prestatiestatistieken van een afgeronde typ-oefening op.

In Sprint 2 wordt het uitgebreid met Reactietijden[] en TotaalTijd, die nodig zijn voor nauwkeurige foutanalyse én de adaptieve generator.

**Attributen:**

* ResultaatID (PK) – Uniek ID van het resultaat.
* GebruikerID (FK) – Verwijzing naar de Student die de oefening uitvoerde..
* OefeningID (FK) – Verwijzing naar de Oefening.
* Snelheid – Typ­snelheid in woorden per minuut (WPM).
* AantalFouten – Aantal gemaakte fouten tijdens de oefening.
* Nauwkeurigheid (%) – Percentage correcte toetsaanslagen.
* Reactietijden - Reactietijd tussen toetsaanslagen (ms).
* TotaalTijd - Totale tijd in seconden van begin tot eind.
* Datum – Wanneer de oefening is uitgevoerd.
* Opmerkingen – Extra observaties of notities.

**Relatie:**

* 1 Resultaat hoort bij exact 1 Student (N:1).
* 1 Resultaat hoort bij exact 1 Oefening (N:1).
* 1 Resultaat heeft maximaal 1 Foutanalyse (1:1).
* Resultaat levert input voor de AdaptieveOefeningGenerator.

**7. Foutanalyse**

Bevat de foutgegevens en analyse van een specifieke oefening. De entiteit Foutanalyse wordt gebruikt om fouten diepgaand te analyseren.

In Sprint 2 wordt deze uitgebreid met lijsten en een frequentiescore, die essentieel zijn voor adaptief leren.

**Attributen:**

* AnalyseID (PK) – Uniek ID voor de foutanalyse.
* ResultaatID (FK) – Verwijzing naar het bijbehorende resultaat.
* FouteLetters – Letters die fout zijn getypt, inclusief duplicaten.
* FouteWoorden – Volledige woorden die fout zijn getypt.
* Correctievoorstel – Suggestie voor verbetering.
* GemiddeldeReactietijd – Tijd tussen toetsaanslagen.

**Relaties:**

* 1:1 met Resultaat.
* Wordt gebruikt door AdaptieveOefeningGenerator om nieuwe oefeningen te genereren.

**8. AdaptieveOefeningGenerator**

Automatisch systeem dat nieuwe oefeningen aanpast op basis van fouten en prestaties.

**Attributen:**

* GeneratorID (PK) – Uniek ID van de generator.
* GebruikerID (FK) – Verwijzing naar de Student.
* DatumLaatsteAnalyse – Laatste datum waarop de analyse is uitgevoerd.
* GebaseerdOpAnalyse (boolean) – True/False of de oefening gepersonaliseerd is.
* AanbevolenOefeningID (FK) – Oefening die als volgende wordt aanbevolen.
* ComplexiteitScore (Integer [1-10]) - Score die aangeeft hoe moeilijk de nieuwe oefening is.

**Relaties:**

* Gebruikt meerdere Foutanalyses als input (N:1).
* Genereert exact 1 nieuwe Oefening (1:1).
* Is gekoppeld aan exact 1 Student.

**9. Statistiek** *[valt buiten de scoop van Sprint 2]*

Verzamelt overkoepelende prestatiegegevens van een Student.

**Attributen:**

* StatistiekID (PK) – Uniek ID van de statistiek.
* GebruikerID (FK) – Verwijzing naar de Student.
* GemiddeldeSnelheid – Gemiddelde woorden per minuut.
* GemiddeldeNauwkeurigheid – Gemiddelde juistheid in %.
* TotaalOefeningen – Totaal aantal uitgevoerde oefeningen.
* LaatsteUpdate – Datum van laatste berekening.

**Relatie:**

1:1 met Student.

**Samenvatting**

|  |  |  |
| --- | --- | --- |
| **Relatie** | **Type** | **Betekenis** |
| Docent – Student | 1:N | Eén docent begeleidt meerdere studenten |
| Student – Oefening | N:M (via Resultaat) | Studenten kunnen meerdere oefeningen uitvoeren |
| Oefening – Oefentekst | N:1 | Meerdere oefeningen kunnen dezelfde tekst gebruiken |
| Resultaat – Foutanalyse | 1:1 | Elk resultaat heeft maximaal één foutanalyse |
| Foutanalyse – AdaptieveOefeningGenerator | N:1 | Meerdere analyses kunnen leiden tot één nieuw oefenadvies |
| Student – Statistiek | 1:1 | Eén student heeft één overzicht met gemiddelde resultaten |

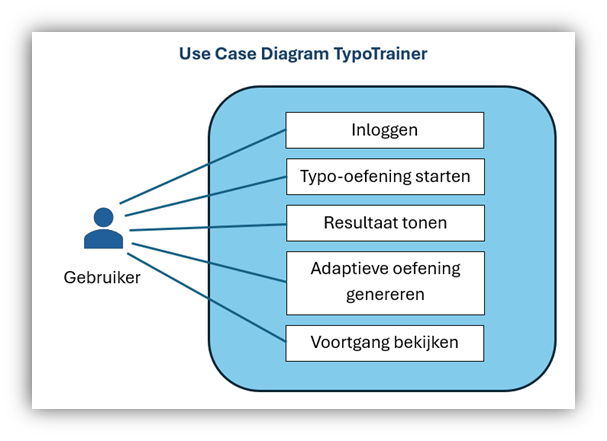
**Opmerking over notaties**

* PK (Primary Key) = unieke sleutel voor identificatie van een record.
* FK (Foreign Key) = verwijst naar de PK van een andere entiteit.
* 1:N / N:M / 1:1 = kardinaliteit (verhouding tussen objecten).
* Boolean = logische waarde (True/False).
* (abstract) = klasse die niet direct wordt geïnstantieerd (zoals “Gebruiker”).

**De opdrachtgever wil een systeem dat:**

* **Gebruiksvriendelijk is, met overzichtelijke schermen voor oefeningen, resultaten en voortgang;**
* **Data consistent opslaat in een beveiligde database, inclusief oefenresultaten en gebruikersprofielen;**
* **Persoonlijke leertrajecten ondersteunt, waarbij oefeningen zich automatisch aanpassen aan de fouten en snelheid van de gebruiker;**
* **Veilig omgaat met gebruikersgegevens, conform de AVG (**Algemene Verordening Gegevensbescherming - Europese wetgeving over de bescherming van persoonsgegevens)**, en betrouwbare sessies biedt bij in- en uitloggen;**
* **Duidelijke en motiverende feedback geeft tijdens en na het typen, zodat de gebruiker inzicht krijgt in snelheid en nauwkeurigheid:**
* **Voorbereid is op uitbreiding, zoals spraakgestuurde oefeningen, multiplayer-modi of integratie met educatieve platforms in toekomstige releases.**

## Use case diagram



*Figuur 2a Use Case Diagram (Student/Docent/Systeem)*

Het Use Case Diagram UI toont de interactie tussen de gebruiker en het systeem.

De belangrijkste use cases zijn:  
- Inloggen  
- Typo-oefening starten  
- Resultaten tonen  
- Adaptieve oefening genereren  
- Gebruikersvoortgang bekijken

Afbeelding met tekst, diagram, Lettertype, lijn

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 2b Uitgebreid Use Case Diagram

**Toelichting op het Use Case Diagram – TypoTrainer**

**Subsystem**

* TypoTrainer

**Actoren**

* Student
* Docent
* Systeembeheerder
* <<Service>> Identity Provider
* <<Service>> Interne database

**Use Cases binnen het subsysteem “TypoTrainer”**

1. Inloggen

* Actor: Student, Docent
* Relatie: <<include>> Authenticatie bij Identity Provider

1. Account aanmaken

* Actor: Student
* Relatie: <<extend>> Validatie via personal code

1. Typo-oefening starten

* Actor: Student
* Relaties:
* <<include>> Oefentekst ophalen uit Interne database

1. Fouten analyseren

* Actor: Student
* Relatie: <<include>> Resultaten opslaan in Interne database

1. Adaptieve oefening genereren

* Actor: Systeem (intern proces)
* Relaties:
* <<include>> Analyse resultaten uit vorige oefeningen
* <<extend>> Genereer moeilijkere oefening

1. Resultaten bekijken

* Actor: Student, Docent
* Relatie: <<include>> Data ophalen uit Interne database

1. Voortgang monitoren

* Actor: Docent
* Relatie: <<include>> Aggregatie van studentenresultaten

1. Systeem beheren

* Actor: Systeembeheerder
* Relatie: <<include>> Gebruikers- en dataonderhoud

**Relaties overzicht**

|  |  |
| --- | --- |
| Actor | Interacties met Use Cases |
| Student | Inloggen, Account aanmaken, Typo-oefening starten, Fouten analyseren, Adaptieve oefening genereren, Resultaten bekijken |
| Docent | Inloggen, Resultaten bekijken, Voortgang monitoren |
| Systeembeheerder | Systeem beheren |
| Speech Recognition API (mogelijke uitbreiding) | Ondersteunt spraakgestuurde oefening |
| Identity Provider | Authenticeert gebruikers tijdens het inloggen |
| Interne database | Slaat oefenteksten en resultaten op en levert ze terug aan de applicatie |

**UML-definitie**

|  |  |  |
| --- | --- | --- |
| Element | Betekenis | Voorbeeld in jouw diagram |
| Subsystem | De afbakening van het systeem dat je modelleert | <<Subsystem>> TypoTrainer |
| Actor | Een externe gebruiker of systeem dat met TypoTrainer communiceert | Student, Docent, Systeembeheerder |
| Use Case | Een handeling of functie die het systeem uitvoert | Inloggen, Oefening starten, Fouten analyseren, Resultaten bekijken |
| Include-relatie | Een verplicht onderdeel dat altijd plaatsvindt binnen een andere use case | *Inloggen* → <<include>> Authenticatie bij Identity Provider |
| Extend-relatie | Een optionele of voorwaardelijke uitbreiding van een andere use case | *Account aanmaken* → <<extend>> Validatie via personal code |
| Service-actor | Externe systemen die ondersteuning bieden | Identity Provider, Interne database |

1. Gevalideerde schermontwerpen en mockups

**Doel:**  
Dit hoofdstuk beschrijft de eerste schermontwerpen (User Interface (UI) -mockups) en de bijbehorende gebruikersscenario’s. De ontwerpen dienen als basis voor validatie met de product owner en toekomstige sprintplanning.

**Validatiecriteria:**

* De schermen sluiten aan op de functionele en non-functionele requirements.
* De navigatie en interacties zijn duidelijk en intuïtief.
* De visuele elementen ondersteunen het leerdoel (typen, feedback, motivatie).
* Feedback van de product owner is verwerkt in de definitieve versie.

**Overzicht schermen (voorbeeld):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr | Schermnaam | Doel | Belangrijkste elementen | Validatiestatus |
| S1 | Inlog/registratie scherm | Account aanmaken  Toegang tot persoonlijke omgeving | Gebruikersnaam, wachtwoordveld, login-knop | Goedgekeurd |
| S2 | Dashboard | Overzicht voortgang en oefeningen | Navigatiebalk, voortgangsbalk, oefening-knop | In review |
| S3 | Typo-oefening | Uitvoeren van typetest | Oefentekst, timer, foutmarkering, pauzeknop | Concept |
| S4 | Resultatenpagina | Weergave van prestaties | Woorden per minuut, foutenpercentage, herstartknop | Concept |

1. Scenario’s per scherm

Voor elk scherm wordt hieronder een gebruiksscenario beschreven volgens het “Given – When – Then”-formaat (zoals in Behavior-Driven Development testen).

**Scenario 1 – Inloggen, account aanmaken**

* Given: De gebruiker bevindt zich op het inlogscherm.
* When: De gebruiker voert een geldige gebruikersnaam en wachtwoord in en klikt op “Login”.
* Then: Het systeem controleert de gegevens, logt de gebruiker in en toont het dashboard.

**Scenario 2 – Typo-oefening uitvoeren**

* Given: De gebruiker bevindt zich op het dashboard.
* When: De gebruiker selecteert een oefening en begint te typen.
* Then: De timer start automatisch, fouten worden gemarkeerd en na afloop wordt een resultatenoverzicht getoond.

**Scenario 3 – Foutenanalyse**

* Given: De gebruiker heeft een oefening afgerond.
* When: De resultatenpagina wordt getoond.
* Then: De app markeert de gemaakte fouten in rood en toont statistieken over snelheid en accuraatheid.

**Scenario 4 – Adaptieve oefening starten**

* Given: De gebruiker heeft meerdere oefeningen voltooid.
* When: De app detecteert veelgemaakte fouten en genereert nieuw oefenmateriaal.
* Then: Het oefenmateriaal bevat meer woorden of letters die eerder verkeerd getypt werden.

# Functionele en non-functionele requirements

De volgende tabellen geven de gevalideerde functionele (FR) requirements en non-functionele (NFR) requirements weer, gebaseerd op de requirementsanalyse.

|  |  |  |
| --- | --- | --- |
| ID | Beschrijving | Prioriteit (MoSCoW) |
| FR1 | De gebruiker kan inloggen met gebruikersnaam en wachtwoord. | Must |
| FR2 | De gebruiker kan de oefening starten en pauzeren. | Must |
| FR3 | De fouten worden real-time gemarkeerd. | Must |
| FR4 | Resultaten worden direct opgeslagen. | Must |
| FR5 | Het systeem analyseert fouten en stelt nieuw oefenmateriaal voor op basis van de resultaten. | Must |
| FR6 | Adaptiviteit maakt aanbeveling zichtbaar. | Must |
| NFR1 | Reactietijd van het systeem < 2 seconden bij laden oefeningen. | Should |
| NFR2 | Dataopslag is AVG-conform en veilig. (AVG - Algemene Verordening Gegevensbescherming) | Must |

# Definition of Done

- Functionaliteit is getest en goedgekeurd door minimaal één reviewer.

- Code bevat geen kritieke bugs en is gekoppeld aan een issue in GitHub/Azure.

- Documentatie is bijgewerkt en gevalideerd door de product owner.

- Alle requirements uit Sprint 0&1 zijn geïmplementeerd of gepland voor Sprint 2.

**Sprint 2 uitbreiding:**

- Oefeningen laden binnen 2 seconden.

- Resultatenpagina toont foutanalyse correct.

# Use cases (UC)

Dit hoofdstuk beschrijft de use cases vanuit functioneel perspectief.

## UC1 Inloggen / (Sprint 1-extra): Dashboard laden

|  |  |
| --- | --- |
| Use case naam | Inloggen / Dashboard laden |
| Actoren | Gebruiker |
| Beschrijving | Na succesvolle login toont het systeem het dashboard met basisinformatie. Het dashboard dient als startpunt voor alle verdere acties. In Sprint 1 bevat het dashboard alleen navigatie naar profiel, toekomstige oefeningen en resultatenpagina’s (placeholder). |
| Precondities | * De gebruiker heeft een geregistreerd account. * Gebruiker kan succesvol inloggen. |
| Hoofdscenario | 1. Gebruiker opent de app.  2. Gebruiker voert gebruikersnaam en wachtwoord in.  3. Systeem valideert de gegevens.  4. Systeem toont het dashboard direct na login.  5. Gebruiker ziet navigatie-elementen: instellingen, oefening starten, adaptieve oefening & resultaten en voortgang (inactief). |
| Alternatieve scenario’s | 1a. Onjuiste login → foutmelding “Ongeldige gegevens”.  1b. Wachtwoord vergeten → gebruiker kiest “Wachtwoord herstellen”. |
| Postcondities | * De gebruiker is ingelogd en heeft toegang tot het persoonlijke profiel. * Dashboard is zichtbaar. Verdere functionaliteiten worden in latere sprints geactiveerd. |

Afbeelding met tekst, schermopname, software, Computerpictogram

Door AI gegenereerde inhoud is mogelijk onjuist.Afbeelding met tekst, schermopname, software, Computerpictogram

Door AI gegenereerde inhoud is mogelijk onjuist. *Figuur* 3 *Wireframe inloggen TypoTrainer* *Figuur 4 Wireframe Dashboard* *TypoTrainer*

### Evil user stories

Geen evil user stories kunnen bepalen. Geen nadere actie nodig.

## UC2 Account aanmaken

|  |  |
| --- | --- |
| Use case naam | Account aanmaken |
| Actoren | Gebruiker (nieuwe student/docent) |
| Beschrijving | Een nieuwe gebruiker maakt een persoonlijk account aan om toegang te krijgen tot TypoTrainer. Na registratie ontvangt de gebruiker (optioneel) een bevestigingsmail voordat hij of zij kan inloggen. |
| Precondities | * De gebruiker heeft nog geen geregistreerd account. * De gebruiker bevindt zich op het inlog- of registratiescherm. |
| Hoofdscenario | 1. De gebruiker opent het inlog-/registratiescherm. 2. Het systeem toont het registratievelden: e-mail, wachtwoord. 3. De gebruiker voert geldige gegevens in. 4. De gebruiker kiest voor “Account aanmaken”. 5. Het systeem valideert de ingevoerde gegevens. 6. Het systeem maakt een nieuw account aan en slaat de gegevens beveiligd op. 7. Het systeem toont de melding: “Account succesvol aangemaakt.” 8. De gebruiker kan nu inloggen met de nieuwe gegevens. |
| Alternatieve scenario’s | 2a. De gebruiker heeft al een account  → Het systeem toont een melding: “Er bestaat al een account met dit e-mailadres.”  4a. Onvolledige of ongeldige invoer  → Het systeem toont een foutmelding en vraagt de gebruiker om de gegevens te controleren. |
| Postcondities | * Het nieuwe account is succesvol aangemaakt en opgeslagen in de database. * De gebruiker kan inloggen met de geregistreerde gegevens. |

De grafische implementatie van het wireframe vrij is in te vullen door de developer.

Afbeelding met tekst, schermopname, software, Computerpictogram

Door AI gegenereerde inhoud is mogelijk onjuist.Afbeelding met tekst, schermopname, software, Computerpictogram

Door AI gegenereerde inhoud is mogelijk onjuist.

### Evil user stories

Geen evil user stories kunnen bepalen. Geen nadere actie nodig.

## UC3 Typo-oefening starten

|  |  |
| --- | --- |
| Use case naam | Typo-oefening starten |
| Actoren | Gebruiker |
| Beschrijving | De gebruiker selecteert een oefening, waarna de applicatie de oefentekst laadt, real-time fouten markeert, de timer start en de voortgang bewaakt. De oefenresultaten worden na afronding doorgestuurd naar UC4 (Resultaten tonen). |
| Precondities | Gebruiker is ingelogd en er is oefenmateriaal beschikbaar. De oefentekst kan probleemloos geladen worden. |
| Hoofdscenario | 1. Gebruiker kiest een oefening.  2. Systeem toont de voorbeeldtekst.  3. Timer start bij eerste toetsaanslag.  4. Fouten worden real-time gemarkeerd.  5. Gebruiker voltooit oefening (laatste karakter getypt).  6. Systeem stopt timer, verzamelt: totale tijd, reactietijden, fouten, nauwkeurigheid, snelheid, en stuurt resultaten naar UC4. |
| Alternatieve scenario’s | 2a. Oefening niet beschikbaar → foutmelding.  4a. Gebruiker pauzeert oefening → systeem slaat voortgang tijdelijk op. |
| Postcondities | Oefening is voltooid en resultaten worden opgeslagen. |

Afbeelding met tekst, schermopname, software, Computerpictogram

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 5 Wireframe Typo Oefening starten

### Evil user stories

Geen evil user stories kunnen bepalen. Geen nadere actie nodig.

## UC4 Resultaten tonen

|  |  |
| --- | --- |
| Use case naam | Resultaten tonen |
| Actoren | Gebruiker (Student), Docent (alleen lezer) |
| Beschrijving | Het systeem toont de resultaten van een afgeronde oefening: snelheid, fouten, nauwkeurigheid, reactietijden en foutanalyse. Tevens wordt informatie klaargemaakt voor de adaptieve generator (UC5). |
| Precondities | Er is een oefening afgerond. Ruwe meetdata (fouten, tijd, reactietijden) is beschikbaar. |
| Hoofdscenario | 1. Gebruiker voltooit oefening.  2. Systeem berekent snelheid (woorden per minuut) en foutpercentage.  3. Fouten worden in rood gemarkeerd in de tekst.  4. Systeem toont: snelheid (WPM); foutpercentage; totaal fouten; tijdsduur; grafiek met WPM &nauwkeurigheid.  5. Systeem slaat resultaten op en maakt samenvatting voor UC5. |
| Alternatieve scenario’s | 3a. Geen resultaten beschikbaar → melding “Nog geen data”.  4a. Geen grafiek-data beschikbaar → Alleen tekstuele statistieken worden getoond. |
| Postcondities | Resultaten zijn opgeslagen en zichtbaar voor gebruiker of docent. |

Afbeelding met tekst, schermopname, software, Computerpictogram

Door AI gegenereerde inhoud is mogelijk onjuist.

*Figuur 6 Wireframe Resultaten tonen*

### Evil user stories

Geen evil user stories kunnen bepalen. Geen nadere actie nodig.

## UC5 Adaptieve oefening genereren (Sprint 3)

|  |  |
| --- | --- |
| **Use case naam** | **Adaptieve oefening genereren** |
| Actoren | Systeem (intern proces) |
| Beschrijving | Op basis van de foutanalyse berekent de applicatie de zwakke punten van de gebruiker en stelt een nieuwe gepersonaliseerde oefening samen. |
| Precondities | Minimaal drie eerdere resultaten beschikbaar. Foutanalyse bevat bruikbare data (foute letters/woorden). |
| Hoofdscenario | 1. Systeem analyseert fouten en letters met hoge foutfrequentie.  2. Op basis van analyse wordt een nieuwe oefening samengesteld.  3. Gebruiker krijgt melding “Aanbevolen oefening klaar”. |
| Alternatieve scenario’s | 1a. Onvoldoende data → systeem biedt standaard oefening aan.  2a. Foutanalyse bevat geen relevante fouten → Systeem kiest oefening gericht op algemene verbetering.  4a. Geen passende oefentekst gevonden → Systeem genereert automatische oefentekst. |
| Postcondities | Nieuwe oefening is toegevoegd aan de beschikbare lijst. Dashboard toont aanbevolen oefening. |
| Aanvullende Info | * Frequente fouten krijgen hogere weging: f(x) = aantal fouten × 1.3. * ComplexiteitScore bepaalt woordlengte, snelheid en moeilijkheid. * UC3 ontvangt adaptieve oefeningen als eerste prioriteit in de lijst. |

## *Figuur 7 Wireframe Adaptieve oefening genereren*

### 7.5.1 Evil user stories

Geen evil user stories kunnen bepalen. Geen nadere actie nodig.

## UC6 Voortgang monitoren (Sprint 3)

|  |  |
| --- | --- |
| **Use case naam** | **Voortgang monitoren** |
| Actoren | Gebruiker, Docent |
| Beschrijving | Overzicht krijgen van leerontwikkeling over tijd. |
| Precondities | Gebruiker heeft minstens één oefening afgerond. |
| Hoofdscenario | 1. Gebruiker opent “Voortgang”.  2. Systeem haalt resultaten op uit database.  3. Resultaten worden in grafiekvorm getoond (WPM – woorden per minuut, foutenpercentage). |
| Alternatieve scenario’s | 2a. Geen resultaten → melding “Nog geen oefenresultaten beschikbaar”. |
| Postcondities | Voortgangsgegevens worden weergegeven en kunnen geëxporteerd worden. |

Afbeelding met tekst, schermopname, software, Computerpictogram

Door AI gegenereerde inhoud is mogelijk onjuist.*Figuur 8 Wireframe Gebruikersvoortgang bijhouden*

### 7.6.1 Evil user stories

Geen evil user stories kunnen bepalen. Geen nadere actie nodig.

# Sprint 1 – Toegepaste Use Cases

Sprint 1 omvat de implementatie van de volgende Use Cases:

* **UC1 – Inloggen**
* **UC2 – Account aanmaken**
* **Dashboard tonen** (onderdeel van UC1)
* Opzetten van database en databasekoppelingen

De overige Use Cases (UC3–UC6) worden geïmplementeerd in latere sprints en vallen buiten Sprint 1.

# Datastroom Sprint 2 (UC3 → UC4)

1. Student selecteert oefening
2. Student typt oefening
3. Systeem slaat real-time events op
4. Oefening eindigt
5. Resultaat wordt berekend
6. Resultaat + foutanalyse opgeslagen
7. Adaptieve analyse draait batch
8. Nieuwe oefening wordt aanbevolen op dashboard
9. Je PO gaat dit waarderen, want het laat zien dat je Sprint 2 begrijpt als een keten en niet losse functies.