## horizontale lijn



Snooty Snakes

22-10-2020

**─**

Analyse project Tutors

Dave Saenen, Wouter Pardon, Niels De Cat, Fabio Puissant

Uhasselt Martelarenlaan 42, 3500 Hasselt

# Probleemstelling

Een student die moeite heeft, met bepaalde onderwerpen uit zijn/haar opleiding, kan niet altijd de gepaste bijles vinden. Het is ook vaak omslachtig om de juiste personen te vinden voor bijles omdat er vaak een tussenpersoon voor nodig is die een tutor koppelt aan een student. Ook is er een sociale drempel die overwonnen moet worden om naar hulp te vragen. Verder is een tutor ook gebonden aan het onderwijsinstituut, terwijl mogelijk tutoren uit andere instituten ook dezelfde kennis hebben.

Dit projectvoorstel heeft als doelstelling studenten, uit hoger onderwijsinstellingen, het gemakkelijker te maken om bijles van een ander persoon (=*tutoring*) te vinden. De student in nood heeft zelf de touwtjes in handen om op zoek te gaan naar de geschikte persoon.

De oplossing op het hierboven vermelde probleem wordt in dit project geïmplementeerd in de vorm van een webapplicatie waarin gebruikers hulp kunnen aanbieden of hulp kunnen vinden voor verschillende vakken.

# Use Cases

## Inloggen

**Actoren:** administrator, tutor en student

**Beschrijving:**

Hoofdscenario: een gebruiker geeft zijn logingegevens aan het systeem en het systeem controleert of deze gegevens correct zijn [A] [B]. Het systeem geeft toegang aan de gebruiker om het systeem te gebruiken.

Alternatief scenario: [A]: de logingegevens zijn foutief. Het systeem geeft een toepasselijke foutboodschap.

[B]: de gebruiker logt zich in met een “third party” (zoals Github, Facebook of Google).

## Tutors beheren

**Actoren:** administrator

**Beschrijving:**

Hoofdscenario: een administrator navigeert op de administrator pagina naar tutors. Het systeem geeft een overzicht van alle tutors. De administrator kan volgende operaties uitvoeren:

1. Tutors bekijken
2. Tutors wijzigen
3. Tutors toevoegen
4. Tutors verwijderen

## Students beheren

**Actoren:** administrator

**Beschrijving:**

Hoofdscenario: een administrator navigeert op de administrator pagina naar students. Het systeem geeft een overzicht van alle students. De administrator kan volgende operaties uitvoeren:

1. students bekijken
2. students wijzigen
3. students toevoegen
4. students verwijderen

## Categorieën beheren

**Actoren:** administrator

**Beschrijving:**

Hoofdscenario: een administrator navigeert op de administrator pagina naar categorieën. Het systeem geeft een overzicht van alle categorieën. De administrator kan volgende operaties uitvoeren:

1. categorieën bekijken
2. categorieën wijzigen
3. categorieën toevoegen
4. categorieën verwijderen

## Meetings beheren

**Actoren:** tutor

**Beschrijving:**

Hoofdscenario: een tutor navigeert naar zijn/haar persoonlijke meetings. Het systeem geeft een overzicht van al deze meetings. De tutor kan volgende operaties uitvoeren [A]:

1. meetings bekijken
2. meetings wijzigen
3. meetings verwijderen
4. meetings kiezen

**let op: een tutor kan geen nieuwe meetings aanmaken!**

Alternatief scenario: [A] de tutor heeft nog geen meetings. Het systeem geeft een melding dat er nog geen meetings zijn.

## Meetings bekijken

**Actoren:** student, tutor

**Beschrijving:**

Hoofdscenario: een student of tutor navigeert naar zijn persoonlijke meetings. Het systeem geeft een overzicht van alle meetings waar hij zich in bevindt. De student of tutor kijkt zijn meetings na [A].

Alternatief scenario: [A] de student of tutor heeft nog geen meetings. Het systeem geeft een melding dat er nog geen meetings zijn.

## Lectures zoeken

**Actoren:** student

**Beschrijving:**

Hoofdscenario: een student navigeert naar de zoekpagina. Het systeem geeft een overzicht van alle lectures [A]. De student opent een lecture [B]. Het systeem geeft details over de geselecteerde lecture.

Alternatief scenario: [A] in het systeem zijn er nog geen lectures. Het systeem geeft een melding dat er nog geen lectures zijn.

[B] De student gebruikt een filter om op categorieën te zoeken. Het systeem geeft een vereenvoudigde lijst van meetings. Hij opent een lecture uit de gefilterde lijst.

## Lectures beheren

**Actoren:** tutor

**Beschrijving:**

Hoofdscenario: een tutor navigeert naar zijn persoonlijke lectures. Het systeem geeft een overzicht van alle lectures die zijn aangemaakt door de tutor [A]. De tutor kan volgende operaties uitvoeren:

1. lectures bekijken
2. lectures wijzigen
3. lectures toevoegen
4. lectures verwijderen

Alternatief scenario: [A] de tutor heeft nog geen lectures. Het systeem geeft een melding dat er nog geen meetings zijn.

## Inschrijven

**Actoren:** student

**Beschrijving:**

Hoofdscenario: een student opent een lecture op de zoekpagina en schrijft zich in op de lecture. Het systeem maakt een tijdelijke meeting aan en vraagt bevestiging aan de tutor. De tutor keurt de meeting goed, het systeem stuurt een notificatie naar de student en het systeem voegt de meeting ook toe aan de persoonlijke meetings van de tutor én de student [A].

Alternatief scenario: [A] de tutor keurt de meeting niet goed. Het systeem verwijdert de meeting en stuurt een notificatie naar de student.

**Let op: onderlinge communicatie tussen tutors en students zoals telefoon of email zitten niet in de scope van dit project.**

# URPS

## Usability:

Het systeem moet in staat zijn een visuele indicatie te geven aan de gebruiker wanneer er geen resultaten zijn voor de opgegeven zoektermen.

Het systeem moet een gebruiker in staat stellen om zich aan te melden met een e-mail paswoord combinatie binnen 1 minuut, startende van op het moment dat de gebruiker op de loginpagina arriveert.

Het systeem moet een de gebruiker met tutor rol efficiënt begeleiden doorheen het aanmaakproces van een lecture.

## Reliability:

Wanneer een administrator een categorie verwijdert, moet het systeem een niet permanente delete query uitvoeren in plaats van een permanente delete query op de database.

De transacties met de database in het systeem moet worden opgebouwd volgens de ACID principes.

Het systeem moet 95% van de tijd elke maand online kunnen blijven tussen 12u en 22u.

## Performance:

Het systeem moet in staat zijn om zoekresultaten voor een gezochte lecture weer te geven in zo dicht mogelijk bij 1 seconde met een maximum van 3 seconden, startende vanaf het moment dat de gebruiker de zoekterm verzend in de webapplicatie.

De database van het systeem moet in staat zijn om minstens 10.000 read transaction te kunnen behandelen per uur.

## Supportability:

Het systeem moet kunnen omgaan met een SQL-database voor dataopslag en databeheer.

Het systeem moet mobile friendly zijn door de webapplicatie schaalbaar te maken afhankelijk van het venstergrootte.

De features in het systeem moeten flexibel worden opgebouwd zodat toekomstige features kunnen toegevoegd worden zonder de bestaande features te breken.

Wanneer het systeem wordt aangepast moeten alle bestaande testen steeds blijven slagen eer het aangepaste systeem kan gereleased worden.

Het systeem moet in staat zijn om in de populairste browsers te kunnen runnen.

# Technologie

Python Flask als backend, React Front end single page webapplicatie.

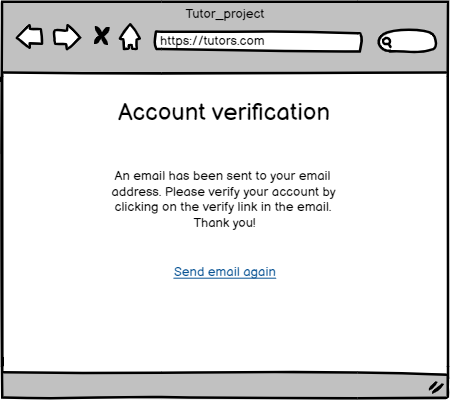
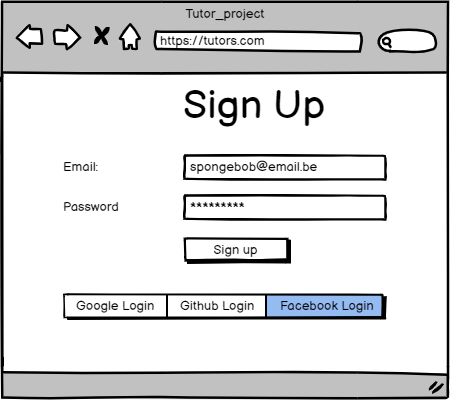
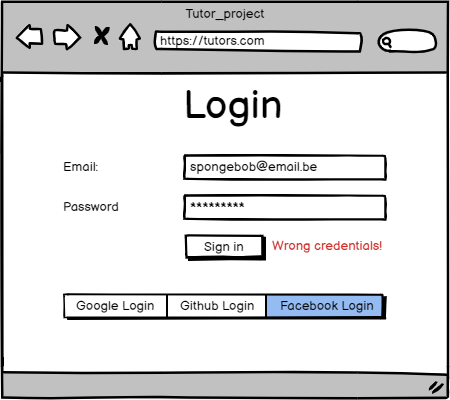
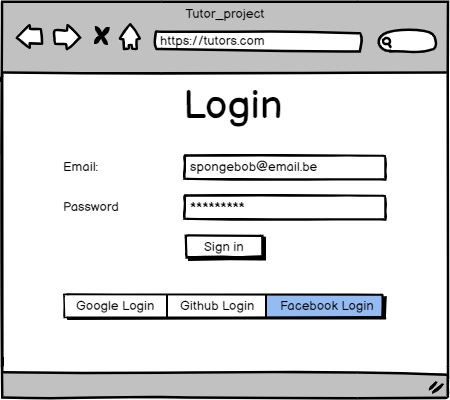
# Effort

|  |  |
| --- | --- |
| Taak | Dagen |
| Requirements Engineering | 3 dagen |
| Analyse | 3 dagen |
| Design | 3 dagen |
| Inloggen | 5 dagen |
| Tutors beheren | 5 dagen |
| Students beheren | 4 dagen |
| Categorieën beheren | 2 dagen |
| Meetings Beheren | 5 dagen |
| Meetings Bekijken | 2 dagen |
| Lectures Zoeken | 4 dagen |
| Lectures Beheren | 4 dagen |
| Inschrijven | 5 dagen |

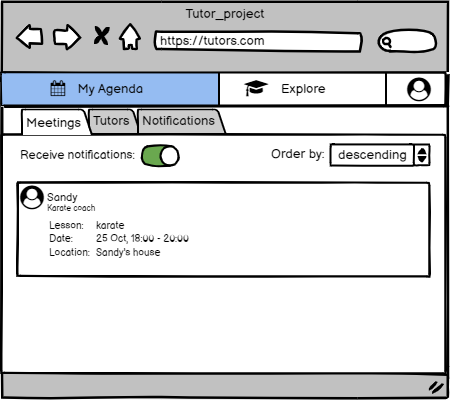
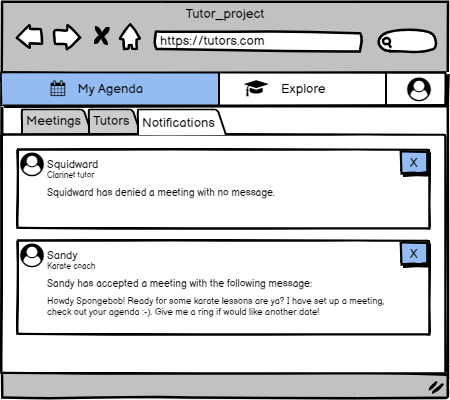
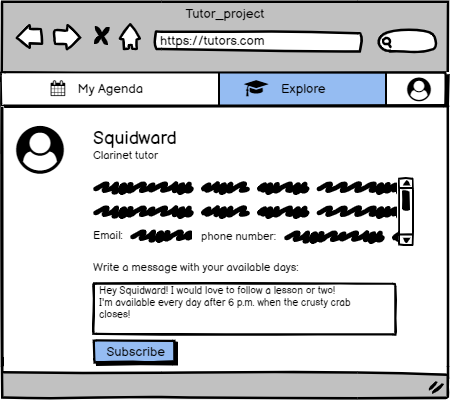
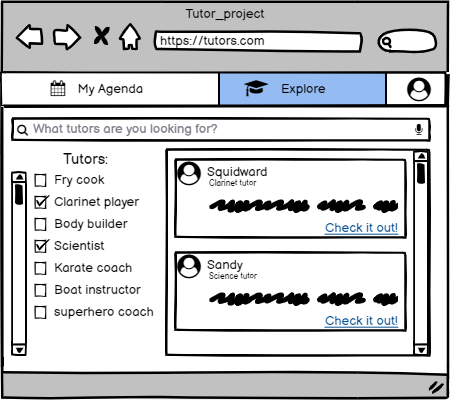
Totaal een ruwe schatting op 45 dagen aan een dagprijs van €500. Een taak bevat de productiecode, de unit testen, het UI design, front end en back end en documentatie.

# GUI Mockups

## Login



## Student: Lectures/Tutors zoeken en mijn agenda



## Tutor: lectures beheren en meetings aanmaken

## Admin:

