MiRoNi

User Stories Aanwezigheidstool

RAC-Academy

Robin Winkels (0909114)
5-2-2023

User Stories Aanwezigheidstool

- Een “bijéénkomst” is bijvoorbeeld een les of een gesprek waarin een docent wil bijhouden welke studenten aanwezig zijn
- Een “check-in” is de actie van het melden van aanwezigheid in de bijéénkomst. Dit wordt door een student zelf gedaan.

De opdracht is om een systeem te bouwen voor het bijhouden van de aanwezigheid van studenten bij bijeenkomsten, waarbij de volgende vereisten gelden:

Samenvatting van de vereisten:

- Als docent wil men een overzicht hebben van aanwezige studenten, aantal studenten die ingecheckt zijn en de mogelijkheid hebben de check-in te sluiten. Ook wil men een apart scherm voor de check-in details kunnen openen. Verder wil men bij check-in een aanvullende vraag kunnen stellen en lessen kunnen plannen met start- en eindtijd en verwachte klassen. Daarnaast wil men later per les zien wie er aanwezig was en per student zien welke lessen hij in het verleden ingecheckt is. Ook moet er een nummer getoond worden met de hoeveelheid studenten die zich hebben ingecheckt.
- Als student wil men snel en gemakkelijk zijn aanwezigheid kunnen aangeven en zien in welke lessen hij verwacht wordt.
- Als beheerder wil men snel klassen en studenten kunnen toevoegen en studenten kunnen verschuiven tussen klassen.
- Als client van de API verwacht men basisbeheer (CRUD) via een REST API: studenten, docenten, klassen en bijeenkomsten.

User Stories:

Als docent:

- Als docent wil ik een overzicht hebben van welke studenten aanwezig zijn in mijn bijeenkomst en wie er nog ontbreken, zodat ik zonder onderbreking op de hoogte ben van het aantal aanwezigen.
- Als docent wil ik een apart scherm kunnen openen met de details voor de check-in en dit op een beamer kunnen tonen, zodat ik real-time inzicht heb in de check-in status.
- Als docent wil ik een aanvullende vraag kunnen stellen bij de check-in, zodat ik een beter beeld krijg van hoe de student er vandaag voor staat.
- Als docent wil ik lessen vooruit kunnen plannen en aangeven wanneer deze starten en eindigen en welke klassen er verwacht worden, zodat ik goed voorbereid ben.
- Als docent wil ik een overzicht hebben van alle lessen waar ik aanwezig was, inclusief de tijden van de check-in en de aanwezige studenten, zodat ik later nog kan terugkijken.
- Als docent wil ik optioneel ook lessen van andere docenten inzien, zodat ik nog beter op de hoogte ben van de les activiteiten.

Als student:

- Als student wil ik snel en gemakkelijk mijn aanwezigheid kunnen aangeven, zodat ik zo snel mogelijk de les kan volgen.
- Als student wil ik inzicht hebben in welke lessen ik verwacht word, zodat ik weet waar ik op word afgerekend.

Als beheerder:

- Als beheerder wil ik klassen en studenten snel kunnen toevoegen, bijvoorbeeld via import vanuit een script of upload van een CSV bestand, zodat ik efficiënt kan werken.
- Als beheerder wil ik studenten kunnen verschuiven tussen klassen, zodat ik flexibel kan werken met de klassen samenstelling.

Als client van de API:

- Als client wil ik de basiszaken van de API kunnen beheren (CRUD), zodat ik snel en efficiënt aanpassingen kan doen.
- Als client wil ik informatie over klassen opvragen via de API, zodat ik snel overzicht heb over de klassen en studenten.

Vereiste functies

1. Authenticatie vereisen: Een gebruiker moet zich authentiseren voordat ze toegang hebben tot de pagina's en REST URLs.

2. Stijlvolle pagina's: Pagina's moeten worden gestyled volgens de RAC huisstijl, maar styling moet ondergeschikt zijn aan andere vereisten.

3. Real-time verversing: Het overzichtsscherm van de docent en het scherm voor aanmelding moeten real-time worden ververst met AJAX.

4. Backend opbouwen: De backend moet worden opgebouwd met Python en Flask of een ander alternatief Python web framework.

5. Datastore: Er moet een datastore zijn met SQLite of een andere database variant, met een ERD en technische implementatie die beoordeeld zullen worden.

6. ORM of SQL: Voor databaseverkeer mag gebruik worden gemaakt van een ORM of direct SQL schrijven.

Optionele Functies:

1. Registratie door docenten: docenten moeten zichzelf kunnen registreren en de beheerder moet deze registraties kunnen goedkeuren.

2. Documentatie van de API met OpenAPI/Swagger specificatie.

3. Authenticatie met sessies en access tokens met beperkte levensduur.

4. Inloggen met een @hr.nl account via OpenAuth (of Oauth). Indien mogelijk, implementatie van inloggen via andere identity providers, zoals Google of Okta, die later aan @hr.nl gekoppeld kunnen worden.