REQUIREMENTS

TA Hiring System

Group 13B

Ali Faruk Yücel
Julie Savelkoul
Martin Mladenov
Mattheo de Wit
Maurits Kienhuis
Winstijn Smit

Contents

Functional Requirements

Must Haves

- The system shall allow users to authenticate using their netid and password, and then issue a token that is used to authenticate the user in other API requests.
- The system shall allow users to register using a netid and password.
- The system shall let students candidate themselves as a TA by sending a request to the API including a short motivation statement and their grade for that course.
- The system shall not accept a TA application for a course if they do not submit a grade or that grade is lower than 6.0.
- The system shall notify students if they have been accepted or rejected as a TA for a particular course by allowing them to submit an API request to retrieve that.
- The system shall let users create a course including a unique course id, start date, and number of students. After the course is created, the user shall be assigned as its responsible lecturer.
- The system shall let students fetch their TA contract including course name, total amount of hours, extra information, and whether the contract is signed, in JSON format from the API.
- The system shall let students sign their TA contract by sending an API request.
- The system shall let lecturers accept students who have applied to TA a course by sending their netid, the respective course id, hours to be worked, and extra contract information to the API. The application status of that student shall then be changed to ‘accepted’.
- The system shall expose the amount of hours and the comments for those hours written by TAs that still need to be approved or rejected for their course for each TA to the responsible lecturers via the API.
- The system shall let lecturers approve or reject working hours declared by TAs of their own courses by posting to the API.

Should Haves

- The system shall let lecturers request a JSON list of all TA applications for one of their courses including netid, grade, status of application, and the existence of former and present TA contracts of students for the same or other courses as well as the rating (if available).

- The system shall not allow a student to candidate as TA later than 3 weeks before the course starts.

- The system shall let lecturers reject students who have applied to TA a course by sending their netid and the respective course id.

- The system shall allow students to withdraw their candidacy unless they can no longer apply as a TA.

- The system shall not allow a student to candidate as TA if they have already applied to be a TA for 3 courses that quarter.

- The system shall not allow a TA to declare hours if this were to go over the limit stated in their contract.

- The system shall let a lecturer rate a TA based on their performance by sending an API request.

- The system shall not allow a lecturer to hire more than 1 TA for every 20 students in the course.

- The system shall allow a TA to specify how many hours they actually worked on the course by posting the course id and the amount of hours to the API.

Could Haves

- The system shall upon request create a list of recommendees, the best candidates based on experience and rating. The returned list shall be in the same format as the list returned by the endpoint to retrieve all applicants.

- The system shall have the functionality to send emails.

- The system shall notify the students of their TA approval via an automatically sent email.

- The system shall let the responsible lecturer of a certain course add other users as responsible lectures of that course with the same rights via an API request.

- The system shall allow any responsible lecturer to remove other responsible lecturer other than themselves from that course via an API request.

Won’t Haves

The following items will not be implemented in our application:

- The system will not use TU Delft Single-Sign-On as an authentication method.
- The system will not present the user with a GUI.

Non-functional Requirements

- The system shall be written in Java 11
- The system shall be built with Spring Boot and Gradle.
- The system shall be scalable and built with microservices.
- The system shall be modular such that it can be expanded later.