Exam grading requirements

SEM group 10

November 2020

1 Functional Requirements

For the grade management system we're going to develop, the requirements regarding functionality and service are grouped under the Functional Requirements. We'll use the MoSCoW model for prioritizing requirements.

1.1 Must Haves

- Teachers need to be able to create course.
- Courses need a course code and a name.
- Multiple grade categories per course (midterm, endterm, computer exam, resit, . . .).
- Any teacher can give a grade to any student
- Students shall be able to do any exam of their choosing.
- Every course should have a grading formula applying weights to every grade category to calculate the final grade.
- Students can only pass a course if they at least get a 5.75 as their final grade.

1.2 Should Haves

- Grades cannot be lowered, but can be increased
- Teachers aren't able to receive grades
- Students should only be able to do an exam once
- Teachers should be able to view pass rates for a course.
- Teachers should be able to view the mean grade for a course.
- Students should be able to view courses they passed

1.3 Could Haves

- Students and teachers could see extra stats about grades (means, variances, standard deviations, ...)

1.4 Would / Won't Haves

- The grading system shall be interacted with via a GUI.
- The grading system shall have an integration with *TU Delft*'s actual *Single Sign On* API for authentication.

2 Non-functional Requirements

- The system should be extendable (i.e. modular).
- It should allow for easy integration via its API.
- It should be easily scalable (using the 'microservices' design pattern).
- It should allow for interactions via its API and shouldn't have a GUI.
- *Java 11* should be used for implementation.
- *Spring boot* and *Gradle* should be used to accelerate building the system.
- The system should utilize *Spring security* to safely store user credentials and to authenticate users.
- Users should be authenticated using their *NetID* and their password