Easy Grade Manager

Project Topic Specification – Web Centric Development 2019/20

Carolina Faria, Kian Schmalenbach

Background and Motivation

In many study disciplines, including the field of Information Systems, some university classes include periodical student assignments as part of the evaluation and grading process. For instance, students might be required to weekly hand in homework, possibly created in small groups. These pieces of work are usually evaluated and eventually graded or marked by a tutor, who can, but must not necessarily be the same person as the person who teaches the course. Furthermore, the results of the students will typically either influence their final grade or determine whether they are allowed to proceed with the exam or a subsequent course.

The setting described above requires the incremental collection and storing of marks or grades of many students or student groups over a certain period of time as well as their final evaluation to determine the students' grade and, hence, their success or failure. Moreover, additional challenges within the scenario might involve the existence of more than one tutor, students joining or leaving a course, students demanding for intermediate results, or a complex definition of the final grading criterion (e.g., "*at least 30% of the marks in each assignment and 50% of the overall marks*," etc.).

In many cases, the required data is collected and evaluated using paper-based systems or regular spreadsheets, leading, among others, to the following disadvantages:

- Many "bookkeeping" activities, such as adding/removing students, changing the number of assignments, changing the grading procedure, etc. require manual and possibly redundant actions. This might include adding/erasing rows/columns, (re-)calculating numbers, etc.
- Information might be lost if the paper or spreadsheets are accidently lost or deleted.
- Students' requests, e.g. concerning their intermediate progress, must be answered manually.
- In the case of several tutors, the collected data needs to be assembled and merged manually.

Purpose and Goal of *EasyGradeManager*

To overcome the above-mentioned disadvantages, we propose to implement a server-client based web application named *EasyGradeManager*, which stores data about the assignments and marks in a central database and provides user interfaces for teachers and tutors as well as for students to access and manipulate the data. An administrative user, typically the teacher of the corresponding course, can create entries for classes and tutors, assign her/his students to those, and define the evaluation criteria for the classes. The tutors, who evaluate and grade their students' performance, can then input the corresponding data, allowing the teacher to conduct the final grading at the end of the course. Finally, the students can create student groups (in the case of teamwork), observe their learning progress, and see their final evaluation.

This solution overcomes the previously identified disadvantages as follows:

- All "bookkeeping" activities are now conducted by the software according to the users' inputs.
- Since all information is safely stored in a central database, it is highly unlikely to get lost.
- Students can obtain all necessary information directly by logging on to the system.
- Manual merge of data is not required since all data is already stored in a central database.

Detailed Functionality of *EasyGradeManager*

The following section briefly describes the essential use cases identified for *EasyGradeManager*.

Case 1: Creating a Class

Conducted by:	Teacher
Description:	1. The teacher creates a new entry for a class she/he is teaching 2. The teacher creates *k* assignments and defines evaluation criteria for each assignment 3. The teacher defines a final evaluation criterion derived from those for each assignment
Remarks:	• The teacher can later add, change, or delete assignments and evaluation criteria as long as no corresponding data has been put into the system yet • The teacher can also reset a class to its initial state when a new teaching period (i.e. a new term) begins

Case 2: Assigning Tutors and Students to a Class

Conducted by:	Teacher
Description:	1. The teacher selects a class from his list of classes 2. The teacher adds *m* tutors to the class by creating or selecting the corresponding entries (potentially her-/himself if he also acts as a tutor) 3. The teacher assigns one or more time slots to each tutor 4. The teacher assigns *n* students to a tutor's time slot each by creating or selecting the corresponding entries
Remarks:	• Instead of assigning the students to the tutor, the students might assign themselves to a tutor's time slot if desired • The assignment can be adjusted later if necessary • On class reset (see *case 1*), the teacher can decide to keep or discard the tutors and time slots from the previous term

Case 3: Creating a Student Group

Conducted by:	Student
Description:	1. The student selects a class from the list of her/his classes 2. The student selects *k* students from her/his time slot to form a group

Case 4: Grading a Student's Assignment

Conducted by:	Tutor
Description:	1. The tutor selects a student or student group from one of her/his time slots 2. The tutor assigns a grade to the student's or student group's assignment

Case 5: Performing Student's Final Evaluation

Conducted by:	Tutor, Teacher
Description:	1. The tutor confirms to the system that she/he finished grading her/his students' assignments 2. The system performs the students' final evaluation and notifies the teacher once all students of a class are evaluated 3. The teacher adjusts or confirms the evaluation 4. The students can see their final evaluation in the system

Note: In addition to the functionality described here, it is possible at any time to see the intermediate state of a class, an assignment, a student etc. according to the user role and to change personal data. Moreover, we assume that the creation and deletion of all accounts are managed by an administrative entity (e.g. the university administration), which is not part of the described software.