Requirements Analysis Document

Lucía Asencio y Juan Riera

February 16, 2017

1. Introduction

   1.1. Purpose of the system
   The system is an application with educational purposes. It offers a communication and evaluation platform between professors and students.

   1.2. Scope of the system
   The application should have two types of users: students and teachers. There will be courses, created by teachers, which will be divided in units, subunits, subsubunits and so on. In each unit there will be exercises and notes all of which are organized by subjects.. Therefore it also serves as a database, that also provides a statistic calculation system detailed below.
   Students will have as many accounts as te number of students using the application, however, there will only be one teacher account for all teachers.

   1.3. Objectives and success criteria of the project
   The objective is to create a user-friendly application that meets the functionality detailed in the next section.

   1.4. Definitions, Acronyms and abbreviations
   Although sometimes in this document we will talk about subjects and sometimes about courses, we will be referring in both cases to the same thing.

2. System Description

   2.1. Functional Requirements The application has two types of users:

      2.1.1. User type 1: student

         2.1.1.1. Log in to the application

         2.1.1.2. View available courses and send an application to a course and get e-mail notifications when they are accepted or declined.

2.1.1.3. Access to each subject unit/subunit/subsub... and unit notes
Each item will only be visible after certain date (decided by teacher) if the student belongs to the course and is not expelled.

2.1.1.4. Access and solve exercises, each of which belongs to a unit, which belongs to a course. Exercises will only be visible and solvable after and before certain dates decided by teacher
A student may leave a question unanswered, and may quit the exercise without sending it whenever he wants, his answers will not be saved.
Once a student finishes an exercise, he cannot do it again.
Each exercise will have a relevance specified by the teacher, this relevance should be displayed at any time. After the deadline of an exercise, students can view their marks in that exercise (normalized from 0 to 10), as well as correct answers and the answers they wrote.

2.1.1.5. See their mark in a course up to that point, normalized from 0 to 10

2.1.1.6. Communicate with teachers and other students via e-mail

2.1.2. User type 2: teacher

2.1.2.1. Log in to the application

2.1.2.2. Get notifications via email when a student applies to a course.

2.1.2.3. Accept/decline students applications to a certain course

2.1.2.4. Set up courses, unit, subunits, subsubunits...
Also, write and add notes to each unit, subunit etc. This notes will be plain text.

2.1.2.5. Use an exercise maker to add exercises to units
This maker will set the unit where the exercise belongs, the two dates between which it is doable, the number of questions and the type of each of them (multiple choice/one choice/true-false/open answer), the order of the questions (which can be chosen to be random or a fixed order, also chosen by the teacher), the penalty for each wrong answer (if any), the value of the exercise for the final course mark (all of which don't have to sum 10, since they will be normalized), the text of the question, the points of each question, the correct answer and the possible answers in the multiple

choice/one choice type of questions
Teacher can re-set up the whole exercise (excepting for the dates: between which only the deadline can be changed and only if it is postponed), if no student has already done the exercise. (Students will get a notification whenever the exercise turns visible)

2.1.2.6. Make notes and exercises (and, optionally, courses) visible or invisible. This visibility may also be scheduled.

2.1.2.7. Access to all the information of a student, except for his log in data and student's emails. This means teachers have access to every mark of every student, in every exercise and every course, every answer he has given to every question in every exercise he has finished. He also will have access to the course average marks of every student.

2.1.2.8. Access courses list.

2.1.2.9. View statistics
Statistics per subject : all the normalized marks, average mark, number of fails and number of students who have passed.
Statistics per exercise : marks, average mark for each exercise.
Statistics per question : not answered, answered, correct answers, wrong answers.

2.1.2.10. Expel students (and readmit them in the course), also access to a list of the expelled students

2.1.2.11. Communicate with students via e-mail

2.2. Non-functional Requirements

2.2.1. The application should be prepared to be used by the common user who does not have computer knowledge further than opening google.

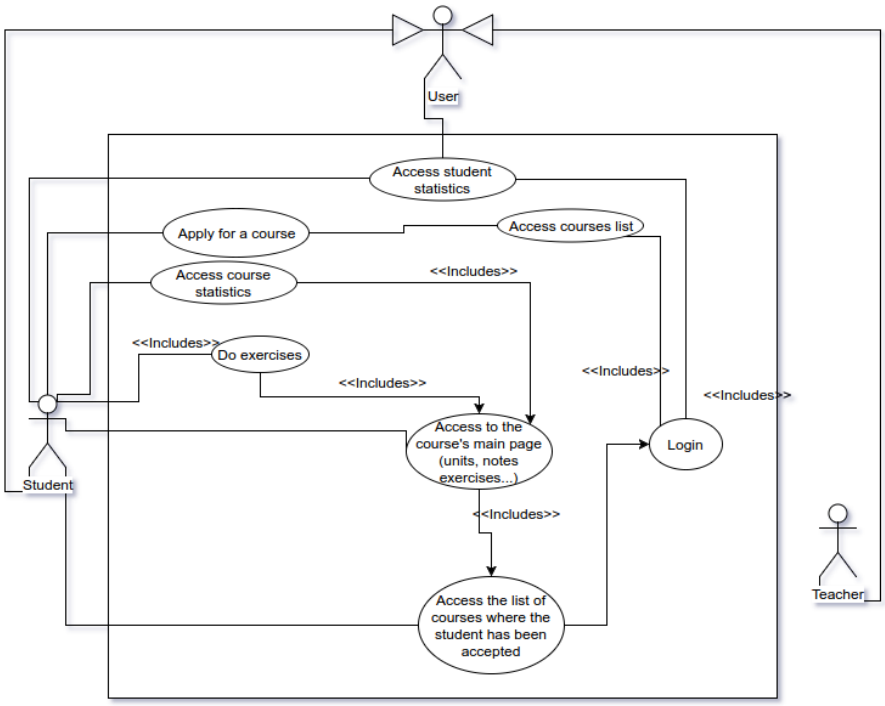
2.2.2. The statistics calculations should not take longer than a few seconds.

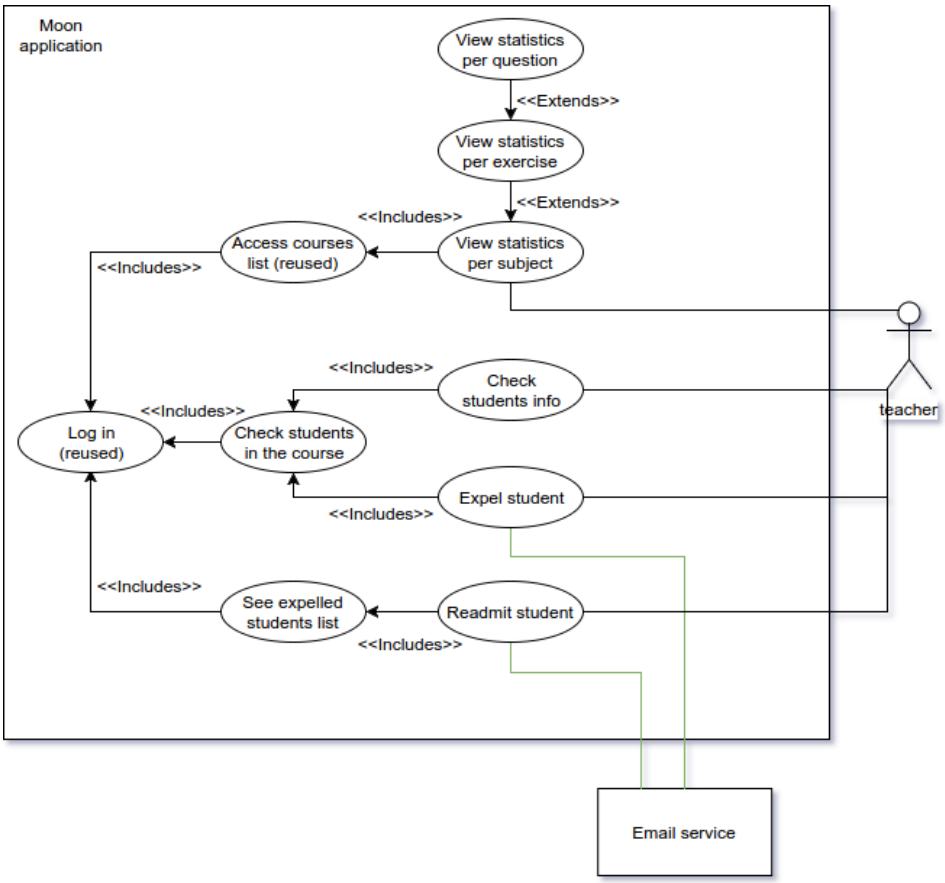
3. Use Cases

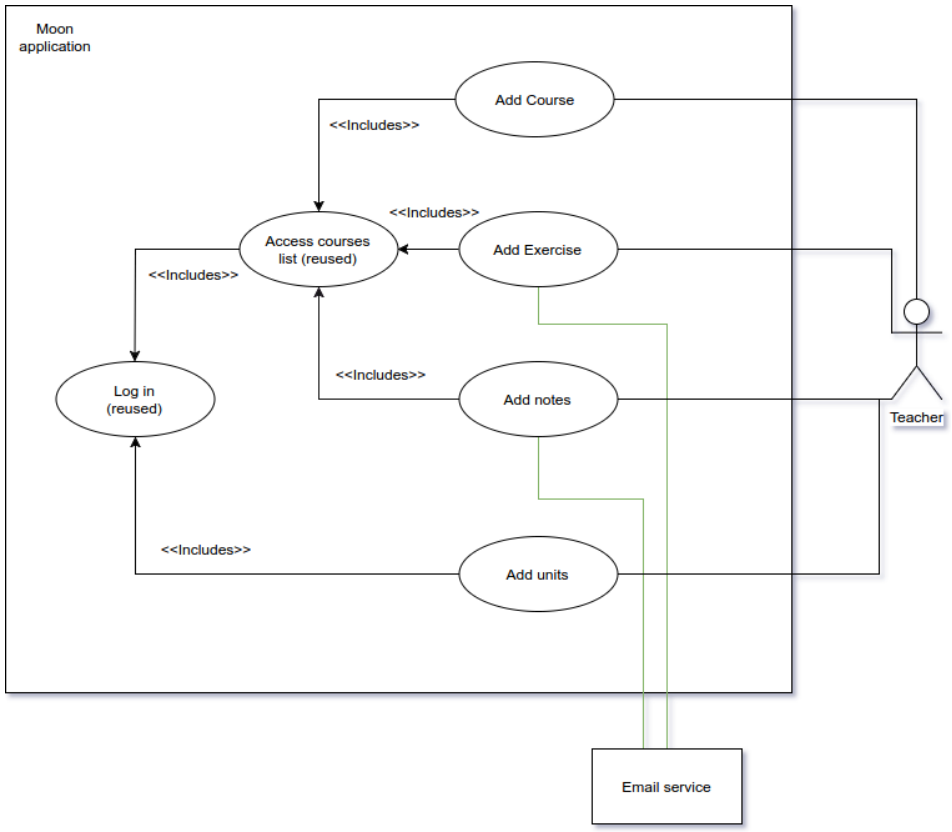
3.1. Use Case diagram

We divided our diagram in three parts so that it was easily understandable.

- ⋆ First one is for student use cases
- ⋆ Second one is for teacher use cases
- ⋆ Third one is also for teacher use cases




Moon application
View statistics per question
<<Extends>>
View statistics per exercise
<<Extends>>
<<Includes>>
Access courses list (reused)
View statistics per subject
<<Includes>>
<<Includes>>
Check students info
<<Includes>>
Log in (reused)
Check students in the course
Expel student
<<Includes>>
<<Includes>>
See expelled students list
Readmit student
<<Includes>>
teacher
Email service



3.2. Use case descriptions

3.2.1. Use case: View statistics per subject

- Primary Actor: Teachers
- Stakeholders and Goals: Teachers who want to check general statistics (this means, not student specific)
- Preconditions: the user has logged in as a teacher, accessed the courses list and selected one.
- Success Guarantee: the needed statistics are displayed in some seconds.
- Main Success Scenario:
  - 1. The teacher selects "Course statistics and data"
  - 2. The following data are displayed:
    - Average mark in the subject.
    - Number of students who have passed in the subject.
    - Number of fails in the subject.
    - List of all the student's normalized marks in the subject.

    - List of exercises in the subject.
- Extensions:
  - 2a. The user clicks on an exercise.
  - 3a. The following data is displayed:
    - Average mark in the exercise.
    - Number of students who have passed the exercise.
    - Number of fails in the exercise.
    - Relevance of the exercise.
    - List of questions.

    The teacher may also click on one question from the list:

  - 3b. The teacher clicks on one question from the list displayed in 3a.
  - 4b. The following data is displayed:
    - Number of correct answers.
    - Number of incorrect answers.
    - Number of students who did not answer.
    - Correct answer.
    - List of answers given by the students.
- Special Requirements: Quick response (a few seconds) in the process of calculating and displaying the data.
- Frequency: no concurrency, one user at a time.

3.2.2. Use case: Teacher makes an exercise

- Primary Actor: Teachers
- Stakeholders and Goals: Teachers who want to use the exercise maker so that they can evaluate the students
- Preconditions: Have logged in as a teacher
- Success Guarantee: The exercise will be added to the course exercises.
- Main Success Scenario:
  - 1. Teacher selects "Add exercise".
  - 2. A list of available courses is displayed.
  - 3. Teacher selects the course where to add the exercise.
  - 4. A list of units is displayed.
  - 5. Teacher selects the unit where to add the exercise.
  - 6. Teacher selects the dates where the exercise will be visible.
  - 7. A list with the four available types of questions (described in System Description 2.1.2.5)is displayed, teacher selects "multiple choice".

- 8. Teacher writes down how many questions the exercise will have.
- 9. Teacher selects the order of the questions in the exercise (random or not).
- 10. Teacher selects the penalty, if any, for each wrong answer.
- 11. Teacher selects the value of the exercise in the final mark.
- 11. For each question, teacher decides how many statements to show,writes them down and assigns a truth value to each statement.
- 12. Also, adds a value to each question in the exercise.
- 12. When all the info is filled in, teacher selects "Save exercise".

· Extensions

- Extension 1: one choice test
  - 7a. Teacher selects "one choice" exercise.
  - 11a. For each question, teacher decides how many statements to show, writes them down and *selects one* true statement.
- Extension 2: Teacher selects "true/false" exercise.
  - 7b. Teacher selects "true/false" exercise.
  - 11b. For each question, teacher *writes down one* statement and assigns a truth value to it.
- Extension 3: Teacher selects "open answer" exercise.
  - 7c. Teacher selects "open answer" exercise.
  - 11c. For each question, teacher writes down *one question* and writes down *one answer*.

· Technology and Data Variations List: The new exercise is added to the database.

· Frequency: No concurrency

· Open Issues

3.2.3. Use case: User applies for a course

· Primary actor: Student

· Stakeholders and goals: Students who want to apply for a new course

· Preconditions: To have logged in the application.

· Success Guarantee: an email will be sent back with the teacher's answer

· Main Success Scenario

- 1. Student selects "Application".

  - 2. A list of available courses is displayed.
  - 3. Student selects the course he wants.
  - 4. An "Are you sure you want to apply for this course?" message is displayed on screen.
  - 5. Student selects "Yes".
  - 6. Student selects "Exit application mode".
  - 7. A notification with the application is sent via e-mail to the teacher.
- Extensions
  - Extension 1
    - 5a. Student selects "No".
    - 6a. Student selects "Exit application mode".
- Technology and Data Variations List: If teacher accepts, the Courses Students List is modified.
- Frequency: no concurrency, one user at a time.