Víctor García Carrera
Carlos Ignacio Isasa Martin

victor.garciacarrera@estudiante.uam.es
carlos.isasa@estudiante.uam.es

Requirements Analysis Document

Application: Coorse

Date: 14-2-2017

Víctor García Carrera
Carlos Ignacio Isasa Martin

victor.garciacarrera@estudiante.uam.es
carlos.isasa@estudiante.uam.es

Index

Víctor García Carrera
Carlos Ignacio Isasa Martin
victor.garciacarrera@estudiante.uam.es
carlos.isasa@estudiante.uam.es

1. Introduction

1.1 Purpose of the system

Coorse enables professors to create courses and manage them. They can create units and subunits, and inside them they can create exercises and notes. Students have to apply for this courses and the professor must decide whether he accepts him or not. Students have to complete exercises and they get the corrections and marks after that.

1.2 Scope of the system

Our application is expected to provide an easy access platform for education centers. It helps professors organize their courses and it gives an easy way to students to access resources such as notes or exercises.

1.3 Objectives and success criteria of the project

The success of this application is achieved by completing the following points:

- Creation of courses, units, subunits, notes and exercises by professors
- Students being able to do exercises, get their marks and their exam corrected
- Store and show some statistics on every course about the progress of every student to students and professors

1.4 Definitions, Acronyms, and abbreviations

The language used throughout the document is exempted of technicisms and any other kind of word or aconym that may be not understood.

Víctor García Carrera
Carlos Ignacio Isasa Martin
victor.garciacarrera@estudiante.uam.es
carlos.isasa@estudiante.uam.es

2. System Description

2.1 Functional Requirements

In this application, there are two types of users:

- **2.1.1 Student**: The student user launches the application to see the available courses at the moment, and to work and use the information displayed on the courses where he is registered. He also can see his progress on every course.

Here are all the functional requirements for every student's account:

2.1.1.1 *Credentials*: Every student account has a name, ID, user name, password and email. This information is given to the students, and it can't be changed (maybe as an optional requirement).

2.1.1.2 *Register into any existing course*: Every student must be able to enroll in any course he wants, although the professor is the one who accept him in the requested course. After entering the course, the student can't leave it, he just can be expelled. If he is expelled, he can apply for entering the course again, and the professor is able to readmit him or not.

2.1.1.3 *Do exercises and read notes*: When a student is accepted into a course, he can interact with all the information displayed of that course. He can read notes, do exercises just one time (when he is able to, just in specific and given days), although he can exit the exercise without saving what he have done (just by canceling the exercise) and after that, restarting it again (if he access before the expiration time of the exercise). When he finally finish the exercise, his answers are saved (so the professor can see what he has answered and the student is able to see in a future what he answered).

After doing an exercise, and after the expiration day and time, the student is able to see whenever he wants his mark on the exercise, every questions with the text of the question, what he answered to that question and the correct answer, and it's punctuation on every question.

2.1.1.4 *Receive relevant notifications*: Every student receives an email to his email account (the one given with his credentials) when he is accepted into a course (that he has applied for it before), when a unit is opened (it was hidden before or maybe it didn't exist and the professor has created it) and when a note or an exercise of a course where he is

enrolled appear (the same that before, maybe it was hidden or the professor has just created it)

2.1.1.5 *See his progress on every course*: when the student access to any of the courses where he is enrolled, he is able to see his marks on the exercises that he have done, and his global mark of that course. He can also see the relevance of every exercise on the global mark.

- **2.1.2 Professor**: The professor user launches the application in order to edit the courses available, making exercises and deciding what he wants to be open for students and what not. He also can see different stats of every course, which are detailed below.

Here are all the functional requirements for the professor account:

2.1.2.1 *Credentials*: As there is only one professor account, every professor who wants to access *Coorse* must login with the same credentials. This account just has a user name and a password. As a consequence of this, *Coorse* just admits one professor at a time. This user name and password can't be changed.

2.1.2.2 *Create courses, units (and subunits, which are units that are inside an existing unit), notes, and exercises*: To create a course, the professor just needs to give it a name. After that, the course will have been created, and it will appear on the main page of *Coorse* (the first page you see after logging in). There is no limitation of number of students. When a course is created, students can apply for entering that course. Every course will have a list of students enrolled to it (which the professor can see) that changes if a student is expelled or if a student is (re)admitted.

To create an unit, the professor just needs to give it a name and decide where he wants to create it (inside an specific course or even inside another unit, which is obviously inside a course). Inside an unit, there can be exercises, notes and other units (which are created the same way). Notes are just plain text, and the professor can edit or delete them whenever he wants. To decide where to create notes, is the same as with units. To create an exercise, there is an exercise editor available (which is explained in detail below).

Units, exercises and notes can be hidden or not. This means the professor chooses if he wants a note/exercise/unit to be visible for students or not, so if it's invisible, they can't see or access them. If an unit is hidden, everything inside it will be hidden.

2.1.2.3 *Use an exercise editor*: In order to make it easier for professors to create exercises, *Coorse* has an exercise editor where professors can create easily and comfortably their exercises. The professor decides the name of the exercise, where to create it, the relevance on the global mark of the course of that exercise, the relevance of

every question of the exercise (the number of points of every question), when students are able to do the exercise (start day and time) and when not (expiration day and time), the type of questions (there are 4 types; unique choice, true-false, multiple choice and text questions where students must write), the solution (or solutions) of them, the different possible answers of the questions, the penalty for not answering correctly a question (maybe you just have 0 points on a question that you answered wrongly or maybe it can subtracts 1 point of you exercise mark) and whether he wants a certain order for the questions for everybody or a random order of the questions for each student (he can also do that with the order of the different answers of a question).

Your answer of a question is incorrect if it isn't the exact solution of the question (which was previously given by the professor). In the multiple choice type of questions, there is an option for student to clean his answers. The expiration date can be changed by the professor at any moment, and after the expiration date, students mustn't be able to do the exercise.

Every exercise is punctuated from 0 to 10. As the professor chooses a penalty for not answering right a question of the exercise, although the final mark of the exercise is under 0 (cause he has chosen a penalty that subtracts points), the final mark of the exercise of that student will be a 0. When students do an exercise (and finish it, saving their answers), it will be automatically corrected (as the professor has given all the information needed for this), and different statistics of the exercises done by students must be stored and showed before to the professor (those are explained below).

2.1.2.4 *Modify an exercise*: If an exercise is already created, the professor is able to modify any part of it by using the exercise editor, for example the expiration date. If a student has already done an exercise (so their answers have been saved), the professor can't change anything about questions(before that, there is no problem).

2.1.2.5 *Admit students in courses*: The professor, after a student has applied for entering a course, can decide to accept his request or to decline it. He is able to expel a student, and to readmit him later. The professor has a list with all this requests from students.

2.1.2.6 *See the progress of his students*: On every course, the professor is able to see some statistics, based on the answers of the exercises sent by the students enrolled in that course. This statistics are from each exercise and from the course in general. In a general view, it must appear the global mark and the different marks of the exercises of each student enrolled in that course. In every exercise, statistics provided to the professor consists on information about each question, as the number of students that answered, the number of students that didn't answer, the number of students that answered correctly and the number of students that answered incorrectly.

Coorse application works with up to one student and one professor at a time. All changes made are saved, so when you login again, everything stays the same way it did before closing it.

2.1.3 Optional Capabilities

There are several functionalities that may be implemented. Just to show a few of the possible ones, *Coorse* can calculate, on every exercise, the average mark of an exercise based on the marks of all students. Proffesor may be able to choose and change the settings of the notifications sent to students, so whenever he opens (make it visible for students) an exercise or a note, maybe he can change sending an email every time he do that. It is possible also to create forums where people can ask for information or help, and maybe students can send messages to other students or to teachers.

2.2 Non-functional Requirements

2.2.1 Usability:
Almost everything is managed through the mouse. To open a course you must click on it. The same applies to access units, subunits and exercises. The only situation where you are required to use a keyboard is if the exercise asks for it, giving you an open question. Also it may be implemented any kind of searcher in order to navigate through the courses and units faster (or to access somewhere you don’t know exactly where it is, just by writing its name).

2.2.2 Reliability:
The application must handle as many courses as the professors want, a group of 10-15 professors and a group of 100-1000 students.

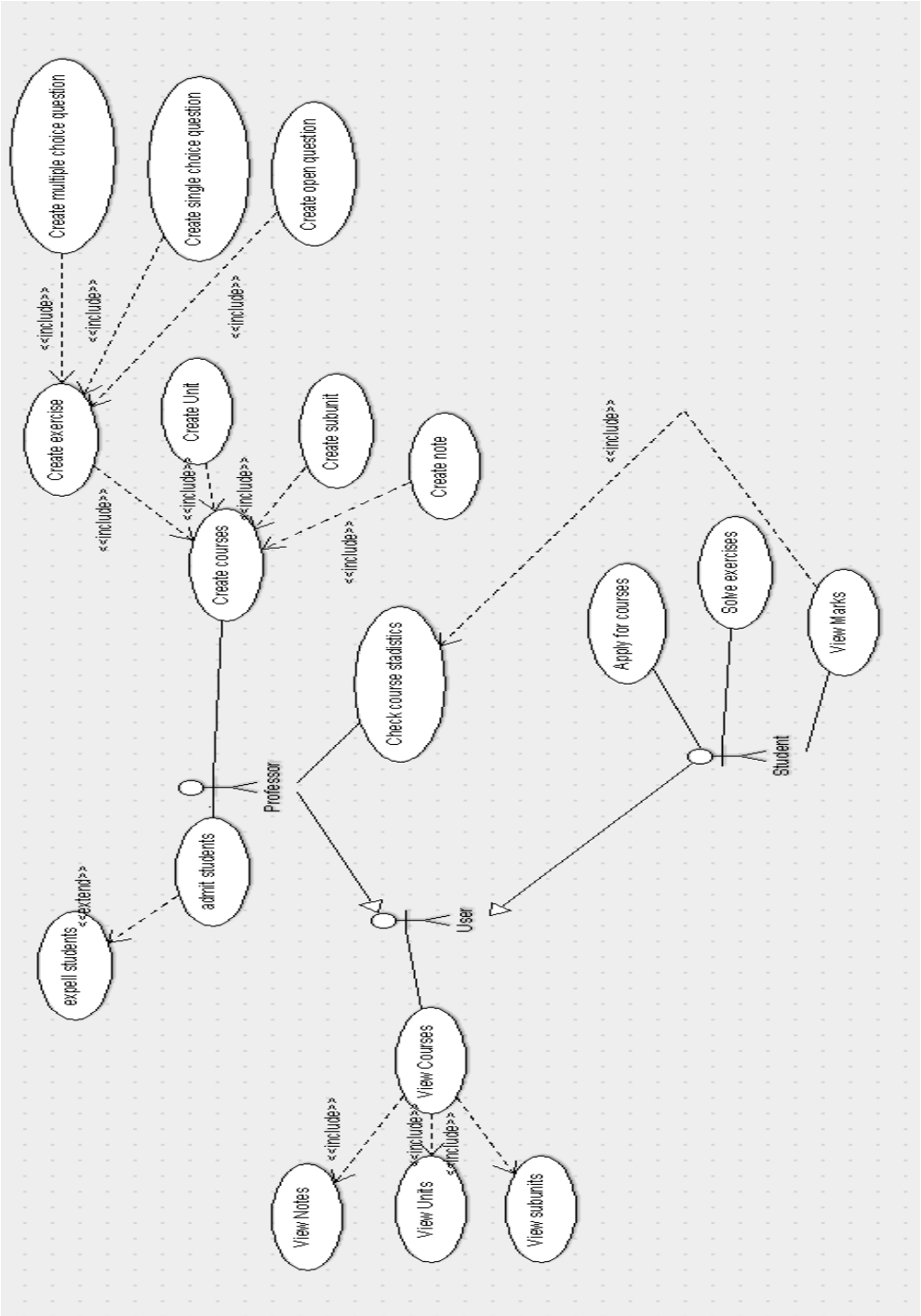
2.2.3 Performance:
The application shouldn't have any performance constraints and user interaction shouldn't cause any visible disturbance

2.2.4 Maintainability and portability:
As the application *Coorse* is going to be implemented using java language, it is compatible with Windows, Linux and lots of different operative systems.

Víctor García Carrera
Carlos Ignacio Isasa Martin
victor.garciacarrera@estudiante.uam.es
carlos.isasa@estudiante.uam.es

3. Use Cases

3.1 Use Case diagram


Create multiple choice question
Create single choice question
Create open question
Create exercise
Create Unit
Create subunit
Create note
Create courses
Check course stadistics
Apply for courses
Solve exercises
View Marks
Professor
Student
admit students
expell students
User
View Courses
View Notes
View Units
View subunits
<<include>>
<<extend>>

3.2 Use case descriptions

3.2.1 Use Case: Solving an exercise

Primary Actor : A student
Stakeholders and Goals:
Student: Doing an exercise
Professor: Makes the exercise, and reviews the mark
Preconditions:
The student must be registered in the course, the exercise must be visible.
Success guarantee (Post-conditions):
The student's answers are recorded and matched with the correct solution. A notification will be send to the student with his mark.
Main Success Scenario:

- The student selects an exercises
- The student completes all the questions
- The system compares the student's answers with the solutions provided by the professor
- The system notifies the student with the mark

Extensions (Alternative paths):
--
Special Requirements:
The student must solve the exercise before the expiration date
Technology and Data Variations List:
--
Frequency:
One for each student and each exercise.
Open Issues:
--

3.2.2 Use Case: Creating an exercise

Primary Actor : A professor
Stakeholders and Goals:
Professor: Making the exercise
Preconditions:
The professor must be log on with the professor account
Success guarantee (Post-conditions):
The exercise must be created and the solution provided
Main Success Scenario:

- The professor opens the desired unit in the desired course.
- The professor creates the questions, choosing between multiple choice questions with only one answer, multiple choice questions with several answers and open questions.
- For each question, the professor provides the question itself, the correct answer and the weight on the mark.
- The professor sets an expiration date.
- The professor decides if the exercise is visible or not.

Extensions (Alternative paths):

The professor chooses the type of question for each one. The professor decides if he wants the exercise to be visible or invisible.

Special Requirements:

--

Technology and Data Variations List:

--

Frequency:

As many as the professor wants

Open Issues:

Implementing other question types, such as essay questions.

3.2.3 Use Case: Modifying an exercise

Primary Actor : A professor

Stakeholders and Goals:

Professor: Modifies the exercise

Preconditions:

The exercise must exist. No student have done the exercise.

Success guarantee (Post-conditions):

The question or questions are changed successfully

Main Success Scenario:

- The professor selects an exercises
- The professor changes the question and the correct answer
- The exercise is saved with the changes

Extensions (Alternative paths):

The professor may delete the question entirely.

Special Requirements:

No student must have done the exercise

Technology and Data Variations List:

--

Frequency:

As many as the professor wants

Open Issues:

--

3.2.4 Use Case: Viewing marks as a student

Primary Actor : A student
Stakeholders and Goals:
Student: views the mark
Professor: must have posted an exercise
Preconditions:
The student must have solved an exercise
Success guarantee (Post-conditions):
The student views his mark
Main Success Scenario:

- Student selects course
- The student clicks on "marks"
- Each mark appears independently
- The student views the marks

Extensions (Alternative paths):
The student might want to see his mistakes. He can click in the concrete exercise to do it
Special Requirements:
There is at least one exercise solved
Technology and Data Variations List:
--
Frequency:
As many as the student wants
Open Issues:
Maybe the student wants to view his average on this course.

3.2.5 Use Case: Viewing marks as a professor

Primary Actor : A professor
Stakeholders and Goals:
Student: must have solved an exercise
Professor: view the exercise
Preconditions:
There are marks to review
Success guarantee (Post-conditions):
The professor views the marks

Main Success Scenario:
- Professor selects course
- The professor clicks on "marks"
- The professor selects an exercise inside the marks
- Each mark appears independently for each student
- The professor views the marks

Extensions (Alternative paths):
The professor can view each student mistakes such as if he was one of them
Special Requirements:
There is at least one exercise solved by at least one student
Technology and Data Variations List:
--
Frequency:
As many as the professor wants
Open Issues:
Maybe the student wants to view his average on an exercise or a course

3.2.6 Use Case: Applying for a course

Primary Actor : A student
Stakeholders and Goals:
Student: applies por the course
Professor: gets the application
Preconditions:
None
Success guarantee (Post-conditions):
The student sends the application
Main Success Scenario:
- Student selects course
- The student clicks on "apply course"

Extensions (Alternative paths):
--
Special Requirements:
There has to be a course
Technology and Data Variations List:
--
Frequency:
As many as the student wants
Open Issues:
--

3.2.7 Use Case: Accepting/declining an application

Primary Actor : A professor
Stakeholders and Goals:
Student: applies por the course
Professor: accepts or declines the application
Preconditions:
A student must have applicated
Success guarantee (Post-conditions):
The student gets admited or rejected
Main Success Scenario:

- Professor selects course
- Professor selects "pending applications"
- Professor selects "accept" or "decline"

Extensions (Alternative paths):
--
Special Requirements:
There has to be a course
Technology and Data Variations List:
--
Frequency:
As many as students want to apply
Open Issues:
Maybe the system could send a notification to the student applying

4. Mockups

When the student or the professor login *Coorse*, a page with all the existing courses will appear, although the are some differences between both users pages as this is the “main page” of the application (and they have different functionalities)