UNIVERSITY OF THESSALY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Master Thesis

"Implementation of a Platform Supporting Programming Assignments in University Courses"

Author

Machairas Dimitrios

Supervisor

Antonopoulos Christos, Assistant Professor

Committee Members

Bozanis Panagiotis, Associate Professor

Vavalis Emmanouil, Professor

Volos, September 2014

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Μεταπτυχιακή Διπλωματική Εργασία

"Υλοποίηση Πλατφόρμας για την Υποστήριξη Προγραμματιστικών Εργασιών σε Πανεπιστημιακά Μαθήματα"

Συγγραφέας

Μαχαίρας Δημήτριος

Επιβλέπων Καθηγητής

Αντωνόπουλος Χρήστος, Επίκουρος Καθηγητής

Μέλη Επιτροπής

Μποζάνης Παναγιώτης, Αναπληρωτής Καθηγητής

Βάβαλης Εμμανουήλ, Καθηγητής

Βόλος, Σεπτέμβριος 2014

UNIVERSITY OF THESSALY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

"Implementation of a platform supporting programming assignments in university courses"

A Thesis presented

By

Machairas Dimitrios

APPROVED BY:

Antonopoulos Christos, Assistant Professor

Bozanis Panagiotis, Associate Professor

Vavalis Emmanouil, Professor

Volos, September 2014

Abstract

As the time passes, the number of university students is rapidly increasing, and the whole procedure of the submissions management is getting much more difficult. Each university department has its own needs, in accordance to the subject that is being taught. For example, a computer science department, may need web applications for programming courses, where professors will be able to assign programming tasks to their students. Due to the fact that it concerns programming courses on different programming languages, when students submit their tasks, they should be able to view compilation results and results from test cases. Moreover, since it is a rather common phenomenon that many students try to cheat and copy from each other, it is vital that such an application use a kind of plagiarism system, which will detect the same code parts. Furthermore, after the deadline, professors should have the ability to assign each submission to a grader for grading and commenting. Consequently, when grades and comments are ready, professors should be able to publish them so that the students can directly view them.

The purpose of this project is to develop a web platform, which can support programming assignments (mainly in C programming language), and fulfill the functionality which was described in the previous paragraph. In some words, we develop a web platform, where there are 4 kinds of users: *Administrator*, *Professor*, *Assistant* or *Grader* and *Student*. Each user has his own functionality. In particular, every kind of user can administer courses, labs and homeworks in his own way, apart from administrator, who can also administer even users.

Last but not least, we used MVC 5 architecture and Bootstrap 3 framework in order to develop our platform. We have also used SQL for our database, Cygwin software for compilation functionality and Moss script for system plagiarism's functionality.

Περίληψη

Καθώς ο αριθμός των πανεπιστημιακών φοιτητών αυξάνεται, η διαδικασία της διαχείρισης των εργασιών γίνεται όλο και δυσκολότερη. Κάθε τμήμα ενός πανεπιστημίου όμως, έχει τις δικές του ανάγκες, σύμφωνα με το αντικείμενο, το οποίο διδάσκεται. Για παράδειγμα, ένα τμήμα πληροφορικής, ίσως χρειάζεται κάποιες διαδικτυακές εφαρμογές για προγραμματιστικά μαθήματα, στα οποία οι καθηγητές θα πρέπει να μπορούν να αναθέτουν προγραμματιστικές εργασίες στους φοιτητές τους. Εφόσον πρόκειται για προγραμματιστικά μαθήματα, σε διαφορετικές γλώσσες προγραμματισμού, όταν οι φοιτητές υποβάλλουν τις εργασίες τους, θα πρέπει να μπορούν να δουν τόσο τα αποτελέσματα του compilation όσο και τα αποτελέσματα των δοκιμαστικών περιπτώσεων που οι καθηγητές τους έχουν ορίσει για τη συγκεκριμένη εργασία. Καθώς όμως υπάρχουν πολλές περιπτώσεις αντιγραφής μεταξύ εργασιών, είναι απαραίτητο σε μία τέτοια εφαρμογή να υπάρχει και κάποιο είδος συστήματος ελέγχου αντιγραφής, το οποίο θα μπορεί να ανιχνεύει τυχόν ίδια κομμάτια κώδικα. Επιπλέον, μετά το πέρας της προθεσμίας, οι καθηγητές θα πρέπει να έχουν τη δυνατότητα να αναθέτουν τις υποβολές των εργασιών των φοιτητών στους αντίστοιχους βαθμολογητές, για βαθμολόγηση και σχολιασμό. Συνεπώς, όταν οι βαθμοί και τα σχόλια είναι έτοιμα, οι καθηγητές θα πρέπει να μπορούν να τους δημοσιεύουν, προκειμένου να είναι άμεσα προσπελάσιμοι από τους μαθητές τους.

Ο κύριος στόχος αυτής της εργασίας είναι να αναπτύξουμε μια πλατφόρμα, η οποία να υποστηρίζει προγραμματιστικές εργασίες (κυρίως στη γλώσσα προγραμματισμού C), και να καλύπτει την προαναφερθείσα λειτουργικότητα. Για την ακρίβεια, αναπτύσσουμε μια πλατφόρμα, στην οποία υπάρχουν 4 κατηγορίες χρηστών: *Διαχειριστής*, *Καθηγητής*, *Βοηθός* ή *Βαθμολογητής* και *Φοιτητής*. Κάθε χρήστης μπορεί να κάνει κάποια συγκεκριμένα πράγματα στην εφαρμογή. Πιο συγκεκριμένα, το κάθε είδος μπορεί να διαχειριστεί μαθήματα, εργαστήρια και εργασίες για το σπίτι με το δικό του τρόπο, εκτός από το διαχειριστή ο οποίος μπορεί επίσης να διαχειριστεί ακόμη και χρήστες.

Συνοψίζοντας, προκειμένου να αναπτύξουμε την εφαρμογή μας, χρησιμοποιήσαμε την αρχιτεκτονική MVC 5 και το Bootstrap 3. Επιπροσθέτως, για την υλοποίηση και τη διαχείριση της βάσης δεδομένων χρησιμοποιήσαμε την SQL, για τη λειτουργία του compilation χρησιμοποιήσαμε το λογισμικό Cygwin ενώ για το έλεγχο αντιγραφής εργασιών ένα script που ονομάζεται Moss.

Acknowledgments

I would like to acknowledge the help and support of Dr. Christos Antonopoulos who was the supervisor of this thesis, as well as of Dr. Panagiotis Bozanis and Dr. Emmanouil Vavalis. I would also like to thank Mrs. Vana Doufexi, who has helped me to a great extent. I also appreciate the help and the love of my family and my friends, who took a real part in my studies.

Dedicated to my family!!!

Table of Contents

Abstract 6

Περίληψη 8

Acknowledgments 10

Table of Contents 12

List of Tables 16

Table of Figures 18

1 Introduction 21

  1.1 Thesis Scope 22

  1.2 Technologies Used 23

  1.3 Thesis Organization 24

2 Background 26

  2.1 MVC Architecture 26

    2.1.1 Interactions 26

    2.1.2 Use in Web Applications 27

    2.1.3 MVC Benefits 27

  2.2 Bootstrap Framework 28

    2.2.1 Features 28

    2.2.2 Structure and Function 28

    2.2.3 Usage 29

  2.3 Plagiarism Detection - Moss 29

    2.3.1 Languages 30

    2.3.2 An Internet Service 30

    2.3.3 Moss Script Versions and Usage 30

  2.4 SQL 30

  2.5 Cygwin 31

  2.6 Development Tools 31

3 Requirements' Analysis 33

  3.1 Administrator 35

    3.1.1 User Administration 36

3.1.2 Course Administration 36
3.1.3 Lab Administration 36
3.1.4 Homework Administration 36
3.2 Professor 37
3.2.1 Course Administration 38
3.2.2 Lab Administration 38
3.2.3 Homework Administration 38
3.3 Assistant 38
3.3.1 Course Administration 39
3.3.2 Lab Administration 39
3.3.3 Homework Administration 40
3.4 Student 40
3.4.1 View Enrolled Courses 40
3.4.2 Lab Participation 40
3.4.3 Homework Participation 40
4 Design and Implementation 43
4.1 Application Database 43
4.1.1 Database Schema 45
4.2 Web Application Implementation 45
4.2.1 Internet Information Services Manager 45
4.2.2 Web Application & Database Connection 46
4.2.3 Application Models 48
4.2.4 Application Controllers 51
4.2.5 Application Views 54
4.2.6 Compilation & Testing Functionality - Security Considerations 55
4.2.7 Moss Functionality 55
5 Deployment and Results 58
5.1 User Interface Model 58
5.2 User Login 58
5.3 Administrator's Panel 59
5.3.1 Administrator's Home Page 59
5.3.2 Create User 60

5.3.3 Create Assistants, Professors and Students in Groups 62
5.4 Professor's Panel 63
5.4.1 Professor's Home Page 64
5.4.2 Create Course and Edit Course Information 64
5.4.3 Create Labs and Homeworks 67
5.4.4 Lab and Homework Administration 71
5.5 Assistant's Panel 75
5.5.1 Assistant's Home Page 75
5.5.2 Enrolled Courses 76
5.5.3 Lab and Homework Administration 76
5.6 Student's Panel 77
5.6.1 Student's Home Page 77
5.6.2 Course Enrollment 78
5.6.3 Assignment Submission 80
5.6.4 Submission Grade and Comments 84
6 Related Work 86
6.1 E-learning Platforms 86
6.2 Plagiarism Detection Systems 87
6.3 Online Compilation Tools 88
6.4 Automated Testing Systems 88
7 Conclusions 90
Bibliography 92
Appendices 95
Appendix A 95
Appendix B 98

List of Tables

Table 1. Database Connection String....47
Table 2. Database Connection Function....48
Table 3. Grade Model - Attributes....49
Table 4. AddGrade Model Function....50
Table 5. AddGrade ActionResult Method....52
Table 6. Process Run....56
Table 7. Add Grade View....96

Table of Figures

Figure 1. Technologies Used 23
Figure 2. MVC Components Interaction 27
Figure 3. Cygwin window 31
Figure 4. The Life of a Homework - UML Sequence Diagram 34
Figure 5. Administrator - UML Use Case Diagram 35
Figure 6. Professor - UML Use Case Diagram 37
Figure 7. Assistant - UML Use Case Diagram 39
Figure 8. Student - UML Use Case Diagram 41
Figure 9. Database Tables 44
Figure 10. Internet Information Services 45
Figure 11. Internet Information Services (IIS) Manager 46
Figure 12. Application Models 51
Figure 13. Application Controllers 53
Figure 14. Application Views 54
Figure 15. Log In - Form 58
Figure 16. Administrator's Home Page 59
Figure 17. User Registration 60
Figure 18. Role Choice 61
Figure 19. User's Personal Data 61
Figure 20. Professors' Data .csv File 62
Figure 21. Upload Professors using .csv File 62
Figure 22. Application Users 63
Figure 23. Professor's Home Page 64
Figure 24. Course Creation 65
Figure 25. Professor's Course List 65
Figure 26. Edit Professors 66
Figure 27. Edit Course Information 66
Figure 28. Get Course Back Up 67
Figure 29. Create Course Lab 68
Figure 30. Lab Section Creation 69
Figure 31. Lab Sections List 69
Figure 32. Create Public Test Case 70
Figure 33. Public Test Cases List 71
Figure 34. Labs List 71
Figure 35. Get Submissions on .zip File 72
Figure 36. Lab Submissions List 72
Figure 37. Grader Choice 73
Figure 38. Submission Grade and Comments 73

Figure 39. Lab Grades and Comments..74
Figure 40. Moss Results 75
Figure 41. Assistant's Home Page 76
Figure 42. Assistant's Course List...76
Figure 43. Assistant's Labs List...77
Figure 44. Assistant's Labs Submissions 77
Figure 45. Student's Home Page...78
Figure 46. Student's Course List..78
Figure 47. Course Enrollment 79
Figure 48. View My Enrolled Courses Page 79
Figure 49. List of Homeworks 80
Figure 50. Homework Task 81
Figure 51. Partner's Choice 81
Figure 52. Pending Status 82
Figure 53. Accepted Status 82
Figure 54. Compilation Results 83
Figure 55. Submissions List of Homeworks 84
Figure 56. Database Schema...99

1 Introduction

As the time passes, the number of university students is rapidly increasing, and the whole procedure of the submissions management is getting much more difficult. The course professors' have not only to evaluate every submission but also to provide feedback to the students. The already existed old-fashioned tools such as email, are not so efficient any more, as both professors and graders waste a lot of their time searching between the submissions. Even though there are platforms for the management of the submissions, they are not able to fulfill all the needs of each user. Such platforms, for example, are *Open Courses*, *Coursera* and *e-class* systems. As a result, there is the need for software assistance, which will facilitate the whole submissions management.

Each university department has its own needs, depending on the subject that is being taught. For example, a computer science department certainly needs web applications for programming courses, where professors will be able to assign programming tasks to their students. Students must have the ability to view professors' assignments, and upload their submissions. Due to the fact that it comes to programming courses on different programming languages, after submitting their tasks, students should be able to view compilation results and results from test cases. Furthermore, thanks to the fact that courses use a variety of agile methods, students must be able to view other enrolled students so as to become partners (pair programming), and to submit their assignments as a team. Moreover, since many students try to copy from each other, it is vital that such an application use a kind of plagiarism system, which can detect same code parts. As a result, after the deadline and the submission of the tasks, professors will have the ability to see copied assignments and assign each submission to a grader for grading and commenting. Consequently, when grades and comments are ready, professors will be able to publish these grades to students, who will have the opportunity to view them.

Unfortunately, an online interactive application (like the one we have just described) between students and professors does not exist and both professors and students face some difficulties because of this lack of functionality. On the one hand, students might send wrong submissions to course professors, and as a consequence, they will start wondering about submissions grades and comments. On the other hand, professors have to do many different things on different places. First of all, they have to get submissions one by one and save them all together, so as to check later if some of them are copied. Moreover, they have to assign submissions to graders for grading, and before this to check the partner of each student. These users are also obliged to create a sharing file, where they should create a submission list and assign a grader to each submission. After this grading procedure is over, professors should check and publish grades and comments somewhere on a web site for their students to check. Last but not least, graders should have access to the corresponding file, and search among the submissions the ones that are assigned to them for grading.

In conclusion, due to the fact that each user has to do different things on different places, users usually get confused and waste their time. As a result, not only professors but also graders have more difficulty in managing students' submissions, which means that they may make some mistakes on submissions results.

1.1 Thesis Scope

The main *scope* of this thesis, is to develop a web platform, which can support programming assignments (and mainly in C programming language) and reduce the difficulty of submissions management. This web application will consider all the functionality which was described in the previous paragraph, and will make user's life much easier. Specifically, we aim at contributing to the development of a web application, where professors, students and graders will be able to interact with each other.

More specifically, we develop a web platform, where an administrator can manage users, courses, course laboratories or labs and course homeworks. First, professors can create and manage online courses, labs and homeworks. They can also add private and public test cases for unit testing, which will run after each student's submission. In addition, a professor is able to set a deadline for each task. When this deadline comes to an end, he has also the ability to view, of course after running a script for system plagiarism detection, if there are any copied submissions. Afterwards, the administrators or professors can assign a grader to each submission, who must grade and comment the tasks that have been assigned to him. Then, students will receive immediate feedback and they will have the ability to view their course grades and comments, as well as their submissions.

So, to cut a long story short, using this web platform, not only are professors and graders able to do their work more easily and faster, but also students can upload their submissions more quickly and more safely.

1.2 Technologies Used

We have used MVC 5 architecture [1] [2], a generic code framework, which means that we felt free while writing and editing code. As a result, we had the opportunity to test and fully develop the desired functionality of our platform. Using MVC 5, we were also able to use some plug-ins, which have helped us to a great extent. We have also used Bootstrap 3 framework [3], a framework which is appropriate for designing easily and rapidly a web application. Additionally, Bootstrap 3 was already installed, as every MVC new project provides it. We have also developed a SQL database [4], so as to save and edit application's data. We have also used cygwin [5], a console which runs shell commands, as well as moss script [6], a script which checks system plagiarism.

We would like to mention, that we have spent a lot of time using moodle [7], which is a learning platform designed for the development of e-learning systems. However, we have come to the conclusion that it was almost impossible to customize our platform using it. As a result, and since our first try failed, we had to start the web application development from scratch using MVC .

On the next figure (Figure 1), we intend to display how all these technologies were used together, so as to develop successfully our web platform.

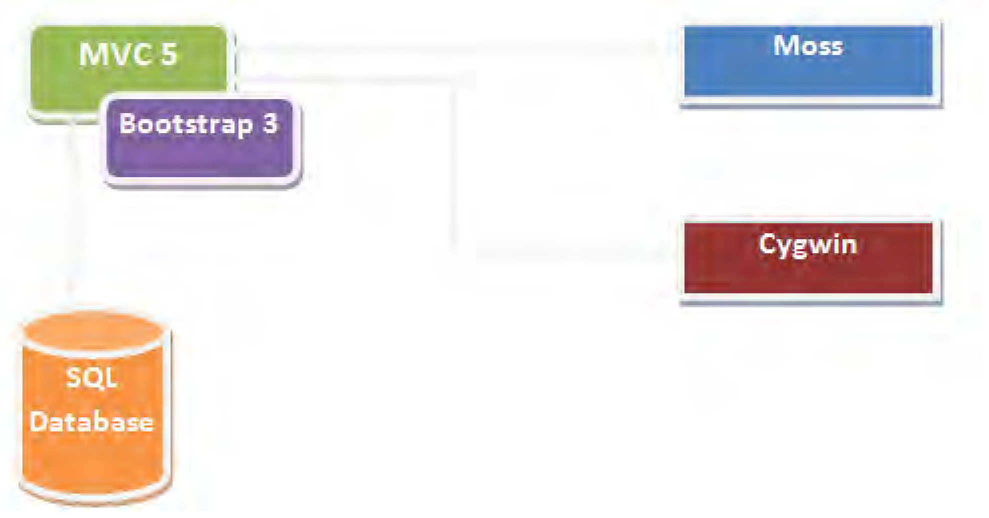


Figure 1. Technologies Used

1.3 Thesis Organization

This thesis is organized into seven chapters, including *Introduction* chapter, as follows:

- **Chapter 2** presents web application used technologies and software. It also contains details and figures about them.
- **Chapter 3** analyzes the web application requirements' analysis. We present individually each kind of user and its own functionality.
- **Chapter 4** describes the design and the implementation of the web platform. In this chapter, there are also some architecture diagrams of the platform as well as little code parts.
- **Chapter 5** focuses on the implementation results and the deployment of the work. It also provides a user guide for the platform.
- **Chapter 6** discusses the related work.
- **Chapter 7** concludes the results of this thesis. There are also some written thoughts of future work.

2 Background

This chapter discusses in brief, about the technologies and the architectures used as a background in our application.

2.1 MVC Architecture

MVC (Model-View-Controller) is an architecture for the development of user interfaces. This architecture divides the application that is being implemented into three different but interconnected parts. [2]

2.1.1 Interactions

The MVC architecture consists of 3 main components: the *Model*, the *View* and the *Controller*. Each component has its own functionality, and the combination of them is indispensable, so as for an application to be developed.

First, the main component of MVC, called *model*, mirrors the behavior of the application. Models, are in fact, classes which represent the data of the application and use validation logic in order to enforce appropriate rules for this information. In this part, the user should define the necessary class attributes, which should be associated with one another. What is more, using a *controller*, the user is able to have access to these attributes (which should be public in order to be used). We would like to mention that controllers are classes that handle incoming browser requests, get model data, and then set view templates that return a response to the browser. For instance, on a controller the user can fill information on the model's attributes and afterwards he can return this data on a *View.* Views are template files that the application uses to dynamically generate HTML pages. Using a view, users have the ability to present the model's data to user interface. On the next figure (Figure 2), we will display the interaction between the three main components of MVC.

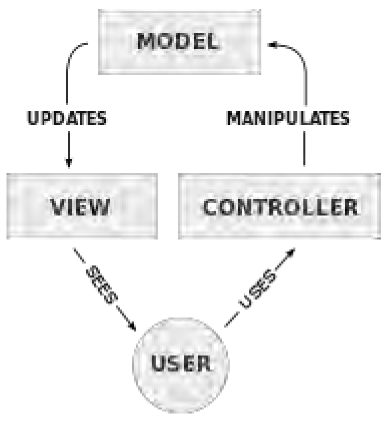


Figure 2. MVC Components Interaction

2.1.2 Use in Web Applications

Many years ago, MVC was developed in order to satisfy desktop application needs. Over the years, this technology has been widely adopted as an architecture for World Wide Web applications in major programming languages. A great number of frameworks have been developed that enforce this pattern. These frameworks may be various but the MVC responsibilities are mainly divided between client and server. In particular, the client sends either requests or form data to the controller and then gets a full complete web page (or other type of file) from the view.

2.1.3 MVC Benefits

The use of MVC architecture has a lot of benefits. First, the separation of the three components, allows the re-use of the web applications. As a result, multiple user interfaces can be implemented without having thoughts of the codebase. What is more, the web developers of Models can focus exclusively on the application's logic implementations, modifications, updates without being concerned about the layout of the application. Furthermore, class developers can build the classes, while the UI developers can involve in designing the layout of the application simultaneously, resulting in interdependence issues and of course time conservation. Last but not least, a lot of updates on the models and on the controllers can be made without being visible to users. [8]

2.2 Bootstrap Framework

"Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web." [3]

2.2.1 Features

Bootstrap can be used along with the latest versions of all major browsers. It has 3 versions. Since version 2.0 the Bootstrap usage also provides responsive web design. This means that the layout of web pages adjusts according to the characteristics of the device used. (desktop, tablet, mobile phone). Moreover, with version 3.0, Bootstrap adopted a mobile first design philosophy, which gives emphasis to responsive design by default. As a result, we can easily use it in order to develop web applications compatible with both desktops and mobiles functionality.

2.2.2 Structure and Function

The main structure consists of 3 parts: The CSS stylesheet, Re-Usable components and Javascript components.

The CSS stylesheet

Bootstrap has many html classes which are available for the most Html tags. These classes may give a modern and stylish appearance not only to the text, but also to the tables and the form fields. In many cases, some rules of these classes are overwritten by the user, according to the needs of his web application.

Re-usable components

Bootstrap also contains some re-usable components. Such components, for example, are modern dropdown lists or modern buttons. These elements can be used in many parts of the project with exactly the same structure and the same calling. The application developer is also able to format these elements.

JavaScript components

Bootstrap also uses plenty of Javascript components, so as to extend the functionality of some html tags such as, for example, an auto-complete function for input fields, modal windows, dropdown lists, nav tabs, popover and tooltip functionality, alert windows, buttons, carousels and collapsing menus.

2.2.3 Usage

The usage of Bootstrap in an HTML page is very simple. The user only needs to download the bootstrap files and add them to his project. Afterwards, he should link bootstrap css and js files, so that the user will be able to use them. Moreover, as we have already noticed, bootstrap rules can be overwritten by the user, according to his application needs. Each user has the ability to change, for example, the layout and in some way, the functionality of bootstrap components. In this thesis, Bootstrap 3 is used for the development of our web application. Many rules, have also been changed according to the application's functionality and layout.

2.3 Plagiarism Detection - Moss

Moss is a technology, which is able to find every possible code similarity between programs written in a wide range of programming languages. Moss uses an algorithm, which is characterized by a really significant improvement, over other similar functionality's algorithms.

Even though, moss uses a really significant algorithm [9], it is not always capable of detecting plagiarism. Moss is not able to detect, for example, why some code parts are similar. Consequently, moss script provides the user with the ability to view the parts of the code that Moss considers as similar and then to choose about whether it comes to plagiarism or not.

In conclusion, it is a misuse of Moss to be based only on the similarity scores. These scores are surely very helpful in order to determine, to some extent, whether some code pairs are same or not, but that is not enough to prove that it comes to plagiarism. We present some moss results in subsection 5.4.4.4 Moss Plagiarism on Figure 40. [6]

2.3.1 Languages

Moss provides scripts that allowed to be used in a lot of programming languages. It can detect code plagiarism in the following languages:

C, C++, Java, C#, Python, Visual Basic, JavaScript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS assembly, a8086 assembly, a8086 assembly, MIPS assembly, HCL2.

2.3.2 An Internet Service

Moss script is able to be used as a part of a web application, as it is being provided as a web service. Moss can be run as a simple web request, which produces HTML pages listing pairs of programs with similar code. Moss provides the user with the opportunity to compare similar or same files, as it notices same texts in programs. Moreover, Moss is able to pass by matches to code, that the user has defined as shared. [6]

2.3.3 Moss Script Versions and Usage

Moss script is provided thanks to *Community Contributions* and it is available in many languages, which all have the same functionality. First of all, a user can run moss script both on Linux System using *Java* programming language and *PHP* language and on Windows System using *C#* Programming Language.

The only precondition before a user utilizes the Moss script, is to obtain a moss account. Then he can use the moss script using the provided moss key.

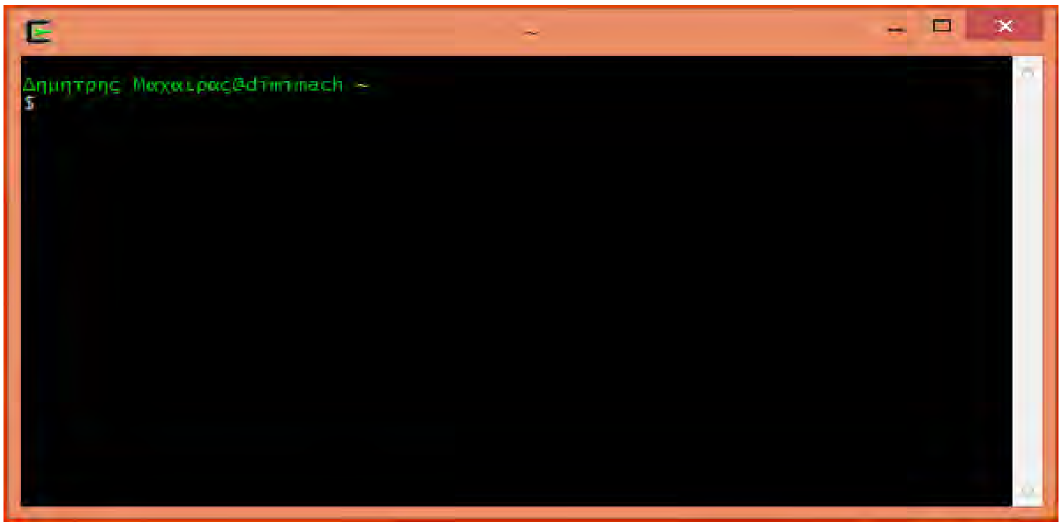
2.4 SQL

SQL is an indispensable part when it comes to data base applications. Using its queries, a user can easily create, first of all, the database, its tables and its table relations. Afterwards, the user can not only insert but also view, update and delete tables' data. A great number of programming languages are characterized by a variety of functions, which can execute such queries and get data from the database or even set data to it. [4]

2.5 Cygwin

"Cygwin is UNIX environment working on Microsoft Windows OS, for simulation environment." [5].

Needless to say, this environment can be easily installed on each and every version of Microsoft Windows as it is a free and open source software. After its installation, the user is able to execute Linux Commands, in the same way that he executes them on a Linux Console. In this thesis, we have mainly used it in order to execute Linux Commands, such as a gcc command for c file compilation. On the figure (Figure 3), we display a simple cygwin screen from our computer.



```
Δημητρης Μιχαλιρας@dimmach ~
$
```

Figure 3. Cygwin window

2.6 Development Tools

In this thesis, Visual Studio 2013 is being used so that a web application can be implemented. In this case, as we have described in previous chapter, MVC 5 is being used by Visual Studio, due to the fact that it is the latest version of MVC architectures. We also use SQL Management Studio 2012, in order to create and manage databases and their contents. This software can be easily connected with other software, (for example with Microsoft Visual Studio), so that data can be easily transmitted from one another.

3 Requirements' Analysis

In this chapter, we describe the functionality of our web platform. We also display some *UML* Diagrams, which we have created using Visual Paradigm software.

The main goal of this application is to simplify the whole functionality of programming courses administration. First of all, using this platform, administrators will be able to administer users in different ways. They can simply create a user, or upload users in groups. Then, they have the ability to view users and edit their data. Additionally, when they create a new user, his password is saved on the database using hash code for security reasons. Moreover, this platform make it easier for professors administer their courses and their content. Using it, professors will be able to create and administer courses and as a result, their homeworks and labs. Particularly, professors can create labs and homeworks, where they can also add new public or private cases, as well as view students' submissions, compilation results and results of the test cases execution, after the specified deadline. They will also be able to check each lab and homework in order to be certain whether there is any plagiarism or not, and then to get and save all submissions. Afterwards, professors can assign each submission to a grader for grading and commenting. As for graders, there are also advantages, as they do not have to search the submissions that have been assigned to them. After entering the platform, they will be able to view for each homework or lab, only their assigned submissions. They can also see the grading instructions and afterwards grade their submissions. What is more, students will have the ability to enroll to courses and view their available homeworks and labs. They can also choose a partner, and submit their tasks online and within the deadline, set by professors, without sending their tasks in emails and late. Afterwards, students will be able to view compilation and test cases results, as well as submission files. Last but not least, students can check the submission file that they have sent, and if it is not correct, they can submit it again and view the new results. In this way, they will be totally sure about their submission without doubting whether they have submitted the right file or not.

On the next figure (Figure 4), we intend to present a UML Sequence diagram which depicts the life of a Homework. First, we can see 3 actors (Professor, Grader, Student) and the Homework object. As we can also observe, each actor sends messages to the object. In some cases the Homework returns a message to the actor or sends a message to itself.

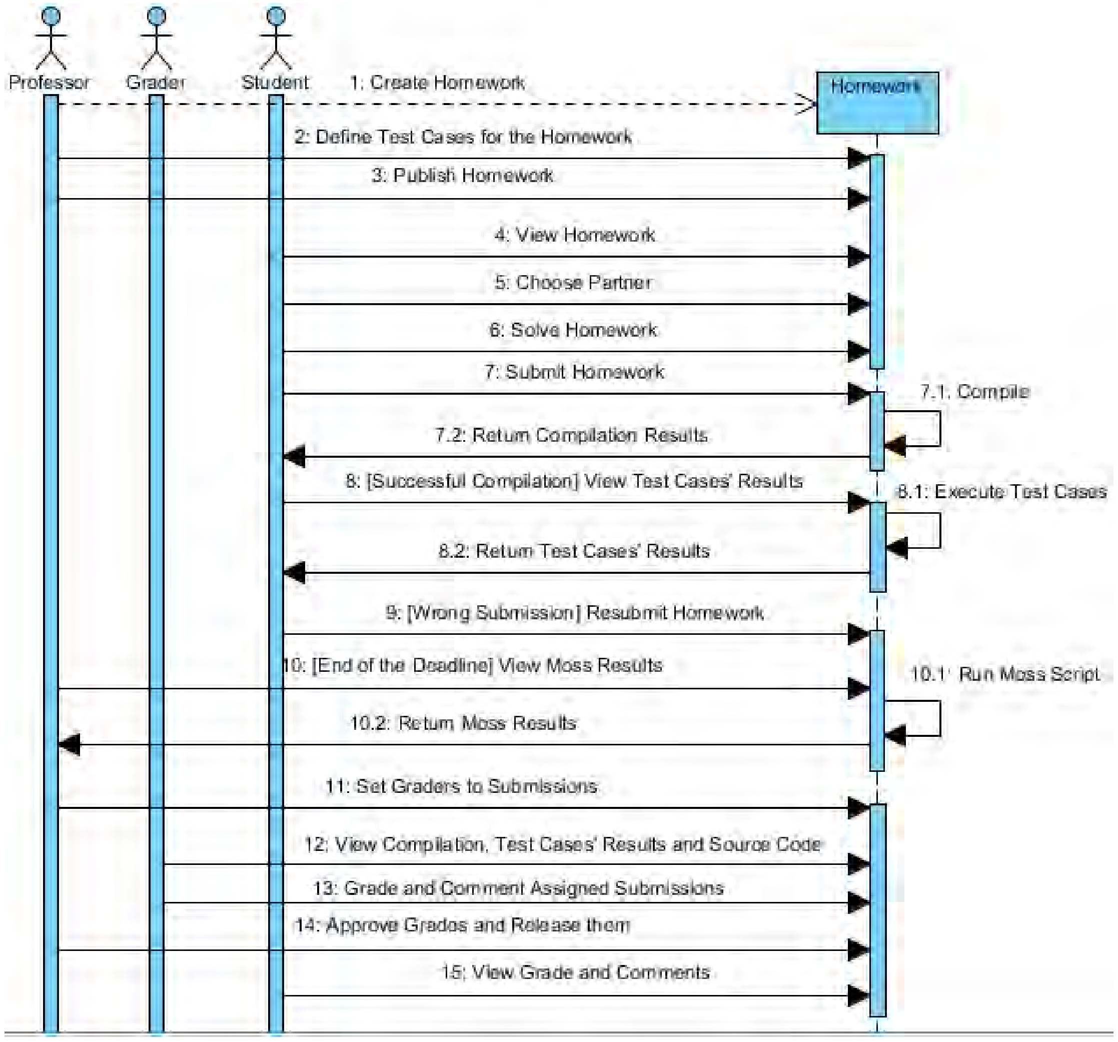

Professor
Grader
Student
Homework
1: Create Homework
2: Define Test Cases for the Homework
3: Publish Homework
4: View Homework
5: Choose Partner
6: Solve Homework
7: Submit Homework
7.1: Compile
7.2: Return Compilation Results
8: [Successful Compilation] View Test Cases' Results
8.1: Execute Test Cases
8.2: Return Test Cases' Results
9: [Wrong Submission] Resubmit Homework
10: [End of the Deadline] View Moss Results
10.1: Run Moss Script
10.2: Return Moss Results
11: Set Graders to Submissions
12: View Compilation, Test Cases' Results and Source Code
13: Grade and Comment Assigned Submissions
14: Approve Grades and Release them
15: View Grade and Comments

Figure 4. The Life of a Homework - UML Sequence Diagram

We would like to mention that this platform has 4 different kinds of users: *ADMINISTRATOR*, *ASSISTANT* or *GRADER*, *PROFESSOR* and *STUDENT*. Every user has its own functionality and as a result, can perform different actions. What is more, each user should have his personal username and password, so as to have access to the platform. On the next sections, we intent to present separately these kinds of users. All users should first log in on the web application so as to have access to its functionality.

3.1 Administrator

First, an administrator has the most powerful role of the application, as he is able to perform whichever functionality is available on it. In particular, an *Administrator* can administer *Users*, *Courses*, *Labs* and *Homeworks*.

On the next figure (Figure 5), we can see a UML Use Case diagram of administrator. First of all, we can see the available use cases for him. As we can observe, there are 5 main use cases, and for each and every one of them, there are other use cases, which extend the main use cases. Particularly, we notice the 5 main use cases, which are *Log In, User Administration*, *Course Administration*, *Lab Administration* and *Homework Administration.* For the second one, for example, we have some extended use cases, whose names are *Create Simple User*, *Create & Edit Users* as well as, *Enable & Disable Users*. These actions (use cases) are optional, which means that an administrator is not obliged to perform them.

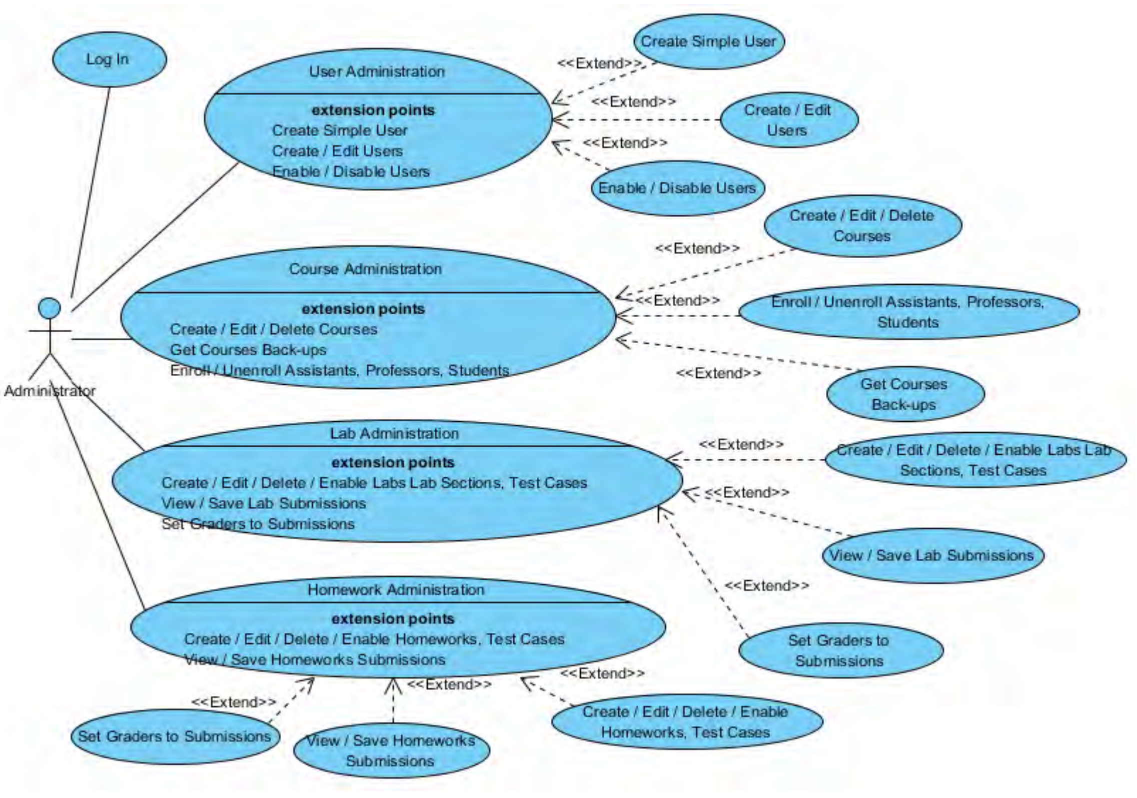

Log In
User Administration
extension points
Create Simple User
Create / Edit Users
Enable / Disable Users
Create Simple User
<<Extend>>
<<Extend>>
Create / Edit Users
<<Extend>>
Enable / Disable Users
Course Administration
extension points
Create / Edit / Delete Courses
Get Courses Back-ups
Enroll / Unenroll Assistants, Professors, Students
Create / Edit / Delete Courses
<<Extend>>
<<Extend>>
Enroll / Unenroll Assistants, Professors, Students
<<Extend>>
Get Courses Back-ups
Administrator
Lab Administration
extension points
Create / Edit / Delete / Enable Labs Lab Sections, Test Cases
View / Save Lab Submissions
Set Graders to Submissions
<<Extend>>
Create / Edit / Delete / Enable Labs Lab Sections, Test Cases
<<Extend>>
View / Save Lab Submissions
<<Extend>>
Set Graders to Submissions
Homework Administration
extension points
Create / Edit / Delete / Enable Homeworks, Test Cases
View / Save Homeworks Submissions
<<Extend>>
<<Extend>>
<<Extend>>
Set Graders to Submissions
View / Save Homeworks Submissions
Create / Edit / Delete / Enable Homeworks, Test Cases

Figure 5. Administrator - UML Use Case Diagram

3.1.1 User Administration

This kind of user, is able to administer all application users. First of all, not only can he add a simple user but also *Assistants*, *Professors* and *Students* in groups from .csv files. Moreover, an administrator has the ability to view and edit users, as well as to enroll users or unenroll them from the platform. Additionally, due to the fact that a student can also become an assistant, an administrator is also able to add the student as an assistant.

3.1.2 Course Administration

A user, who has this role, can create, view, edit and delete courses and their contents. Furthermore, an administrator can enroll assistants to courses. Afterwards, he can also view and edit enrolled assistants, professors and students. Last but not least, such a user can also get and save a course backup.

3.1.3 Lab Administration

An administrator can not only create, view, edit, enable, disable and delete a *Lab*, but also, a *Lab Section*. For each Lab Section he can create, view and delete *Test Cases (Public & Private)*, which will run after every compilation of a student's submission. Additionally, an administrator, is able to view and save submissions, appoint graders for the student submissions, grade and comment submissions and save or publish in groups grades and comments on .csv or .pdf files. We have made the decision that submissions should be saved as .zip files, which contain all submissions and compilation files. Last but not least, an administrator can run moss script and view its results. Consequently, he will have the ability to view, whether there are any copied submissions or not. If there are, the administrator is even able to check the similarities between these submissions.

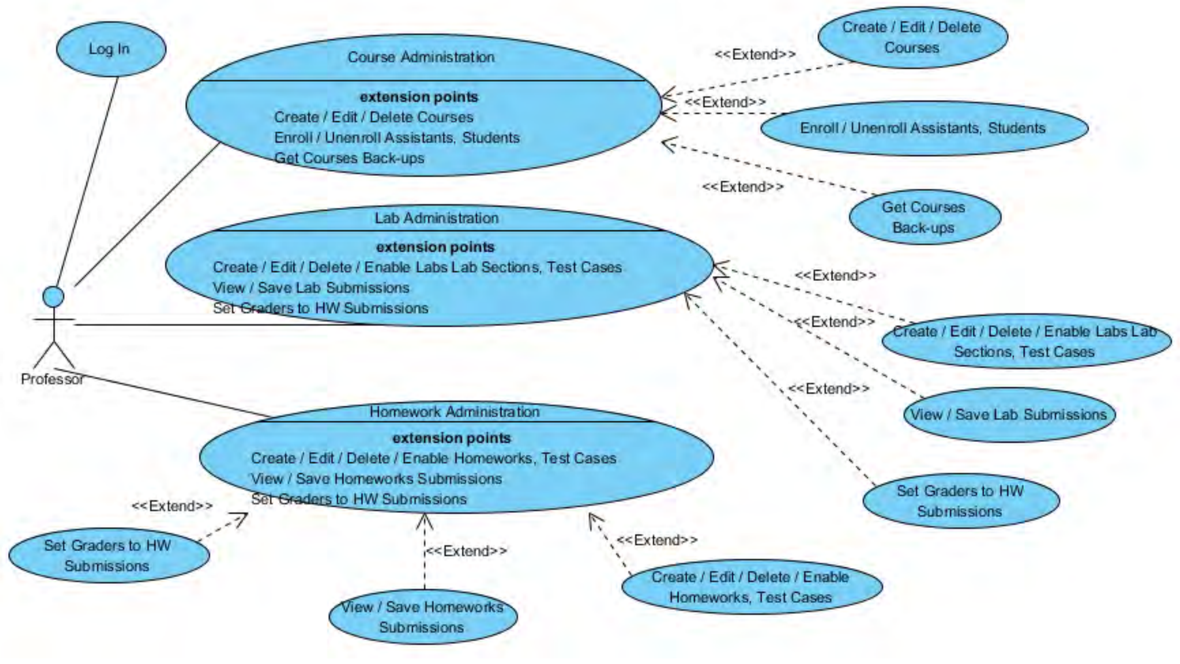
3.1.4 Homework Administration

Due to the fact that a *Homework* has the same functionality as a *Lab Section*, an administrator can make things here that are exactly similar to the works described in Lab Administration.

3.2 Professor

A *Professor* has the second most significant role after the *Administrator*, as both of them have the same functionality, except for *User Administration*. As a result, a professor can administer *Courses*, *Labs* and *Homeworks*.

On the next figure (Figure 6), we can see a UML Use Case diagram of a professor. First of all, we observe the available use cases for a professor. As we can see, there are 4 main use cases, which are the same as the 4 of the administrator's use case diagram. There are also other use cases, which extend the main use cases. Particularly, we can see the 4 main use cases, which are *Log In, Course Administration*, *Lab Administration* and *Homework Administration.* For the fourth one, for example, we have some extended use cases, whose names are *Set Graders to HW Submissions*, *View & Save Homework Submissions* as well as, *Create / Edit / Delete / Enable Homeworks & Test Cases*. These actions (use cases) are optional, which means that a professor is not obliged to perform them. What is also to notice on the next figure, is that the *Lab Administration* use case is almost the same as the *Homework Administration* use case. This fact does not surprise us, as both of them perform the same actions.



Log In
Course Administration
extension points
Create / Edit / Delete Courses
Enroll / Unenroll Assistants, Students
Get Courses Back-ups
<<Extend>>
Create / Edit / Delete Courses
<<Extend>>
Enroll / Unenroll Assistants, Students
<<Extend>>
Get Courses Back-ups
Lab Administration
extension points
Create / Edit / Delete / Enable Labs Lab Sections, Test Cases
View / Save Lab Submissions
Set Graders to HW Submissions
<<Extend>>
Create / Edit / Delete / Enable Labs Lab Sections, Test Cases
<<Extend>>
View / Save Lab Submissions
<<Extend>>
Set Graders to HW Submissions
Professor
Homework Administration
extension points
Create / Edit / Delete / Enable Homeworks, Test Cases
View / Save Homeworks Submissions
Set Graders to HW Submissions
<<Extend>>
Set Graders to HW Submissions
<<Extend>>
View / Save Homeworks Submissions
<<Extend>>
Create / Edit / Delete / Enable Homeworks, Test Cases

Figure 6. Professor - UML Use Case Diagram

3.2.1 Course Administration

A professor, can create, view, edit and delete courses and their contents. Furthermore, he can enroll assistants to courses, as well as view and edit enrolled assistants, professors and students. Last but not least, such a user can also get and save a course backup.

3.2.2 Lab Administration

A professor can not only create, view, edit, enable, disable and delete a *Lab*, but also, a *Lab Section*. For each *Lab Section* he can create, view and delete *Test Cases (Public & Private)*, which will run after each student submission compilation. Additionally, a user who has the role of a professor is able to view and save submissions, set graders to submissions, grade and comment submissions and save or publish, in groups, grades and comments on .csv or .pdf files. We save submissions on .zip files, which contain all submissions and compilation files. Last but not least, a professor can run and view moss script and its results. Consequently, he has the ability to check, whether there are any copied submissions or not. If there are, the professor is able to view the similarities between the submissions.

3.2.3 Homework Administration

Due to the fact that a *Homework* has the same functionality as a *Lab Section*, a professor can follow the same steps and perform the same works as described in the chapter of Lab Administration.

3.3 Assistant

An *Assistant* has the less powerful role of the application, as he can only do minor works on it. In particular, an *Assistant* can administer *Courses*, *Labs* and *Homeworks*.

On the next figure (Figure 7), we can see a UML Use Case diagram of assistant. First of all, we can see the available use cases for an assistant. As we observe, here there are 4 main use cases, and for each one of them, there are one or two other use cases, which extend the main use cases. Particularly, we can see the 4 main use cases, which are *Log In*, *Course Administration*, *Lab Administration* and *Homework Administration.* For the third one, for example, we have 2 extended use cases, whose names are *View Labs* and *View/Grade/Save Lab Submissions*. It is quite important to mention once again that these actions (use cases) are optional, which means that an assistant is not obliged to perform them.

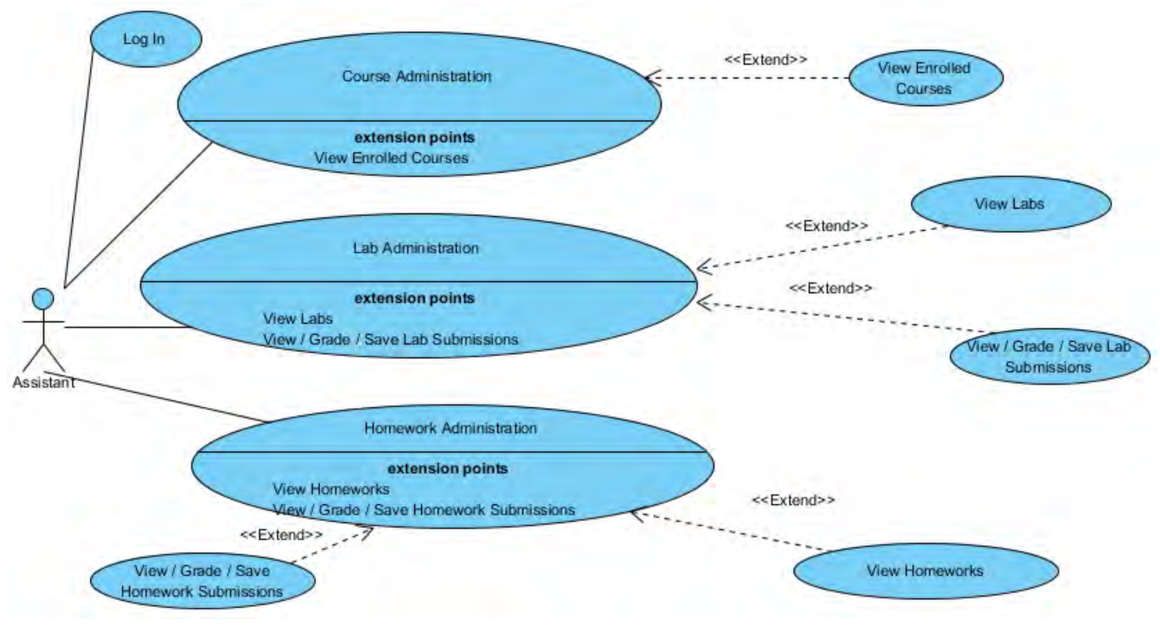


Figure 7. Assistant - UML Use Case Diagram

3.3.1 Course Administration

This kind of user can only view the courses he is enrolled in by professors. An assistant cannot edit any information of the courses, in which he is enrolled, as there is no need of doing it.

3.3.2 Lab Administration

First of all, such a user, can firstly view *Labs*, *Lab Submissions* as well as save, grade and comment submissions, which are assigned by course professors.

3.3.3 Homework Administration

As we have also mentioned in the part dedicated to the administrator's role, due to the fact that a *Homework* has the same functionality as a *Lab,* an assistant can perform the same actions as in Lab Administration.

3.4 Student

A *Student* has a role which is as significant as the role of a *Professor*. A *Student* can view enrolled *Courses* and participate in *Labs* and *Homeworks*.

3.4.1 View Enrolled Courses

A *Student* can enroll to courses only if he knows the course password. He can also unenroll from a course as well as view the courses that he has selected.

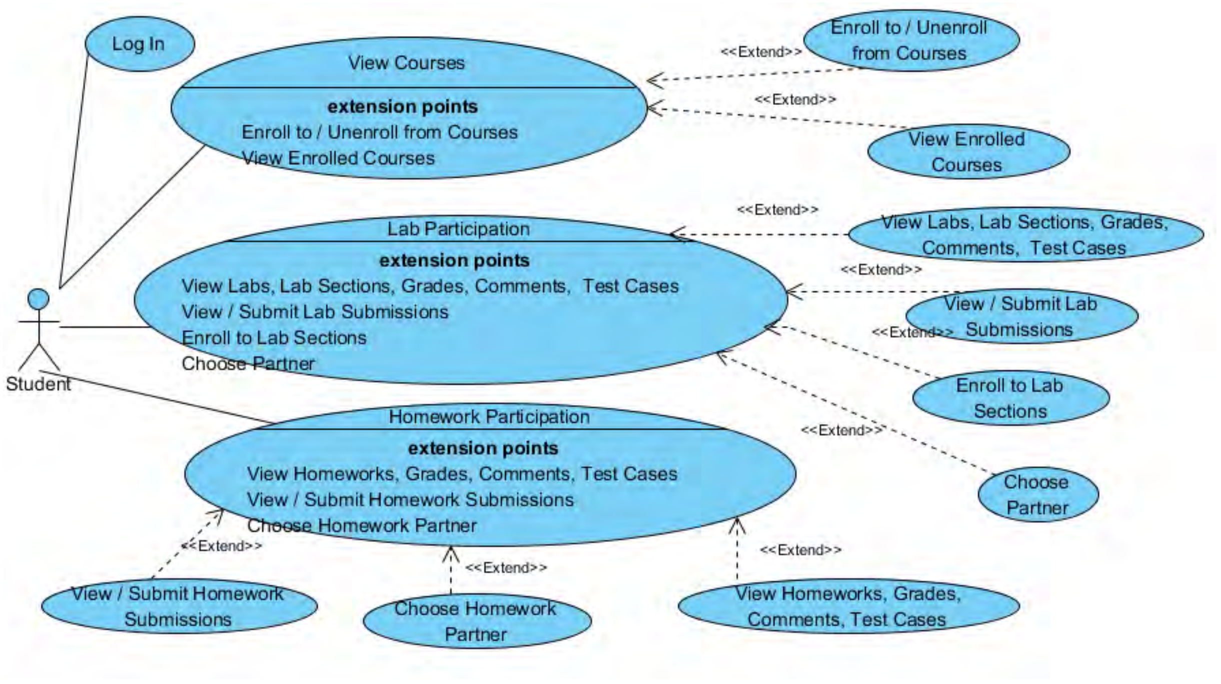
3.4.2 Lab Participation

The most important fact about this kind of user is that he can view *Labs* and enabled *Lab Sections*. Afterwards, he has to enroll to a section and then to choose a *Partner* among the enrolled to the lab section students. Moreover, he can upload his submission, which will be compiled after uploading, and then view the compilation and the test cases results. Furthermore, when the course professors publish lab grades, students are also able to view the submission grade and comments.

3.4.3 Homework Participation

Due to the fact that a *Homework* has the same functionality as a *Lab Section,* a student can perform the exactly same actions as in Lab Participation, apart from enrolling to a homework, as it is not necessary.

On the next figure (Figure 8), we can see a UML Use Case diagram of a student. First of all, we can easily observe the available use cases for a student. As we notice, there are 4 main use cases. There are also other use cases, which extend the main use cases. Particularly, we can see the 4 main use cases, which are *Log In, View Courses*, *Lab Participation* and *Homework Participation.* For the second one, for example, we have some extended use cases, whose names are *Enroll to Courses & Unenroll from Courses* as well as, *View Enrolled Courses*. These actions (use cases) are optional, which means that a student is not obliged to perform them.



Log In
View Courses
extension points
Enroll to / Unenroll from Courses
View Enrolled Courses
<<Extend>>
Enroll to / Unenroll from Courses
<<Extend>>
View Enrolled Courses
Lab Participation
extension points
View Labs, Lab Sections, Grades, Comments, Test Cases
View / Submit Lab Submissions
Enroll to Lab Sections
Choose Partner
<<Extend>>
View Labs, Lab Sections, Grades, Comments, Test Cases
<<Extend>>
View / Submit Lab Submissions
<<Extend>>
Enroll to Lab Sections
<<Extend>>
Choose Partner
Student
Homework Participation
extension points
View Homeworks, Grades, Comments, Test Cases
View / Submit Homework Submissions
Choose Homework Partner
<<Extend>>
View / Submit Homework Submissions
<<Extend>>
Choose Homework Partner
<<Extend>>
View Homeworks, Grades, Comments, Test Cases

Figure 8. Student - UML Use Case Diagram

4 Design and Implementation

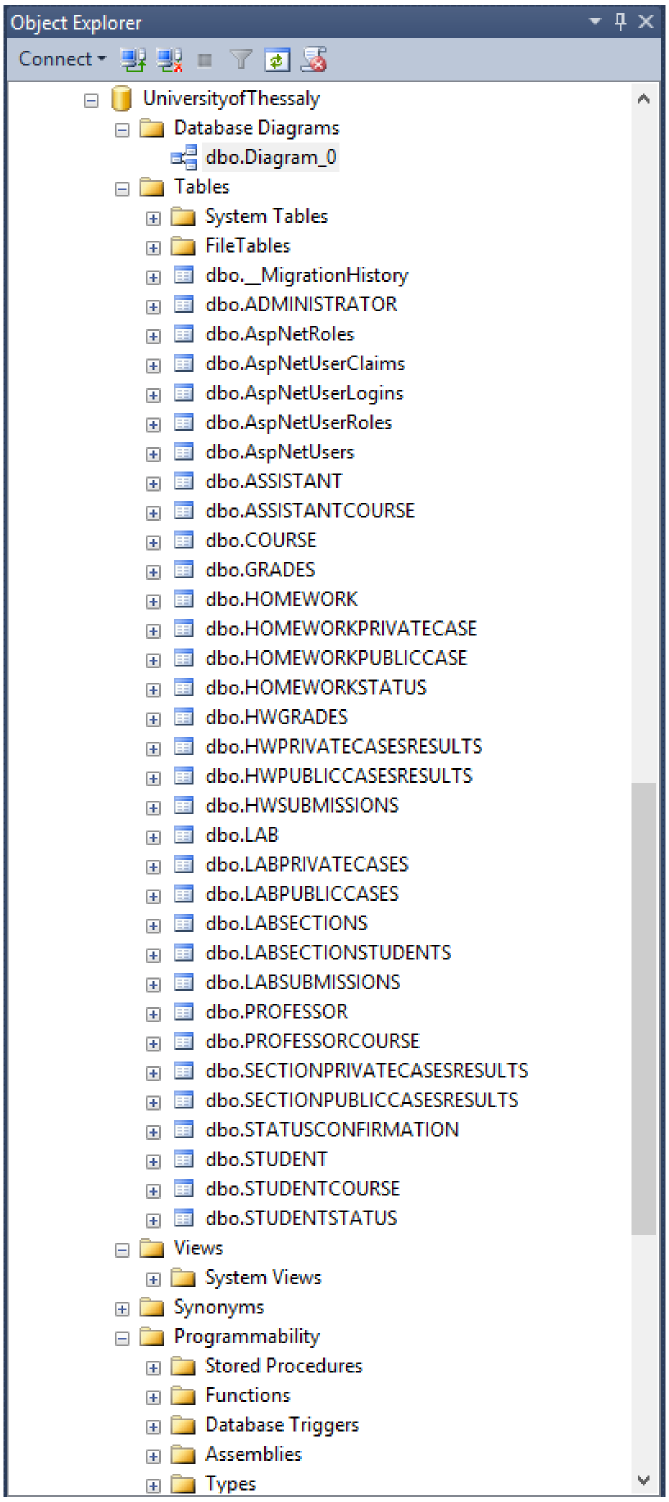
In this chapter, we intend to present the design and the implementation of our platform. First of all, we will describe the database of the application. Afterwards, we will discuss the whole structure and implementation of our platform with the help of figures and code parts.

4.1 Application Database

As it has already been mentioned in previous chapters, we had to develop a database for our application. The database should be correctly designed and developed, since our aim is a functional and rapid database without bugs. As we described in the 2.6 Development Tools paragraph, we use *Microsoft SQL Server Management Studio 2012* for the development of the database, which is really easy to install. This software is accessible on the website of Microsoft [10] [11].

First of all, the database of our platform contains 33 tables, which are related to each other. We had to alter these tables a lot of times, as new facts occurred, which were not previously calculated, while designing the main database structure. This database was developed from scratch, using SQL queries, in order to create the tables, the primary and foreign keys and the relations between tables. Additionally, there are relations between the tables for safety and speed reasons. Not only a *SELECT * ...* query is much more quick to run, but also a *DELETE ... or UPDATE ...* query is much more safe when table relations exist.

Last but not least, due to the fact that a server is required to have access to the database, *localhost* is being used, as it will also be necessary for the implementation of the web application. Furthermore, we have created for web application needs a user, whose name is *dimimach*. On the next figure (Figure 9), we can view these database tables.



UniversityofThessaly
Database Diagrams
dbo.Diagram_0
Tables
System Tables
FileTables
dbo.__MigrationHistory
dbo.ADMINISTRATOR
dbo.AspNetRoles
dbo.AspNetUserClaims
dbo.AspNetUserLogins
dbo.AspNetUserRoles
dbo.AspNetUsers
dbo.ASSISTANT
dbo.ASSISTANTCOURSE
dbo.COURSE
dbo.GRADES
dbo.HOMEWORK
dbo.HOMEWORKPRIVATECASE
dbo.HOMEWORKPUBLICCASE
dbo.HOMEWORKSTATUS
dbo.HWGRADES
dbo.HWPRIVATECASESRESULTS
dbo.HWPUBLICCASESRESULTS
dbo.HWSUBMISSIONS
dbo.LAB
dbo.LABPRIVATECASES
dbo.LABPUBLICCASES
dbo.LABSECTIONS
dbo.LABSECTIONSTUDENTS
dbo.LABSUBMISSIONS
dbo.PROFESSOR
dbo.PROFESSORCOURSE
dbo.SECTIONPRIVATECASESRESULTS
dbo.SECTIONPUBLICCASESRESULTS
dbo.STATUSCONFIRMATION
dbo.STUDENT
dbo.STUDENTCOURSE
dbo.STUDENTSTATUS
Views
System Views
Synonyms
Programmability
Stored Procedures
Functions
Database Triggers
Assemblies
Types

Figure 9. Database Tables

4.1.1 Database Schema

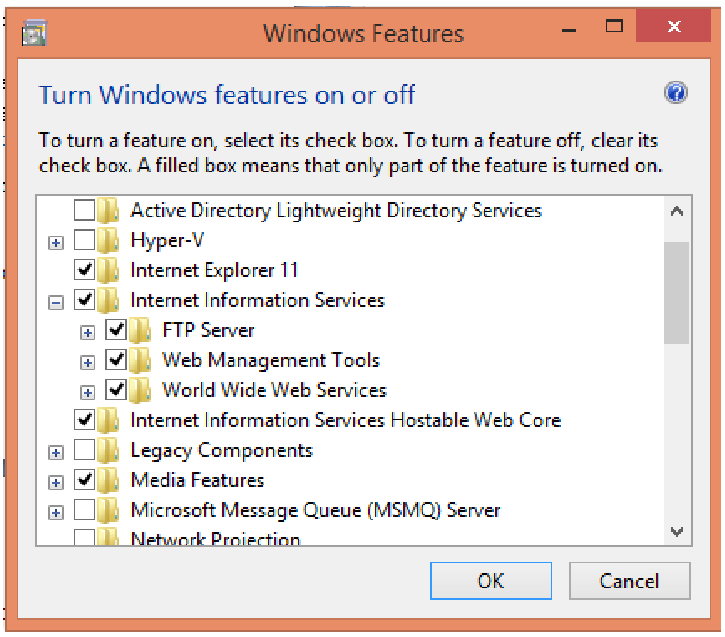
A database schema is in fact, a diagram, which depicts application tables, tables attributes and tables relations. A diagram of our database schema is displayed on Figure 56, in Appendix B.

4.2 Web Application Implementation

As we have said in previous chapters, we have used *Microsoft Visual Studio 2013 Professional* in order to implement our web application. This software is available for downloading on the Microsoft Developer Network website. [12]

4.2.1 Internet Information Services Manager

IIS (Internet Information Services) Manager should be available and enabled on the operation system or to the server, where the application runs. The next figure (Figure 10) presents the necessary enabled features, on which *IIS Manager* runs.



To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- [ ] Active Directory Lightweight Directory Services
- [ ] Hyper-V
- [✓] Internet Explorer 11
- [✓] Internet Information Services
  - [✓] FTP Server
  - [✓] Web Management Tools
  - [✓] World Wide Web Services
- [✓] Internet Information Services Hostable Web Core
- [ ] Legacy Components
- [✓] Media Features
- [ ] Microsoft Message Queue (MSMQ) Server
- [ ] Network Projection

Figure 10. Internet Information Services

In order to enable *IIS Manager*, we have to search it in the Control Panel and then to check all contents from the Internet Information Services folder. After typing and running *IIS* on windows search, the result of next figure (Figure 11) is displayed.

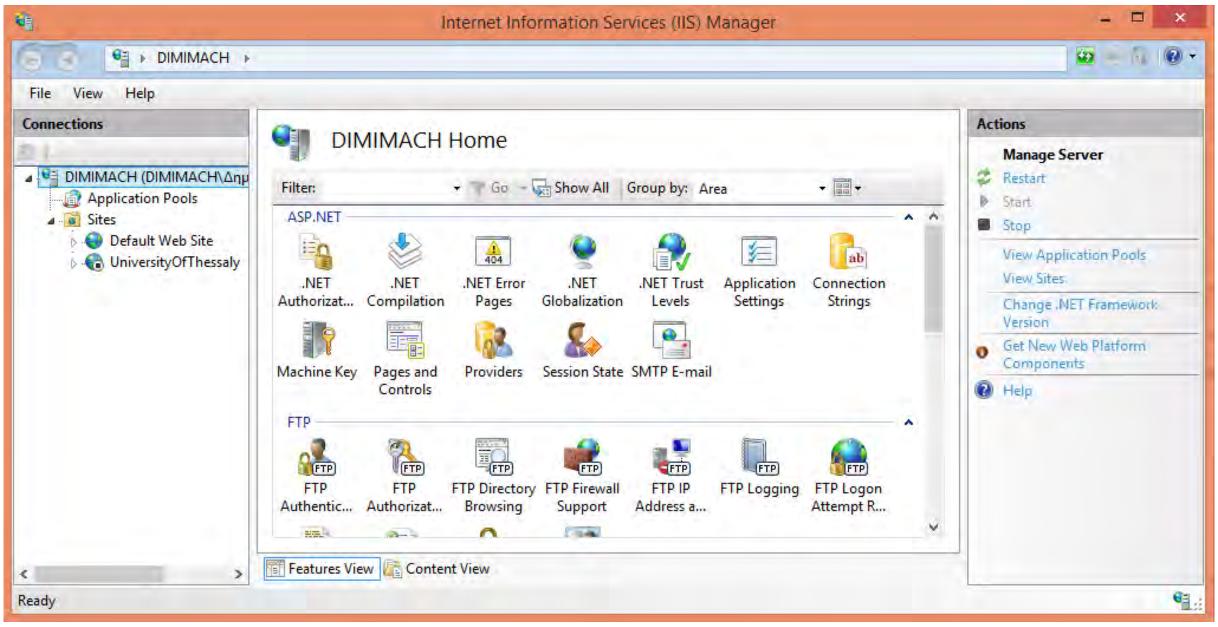


Figure 11. Internet Information Services (IIS) Manager

Using IIS Manager, a user is mainly able to view its functionality, to restart it and to view its websites. For the time being, there is only one website, which is the website of our platform. It is also possible to go through the existed and connected web applications to web browsers, such as Firefox, Google Chrome and Internet Explorer.

4.2.2 Web Application & Database Connection

After installing Microsoft Visual Studio 2013, the first we have to do in order to develop the application is to create a new MVC 5 web project. As it comes to a web project we have to define its url. Using the software (whose function we analyzed in previous subsection), it is quite simple to define the url and the port on which the application runs. In our application, the main url is (locally): *http://localhost/UniversityOfThessaly-ProgrammingCourses*. The next we have to do, is to enable the essential plug-ins according to the application needs. [13] We would also like to mention, that we use many plug-ins and one of the most basic is the Microsoft ASP.NET Identity Framework. In order to use it, we have to execute *enable-migration* command on *Nuget Console*. Then, after the first user registration, AspNet membership system tables and models are automatically created.

Afterwards, we have to initialize the connection between the database and the application. [14] In order to achieve that, we need to find and open web.config file in our project, so as to create a successful connection. This file is an xml file which contains xml tags. Among these tags, there is the connectionStrings tag, which contains the available database connection strings of our application. Consequently, we have to add our database connection string inside connectionStrings tag. We would also like to notice that the connection string attribute specifies the following attributes:

- Data Source: Information for the using SQL Server (in this case is *localhost*, as it was mentioned in 4.1 Application Database).
- Initial Catalog: Name of Application Database.
- User id and password: Information of a user, who has full access on the corresponding database.
- ProviderName: Information for the using Server Type.

As we can see on the next table (Table 1), there are two add-tags inside tag connectionStrings. The former of the two, is the default one, and the latter of the two is the new one that we have added. Moreover, *DefaultConnection* should always exist, as it is used for Microsoft Identity plug in, not only when its connection string is same to another already existed but also when it is not. It is also important to say that, *connectionStrings* tag can have more than one different connections to different databases and on different servers. Consequently, the web application is able, first of all, to connect and then execute different SQL queries on data located in different databases. In this application, one connection string is enough (except for the *DefaultConnection* connectionstring) , as we use only one database for the web platform.

```
<connectionStrings>
    <add name="DefaultConnection" connectionString="Data Source=localhost;Initial
Catalog=UniversityOfThessaly; user id=dimimach; password = pass"
providerName="System.Data.SqlClient" />
    <add name="UniversityOfThessaly" connectionString="Data
Source=localhost;Initial Catalog=UniversityOfThessaly; user id=dimimach; password =
pass" providerName="System.Data.SqlClient" />
  </connectionStrings>
```

Table 1. Database Connection String

4.2.3 Application Models

After enabling the connection between application database and application web platform, the next we have to do, is to develop the appropriate *Models*, which represent the database structure. As it has been discussed in previous chapters, *Models, Views* and *Controllers* are connected to each other if a web application is to have a valid functionality. As a result, a number of classes had to be initialized with the corresponding database tables and table relations.

A *model class* has by default, always, the same structure. First, we create a *public class*, which has usually the same or similar name to the corresponding database table. Inside this class, we create the *class structure* and then we write the *class attributes*. Moreover, the corresponding *constructors* will be developed inside this class. For example, a model *Grade*, which represents a Submission Grade, has class attributes like: *ID, GradeValue, Comments, SubmID, GraderID* e.t.c.

Apart from this class, we also need to create another class in the model. This class should be a *partial* class and should contain the corresponding methods, which do execute the SQL Queries. Inside these methods, there are a lot of checks, according to the validity of the result that the query returns. Last but not least, a *Controller* which uses the database data, will be able to have access to them, only using these model methods.

First of all, we have to initialize a connection between the connection string that we have displayed on Table 1, and our web application. We can see the function that initializes the connection, on the next table (Table 2).

```
public static SqlConnection CreateConnectionCalc()
{
   return new
SqlConnection(WebConfigurationManager.ConnectionStrings["UniversityOfThessaly"].Con
nectionString);
}
```

Table 2. Database Connection Function

Moreover, in a lot of parts of this application, we use many existed model functions from models, which are installed by default on MVC 5. On the next tables (Table 3 and Table 4), we intend to display some code parts of Grade Model.

```
public class Grade
    {
        public int ID { get; set; }

        public string GradeValue { get; set; }

        [AllowHtml]
        public string Comments { get; set; }

        public int SubmID { get; set; }

        public int GraderID { get; set; }

        ....
    }
```

Table 3. Grade Model - Attributes

On the previous table, we presented some attributes of the Grade model. On the next one, we will present a method, which, using an *INSERT INTO*... SQL Query, inserts a row in the *GRADE* table. As we have previously described, we have to create a partial class, and inside this class we will write the *AddGrade* function.

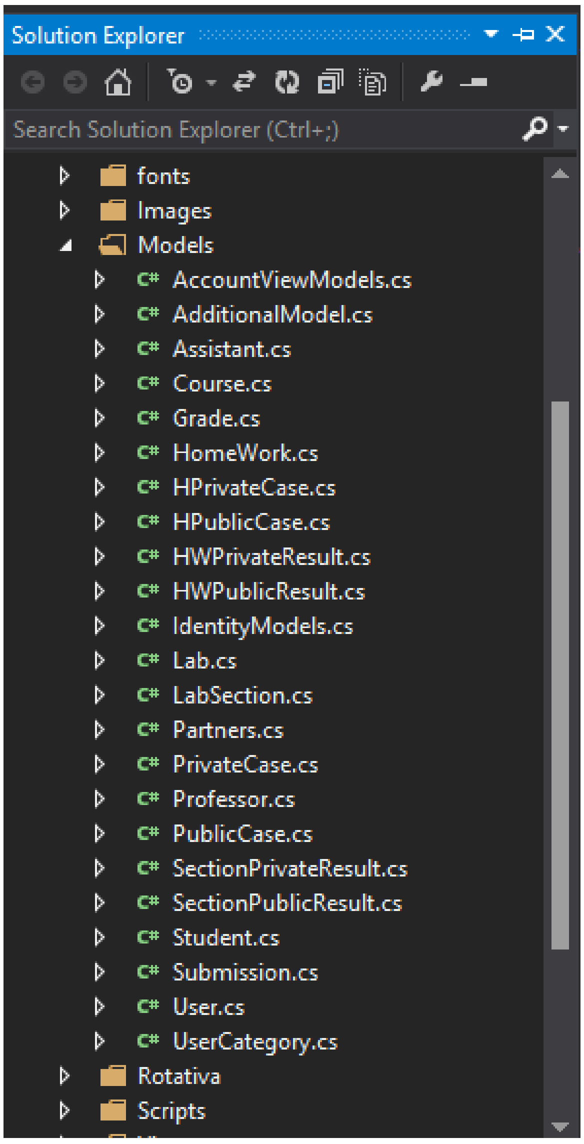
```
public Grade AddGrade(Grade grade, int student1, int student2, int labid)
{
  using (SqlConnection cnn = Manager.CreateConnectionCalc())
  {
     cnn.Open();
     using (var trans = cnn.BeginTransaction())
     {
        try
        {
            SqlCommand cmd = cnn.CreateCommand();
            cmd.Transaction = trans;
            cmd.Parameters.Clear();
            cmd.CommandType = CommandType.Text;
            cmd.Parameters.Clear();
            string sql = string.Empty;
            /*Content Grade*/
            sql = " INSERT INTO GRADES ";
            sql += " ( GRADE , COMMENTS , SUBID , GRADERID , ISPUBLISHED , STUDENT1
, STUDENT2 , LABID) ";
            sql += " VALUES ";
            sql += " ( '' , '', @subid  , @graderid , 0 , @student1 , @student2 ,
@labid ) ";
            sql += " SELECT cast(scope_identity() AS int)  ";
            cmd.Parameters.Add("@subid", SqlDbType.Int);
            cmd.Parameters["@subid"].Value = grade.SubmID;
            cmd.Parameters.Add("@graderid", SqlDbType.Int);
            cmd.Parameters["@graderid"].Value = grade.GraderID;
            cmd.Parameters.Add("@student1", SqlDbType.Int);
            cmd.Parameters["@student1"].Value = student1;
            cmd.Parameters.Add("@student2", SqlDbType.Int);
            cmd.Parameters["@student2"].Value = student2;
            cmd.Parameters.Add("@labid", SqlDbType.Int);
            cmd.Parameters["@labid"].Value = labid;
            cmd.CommandText = sql;
            SqlDataReader sdr = null;
            int newID = -1;
            sdr = cmd.ExecuteReader();

            if (sdr.Read())
            {
               newID = (int)sdr[0];
            }
            sdr.Close();
            grade.ID = newID;
            if (newID == -1)
```

```
                {
                    return null;
                }
                trans.Commit();
            }
            catch (Exception ex)
            {
                grade.ID = -1;
                trans.Rollback();
            }
        }
    }
    return grade;
}
```

Table 4. AddGrade Model Function

On the previous table, we observe how we can execute a sql query. First of all, we initialize a connection to the database using the function *CreateConnectionCalc()*, which we have already created. Afterwards, we define the parameters for the SQL query, and then we execute it. We use *try...catch* functionality, so as to find out whether there are any exceptions or not and if there are, how to handle them.

We have also created functions for *SELECT*, *UPDATE* and *DELETE* queries, which use *JOIN* functionality to a great extend. We will present our application models on the next figure (Figure 12).



```
Solution Explorer
Search Solution Explorer (Ctrl+;)
  fonts
  Images
  Models
    AccountViewModels.cs
    AdditionalModel.cs
    Assistant.cs
    Course.cs
    Grade.cs
    HomeWork.cs
    HPrivateCase.cs
    HPublicCase.cs
    HWPrivateResult.cs
    HWPublicResult.cs
    IdentityModels.cs
    Lab.cs
    LabSection.cs
    Partners.cs
    PrivateCase.cs
    Professor.cs
    PublicCase.cs
    SectionPrivateResult.cs
    SectionPublicResult.cs
    Student.cs
    Submission.cs
    User.cs
    UserCategory.cs
  Rotativa
  Scripts
```

Figure 12. Application Models

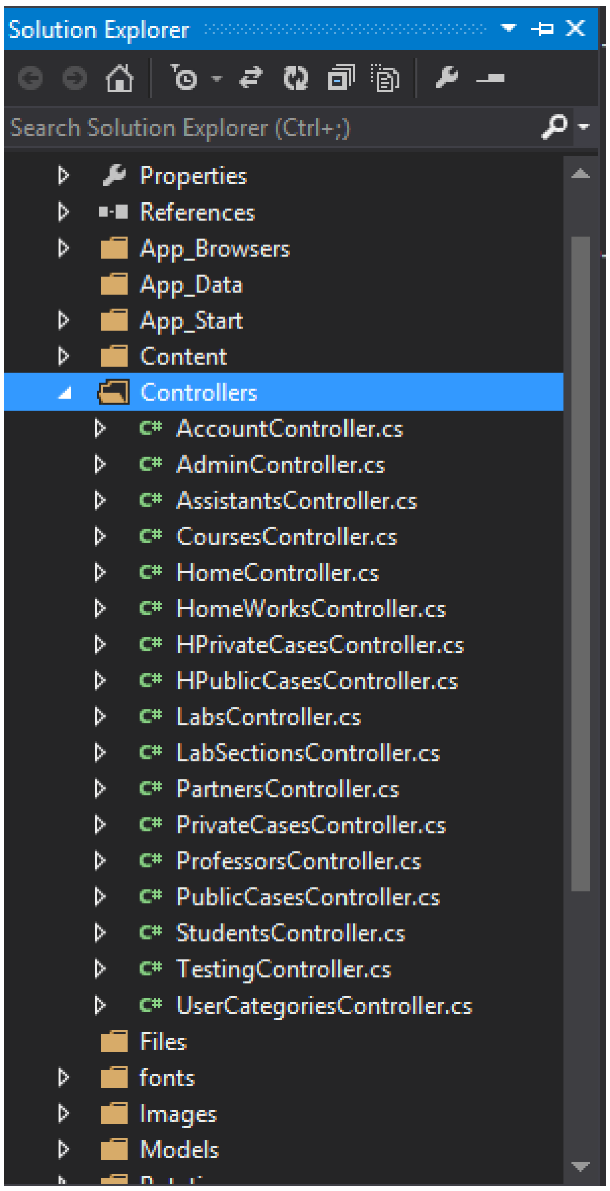
4.2.4 Application Controllers

In order to use model functions, we have to create *Controllers* according to the models that we have created. For the creation of controllers, we use only C# programming language, as they are characterized by methods, which can get data from html pages or set data on them. These functions are called *ActionResult* methods and they can mirror *Views* or simple functions, which have no Views. Inside these functions, we create model objects and we use their attributes and their functions so as to return application data on html pages. As we have already noticed, on controllers, we can both get and set data, not only from Models but also from html pages (Views). For example, there are a lot of times, that we get data from html forms and then we post it to *ActionResult* methods. These methods may call other functions, which we have initialized on application models. On the next table (Table 5), we can see a *Post ActionResult* method, which adds a new grade on the database.

```
[HttpPost]
[ValidateInput(false)]
[Authorize(Roles = "ADMIN, ASSISTANT,PROFESSOR")]
public ActionResult AddGrade(Grade grade, FormCollection formcollection)
{
    Grade newGrade = new Grade();
    int gradeid = Convert.ToInt16(formcollection["gradeid"]);
    int labid = Convert.ToInt16(formcollection["labid"]);
    grade.ID = gradeid;
    newGrade = m.UpdateGrade(grade);
    if (newGrade.ID != -1)
    {
        ViewBag.Success = "You have successfully graded your submission";
    }
    else
    {
        ViewBag.Success = "Something was wrong.";
    }

    ViewBag.LabID = labid;
    ViewBag.Gradeid = gradeid;
    return View(grade);
}
```

Table 5. AddGrade ActionResult Method

On the previous table, we can see AddGrade Action Result and some C# objects, which we need in order to insert the grade on the database. It is also obvious, that this method can only be used by the *ADMIN*, *ASSISTANT* and *PROFESSOR* roles. We can also see, that the return type of this ActionResult is a View(), and inside this return type, we return a *Grade* object. This means, that we must declare the model on the corresponding view. On the next figure (Figure 13), are displayed the application Controllers.


Solution Explorer

Search Solution Explorer (Ctrl+;)

- Properties
- References
- App_Browsers
- App_Data
- App_Start
- Content
- Controllers
  - AccountController.cs
  - AdminController.cs
  - AssistantsController.cs
  - CoursesController.cs
  - HomeController.cs
  - HomeWorksController.cs
  - HPrivateCasesController.cs
  - HPublicCasesController.cs
  - LabsController.cs
  - LabSectionsController.cs
  - PartnersController.cs
  - PrivateCasesController.cs
  - ProfessorsController.cs
  - PublicCasesController.cs
  - StudentsController.cs
  - TestingController.cs
  - UserCategoriesController.cs
- Files
- fonts
- Images
- Models

Figure 13. Application Controllers

Furthermore, we have paid much attention to the controllers functionality, as we use a lot of functions which create and edit files, folders, processes and relative paths. For example, in order to fulfill the whole compilation and moss functionality, we had to create and write the appropriate commands on batch files. We intend to analyze the Compilation functionality in 4.2.6 Compilation & Testing Functionality and the Moss functionality in 4.2.7 Moss Functionality.

4.2.5 Application Views

An *ActionResult Method* has usually one corresponding *View*, which displays whatever the ActionResult function returns. If this method returns a view, which has a model object as parameter, then we must declare it at the beginning of the View file. As a result, we will be able to use the returning data of C# on the view page, which is an html page. Consequently, we will be able to write C# code on html code. In Appendix A, we can see the code of the AddGrade View.

On the next figure (Figure 14), we present the folders, which contain the application Views.

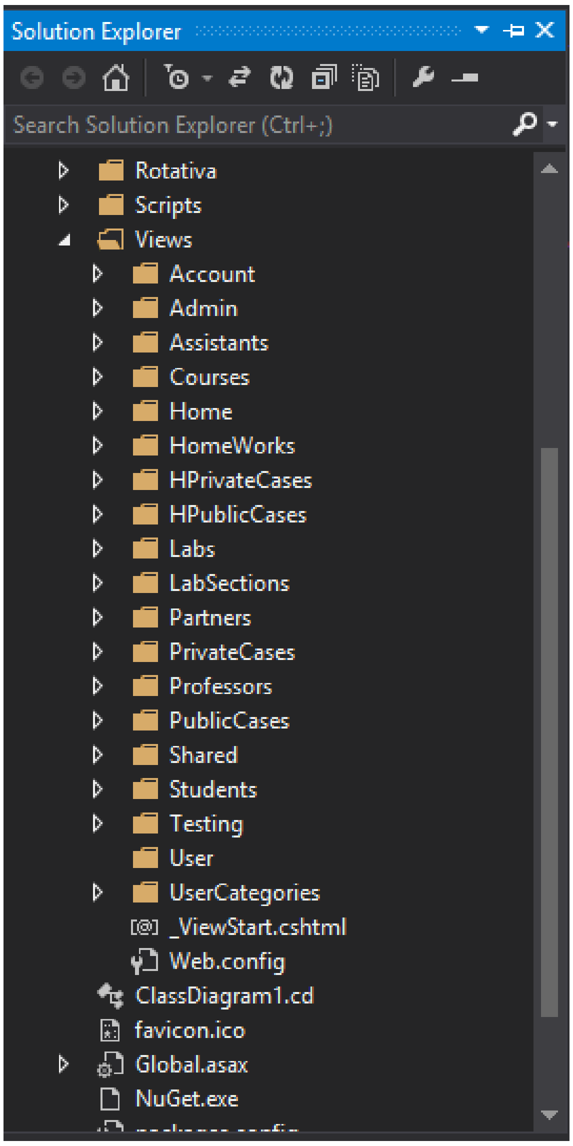


Figure 14. Application Views

4.2.6 Compilation & Testing Functionality - Security Considerations

The compilation functionality is a very significant part of our web platform. In particular, we write the *gcc* command with its appropriate parameters, which will compile the submission file for errors and warnings, whose results we write in a .txt file. If the compiled file has no errors, we get the executable file, and for every public and private case, we write an executable command using the corresponding command line. Moreover, we also write test case results in .txt files. In order to check the similarity of the results between the output files that the professor has given while uploading the test case and the files, where we have saved the results of the submission test cases, we use *diff* Linux command. Using this command, we are able to view, whether the results are the same or not. After saving these commands in the batch file, we run it using a C# process and then we get the corresponding results. Last but not least, due to the fact that we do not know what kind of code we compile and run, we follow some security rules for the whole compilation functionality. In order for our application to be safe from runtime errors, we compile each submission on a specific folder, which is out of our application folders. Afterwards, as it comes to executable files from other users, and we are not able to know whether they are dangerous for the functionality of our platform or not, we create a jail directory for the executables, so that they do not have access outside it. Additionally, we enforce file size quotas, so that the execution does not fill the disk. There is also a watchdog timer that kills the process if it runs for too long. Then, if everything is all right, we copy the compilation files from this folder to our application folders, so as for professors, graders and students to have access to their results.

4.2.7 Moss Functionality

The moss functionality is another very significant part of our web platform. First, we execute moss script, which we have already saved on a specific folder of our platform. Then, we save the url, that the command returns, in a .txt file. Afterwards, we rip Moss site, using *wget* command, with its parameters and the url, written on the previous .txt file. In conclusion, we save moss results on our platform, as they are available online only for 20 days, and then, they are deleted.

On the next table (Table 6), we can see how we start a process, which runs a batch file. The batch file contains either compilation or moss commands.

```
System.Diagnostics.Process process = new System.Diagnostics.Process();
System.Diagnostics.ProcessStartInfo startInfo = new
System.Diagnostics.ProcessStartInfo();
startInfo.WindowStyle=System.Diagnostics.ProcessWindowStyle.Hidden;


... we write here code for the batch file


startInfo.FileName = "compilationfile.bat";
startInfo.WorkingDirectory = path;
process.StartInfo = startInfo;
process.Start();
```

Table 6. Process Run

As we can see, we must first create the batch file, then write in it what is necessary and afterwards, set the relative path, where the server must run this file. After starting the process, which executes the batch file, we have to check when the process exits and then to see the results.

5 Deployment and Results

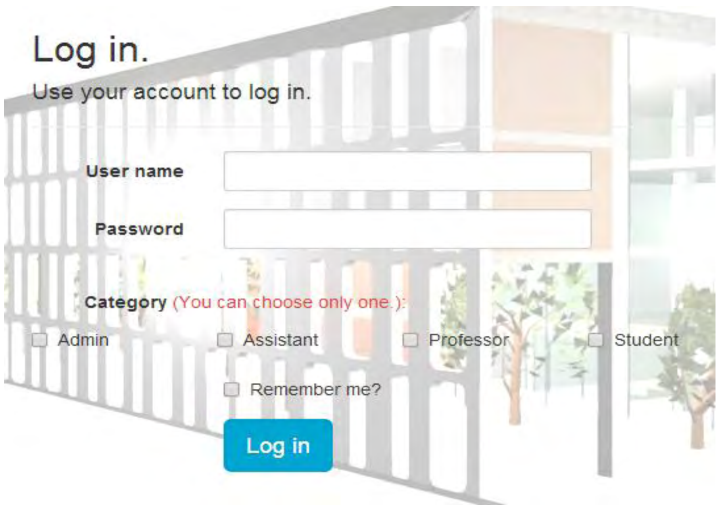
In this chapter we intend to discuss the deployment and the results of our web platform, as a user guide.

5.1 User Interface Model

First, the screen of our platform is divided into 2 panels. On the left panel, we have the user's menu and on the right panel we have the user's main work area. Moreover, we follow a systematic color coding on our buttons. A yellow button means that the user should do something (should click on it), a green button means that the user has already done it and a gray button means that the user is not permitted to perform an action.

5.2 User Login

First, a user should log in on the web application so as to have access to the functionality of his role. So he must log in using his credentials, which were given by the web application administrator. On the next figure (Figure 15), we can see how a user can log in on our web application, using the log in form.


Log in.

Use your account to log in.

User name

Password

Category (You can choose only one.):

☐ Admin ☐ Assistant ☐ Professor ☐ Student

☐ Remember me?

Figure 15. Log In - Form

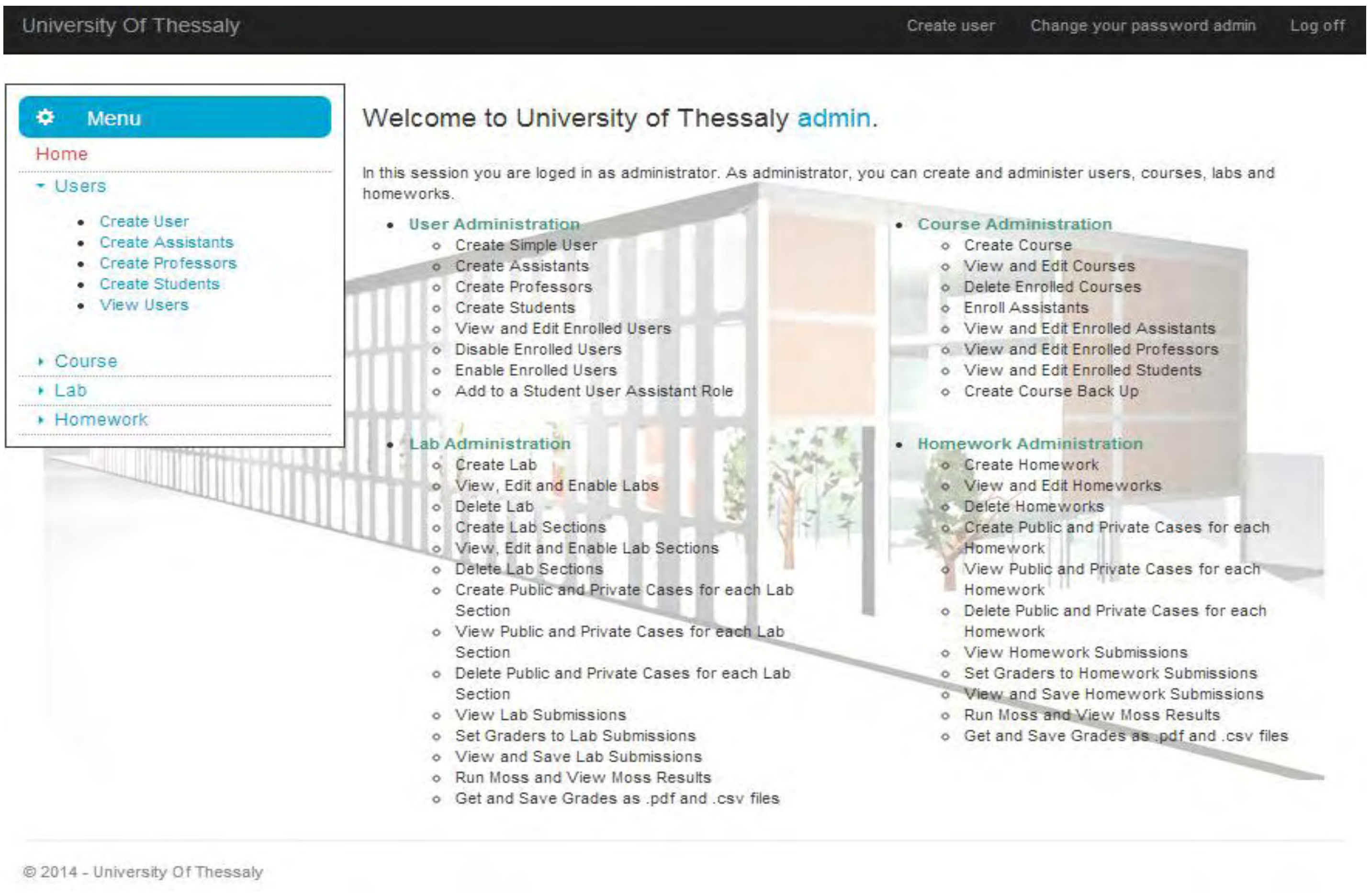
What we can easily notice on the previous figure is that the user should choose a category so as to log in on the application. If the user gives wrong credentials, or if he does not choose the correct category, an error message will be immediately displayed on his screen.

5.3 Administrator's Panel

On this section we present the administrator's functionality. We start right on with the administrator's home page.

5.3.1 Administrator's Home Page

After an administrator user logs in on this web application, he will see the administrator's home page, which is displayed on the next figure (Figure 16).



Welcome to University of Thessaly admin.

In this session you are loged in as administrator. As administrator, you can create and administer users, courses, labs and homeworks.

- User Administration
  - Create Simple User
  - Create Assistants
  - Create Professors
  - Create Students
  - View and Edit Enrolled Users
  - Disable Enrolled Users
  - Enable Enrolled Users
  - Add to a Student User Assistant Role
- Course Administration
  - Create Course
  - View and Edit Courses
  - Delete Enrolled Courses
  - Enroll Assistants
  - View and Edit Enrolled Assistants
  - View and Edit Enrolled Professors
  - View and Edit Enrolled Students
  - Create Course Back Up
- Lab Administration
  - Create Lab
  - View, Edit and Enable Labs
  - Delete Lab
  - Create Lab Sections
  - View, Edit and Enable Lab Sections
  - Delete Lab Sections
  - Create Public and Private Cases for each Lab Section
  - View Public and Private Cases for each Lab Section
  - Delete Public and Private Cases for each Lab Section
  - View Lab Submissions
  - Set Graders to Lab Submissions
  - View and Save Lab Submissions
  - Run Moss and View Moss Results
  - Get and Save Grades as .pdf and .csv files
- Homework Administration
  - Create Homework
  - View and Edit Homeworks
  - Delete Homeworks
  - Create Public and Private Cases for each Homework
  - View Public and Private Cases for each Homework
  - Delete Public and Private Cases for each Homework
  - View Homework Submissions
  - Set Graders to Homework Submissions
  - View and Save Homework Submissions
  - Run Moss and View Moss Results
  - Get and Save Grades as .pdf and .csv files

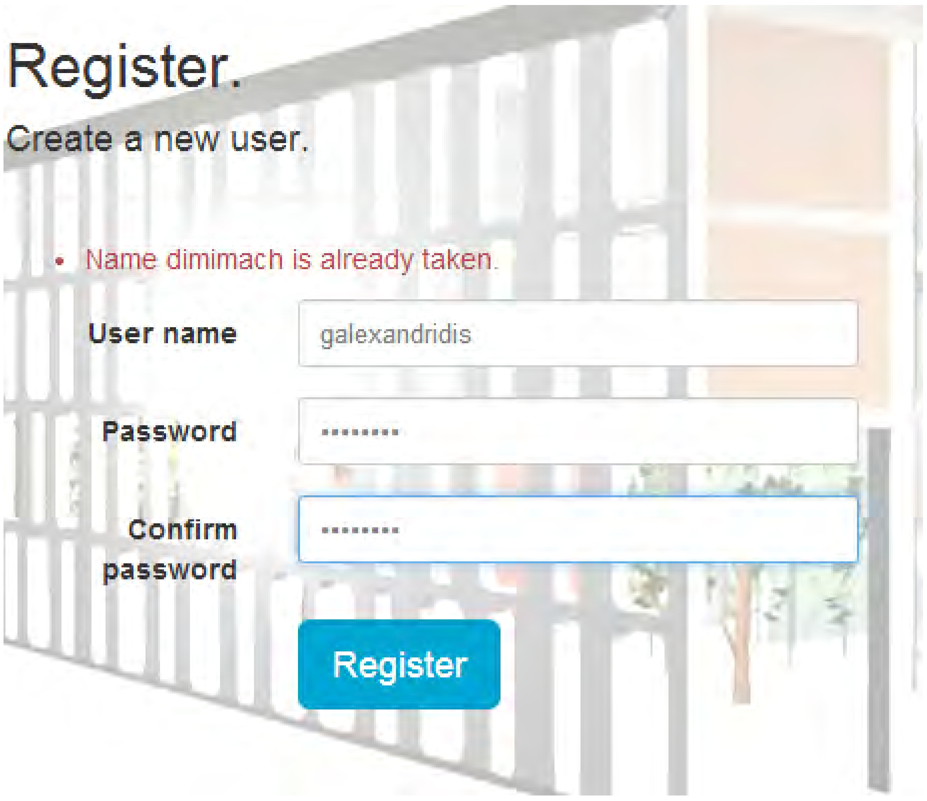
© 2014 - University Of Thessaly

Figure 16. Administrator's Home Page

On this page, an administrator can see lists of the available actions according to his role. On the left side of the screen, there is a menu presenting his role's available actions. As we can see on the top menu panel, an administrator also has the opportunity to change his password.

5.3.2 Create User

An administrator can create a new application user. First, he has to give the username and the password of the user, afterwards to choose the user's role and at the end to fill in the user's personal data. In case that a username already exists, an error message informs the administrator so that he can change it. On the next figures (Figure 17, Figure 18 and Figure 19) we can see how an administrator can add a new user.



Register.

Create a new user.

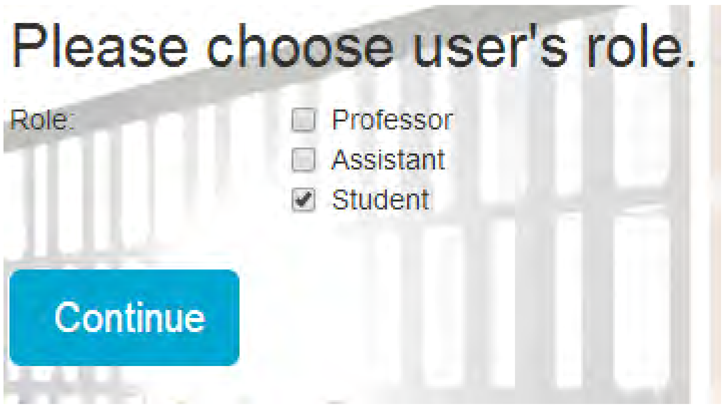
- Name dimimach is already taken.

User name	galexandridis
Password	••••••••
Confirm password	••••••••

Figure 17. User Registration

On the previous figure, we had already a user with the name *dimimach*, so we had to change the username. Consequently, after filling in the new data and clicking on the

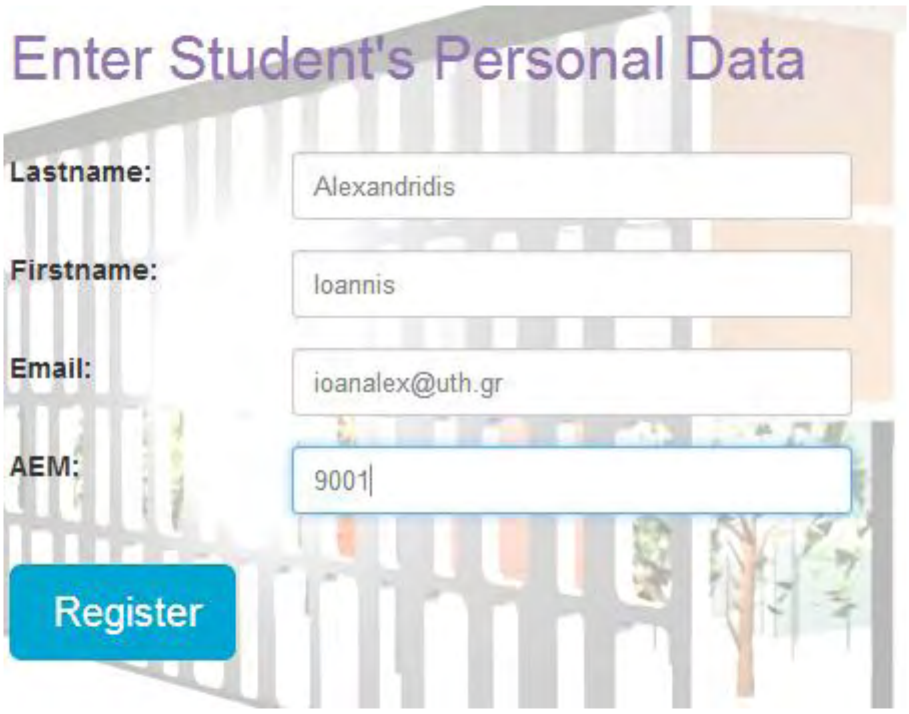
Register button, the administrator gets the results of Figure 18, and he can choose the role of the user.


Please choose user's role.

Role: [ ] Professor [ ] Assistant [✓ Student]

Figure 18. Role Choice

After choosing the role, the administrator has to enter the user's personal data, as it is presented on the next figure (Figure 19).


Enter Student's Personal Data

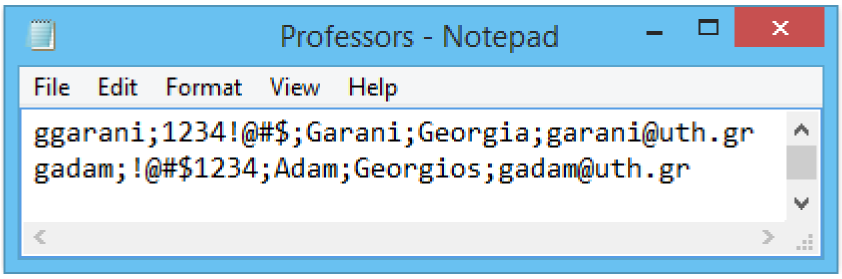
Lastname:	Alexandridis
Firstname:	Ioannis
Email:	ioanalex@uth.gr
AEM:	9001

Figure 19. User's Personal Data

If there are no exceptions, after registering, the new user has successfully enrolled in our application, and a success message is immediately displayed on the administrator's screen.

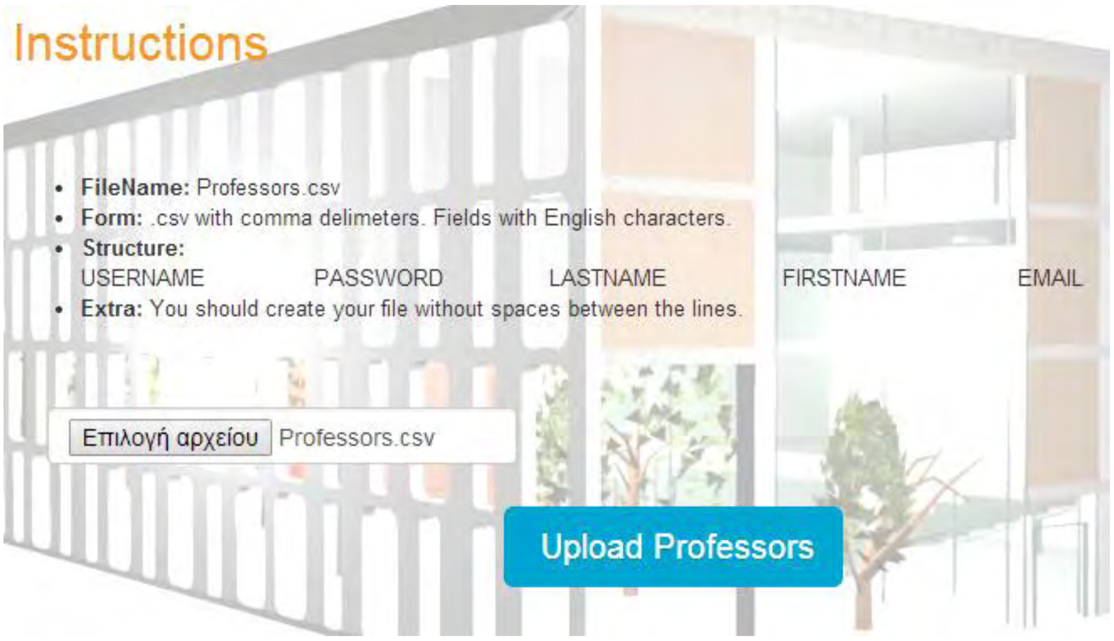
5.3.3 Create Assistants, Professors and Students in Groups

Due to the fact that an administrator is not able to save new entries one by one, an Administrator can add all users (*Assistants*, *Professors*, *Students)* either individually or in groups, using .csv files. When it comes to groups, he must follow the instructions concerning the file structure. On the next figure (Figure 20), we can see such a .csv file, which includes the professors' data, and on Figure 21 how an administrator can upload professors on our web application.



```
ggarani;1234!@#$;Garani;Georgia;garani@uth.gr
gadam;!@#$1234;Adam;Georgios;gadam@uth.gr
```

Figure 20. Professors' Data .csv File

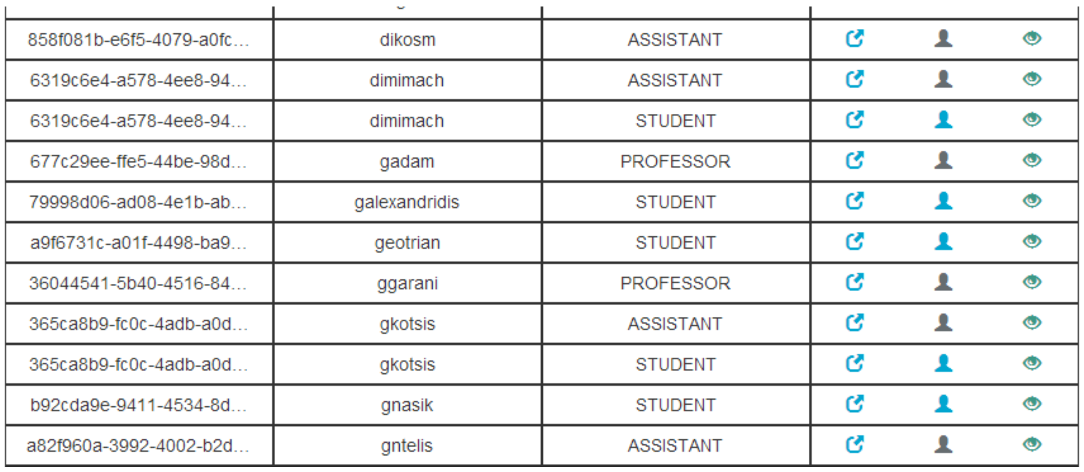


Instructions

- **FileName:** Professors.csv
- **Form:** .csv with comma delimeters. Fields with English characters.
- **Structure:**
  USERNAME PASSWORD LASTNAME FIRSTNAME EMAIL
- **Extra:** You should create your file without spaces between the lines.

Figure 21. Upload Professors using .csv File

Additionally, we would like to mention that we always check the content of the file before adding users on our platform. For example, if some same usernames have been chosen, we display to the administrator a message with the already existing usernames. If everything is ok, the administrator will be redirected to the *View Users* page, where he will be able to view and edit the user's information and roles. We can see the *View Users* page on next figure (Figure 22).



858f081b-e6f5-4079-a0fc...	dikosm	ASSISTANT	↗	👤	👁
6319c6e4-a578-4ee8-94...	dimimach	ASSISTANT	↗	👤	👁
6319c6e4-a578-4ee8-94...	dimimach	STUDENT	↗	👤	👁
677c29ee-ffe5-44be-98d...	gadam	PROFESSOR	↗	👤	👁
79998d06-ad08-4e1b-ab...	galexandridis	STUDENT	↗	👤	👁
a9f6731c-a01f-4498-ba9...	geotrian	STUDENT	↗	👤	👁
36044541-5b40-4516-84...	ggarani	PROFESSOR	↗	👤	👁
365ca8b9-fc0c-4adb-a0d...	gkotsis	ASSISTANT	↗	👤	👁
365ca8b9-fc0c-4adb-a0d...	gkotsis	STUDENT	↗	👤	👁
b92cda9e-9411-4534-8d...	gnasik	STUDENT	↗	👤	👁
a82f960a-3992-4002-b2d...	gntelis	ASSISTANT	↗	👤	👁

Figure 22. Application Users

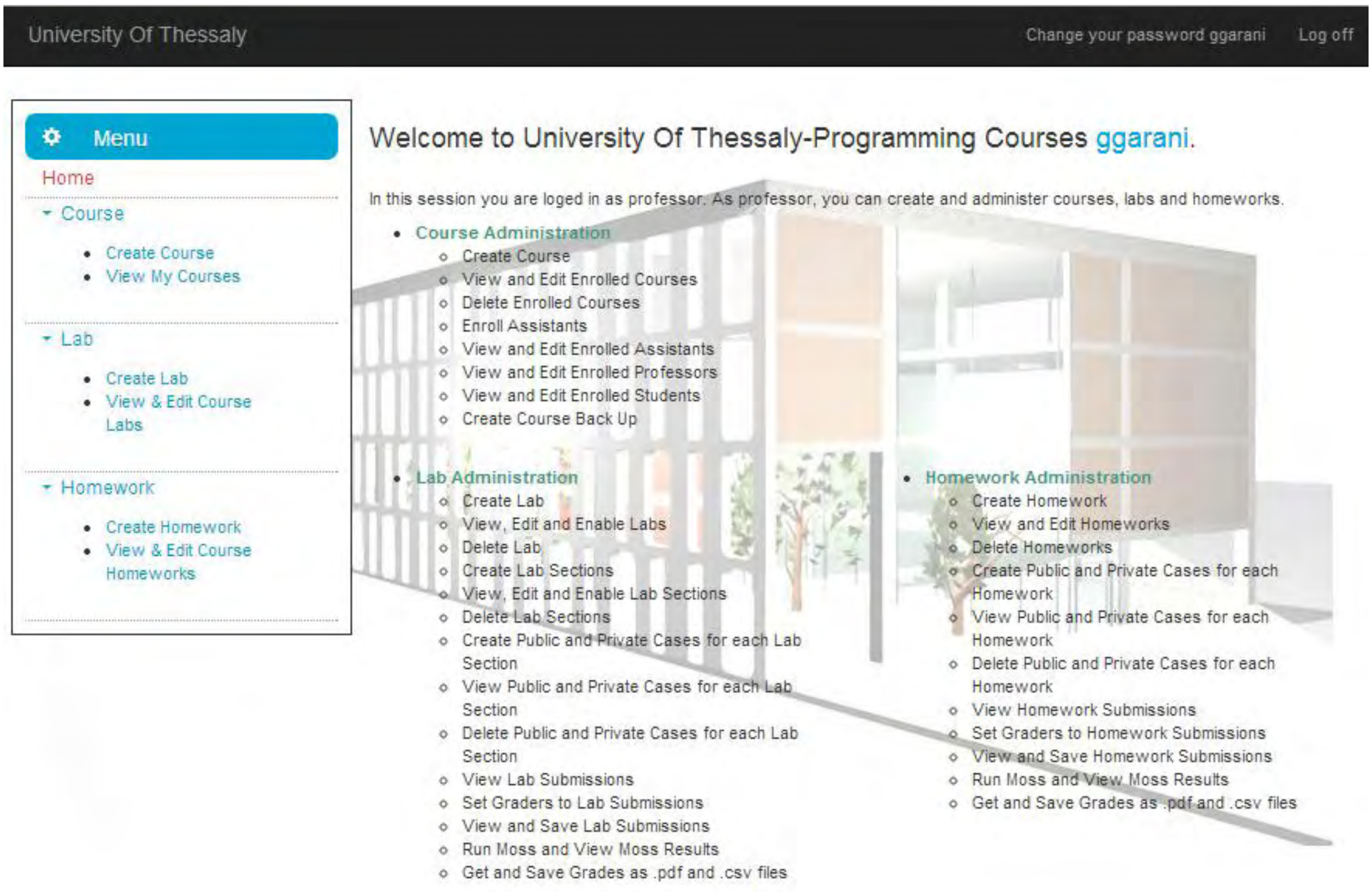
Last but not least, due to the fact that a professor can administer courses, labs and homeworks the same way an administrator does, we display their functionality in 5.4 Professor's Panel.

5.4 Professor's Panel

On this section we will present the professor's functionality. This functionality can also be performed by an administrator. We start with professor's home page.

5.4.1 Professor's Home Page

After a professor user logs in on the web application, he will see the professor's home page, which is displayed on the next figure (Figure 23).



Welcome to University Of Thessaly-Programming Courses ggarani.

In this session you are loged in as professor. As professor, you can create and administer courses, labs and homeworks.

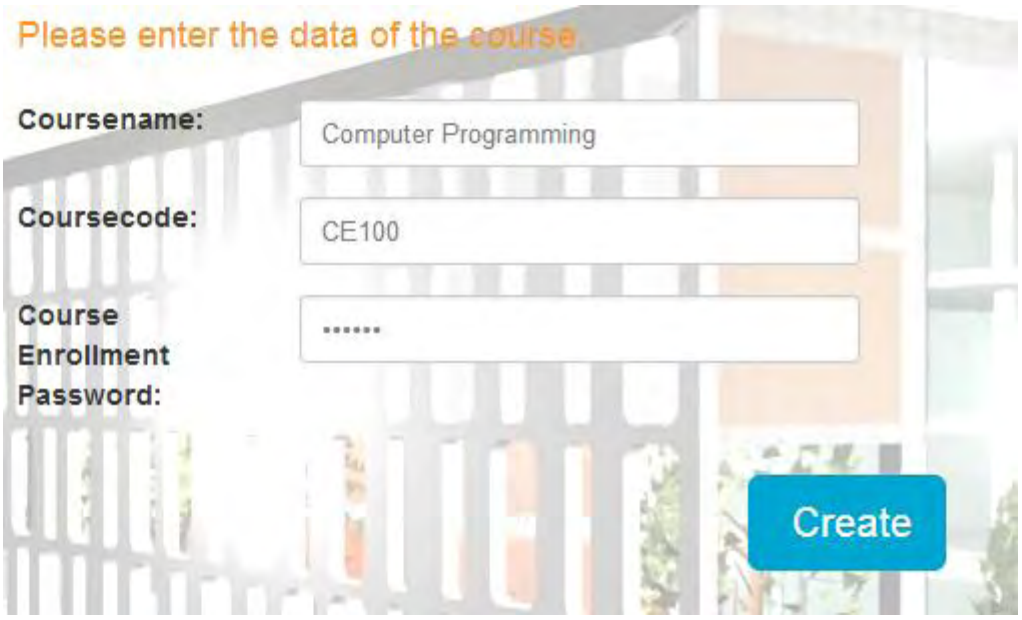
- Course Administration
  - Create Course
  - View and Edit Enrolled Courses
  - Delete Enrolled Courses
  - Enroll Assistants
  - View and Edit Enrolled Assistants
  - View and Edit Enrolled Professors
  - View and Edit Enrolled Students
  - Create Course Back Up
- Lab Administration
  - Create Lab
  - View, Edit and Enable Labs
  - Delete Lab
  - Create Lab Sections
  - View, Edit and Enable Lab Sections
  - Delete Lab Sections
  - Create Public and Private Cases for each Lab Section
  - View Public and Private Cases for each Lab Section
  - Delete Public and Private Cases for each Lab Section
  - View Lab Submissions
  - Set Graders to Lab Submissions
  - View and Save Lab Submissions
  - Run Moss and View Moss Results
  - Get and Save Grades as .pdf and .csv files
- Homework Administration
  - Create Homework
  - View and Edit Homeworks
  - Delete Homeworks
  - Create Public and Private Cases for each Homework
  - View Public and Private Cases for each Homework
  - Delete Public and Private Cases for each Homework
  - View Homework Submissions
  - Set Graders to Homework Submissions
  - View and Save Homework Submissions
  - Run Moss and View Moss Results
  - Get and Save Grades as .pdf and .csv files

Figure 23. Professor's Home Page

On this page, a professor can see lists of the available actions for him according to his role. On the left side of the screen, there is a menu displaying the role's available actions. As we can see on the top menu panel, the professor user can also change his password.

5.4.2 Create Course and Edit Course Information

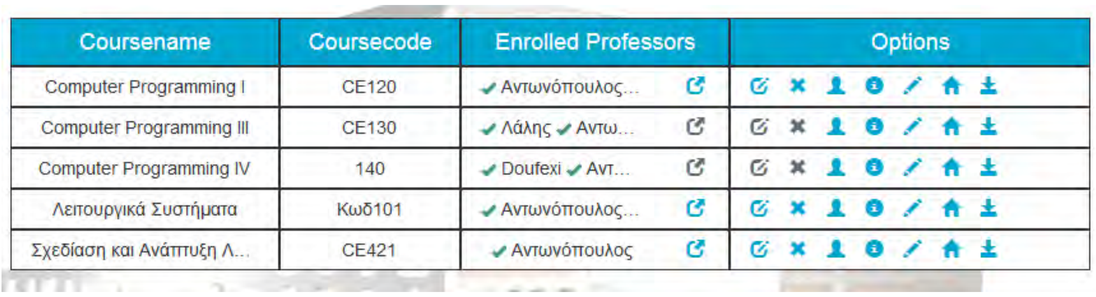
A professor can create and edit courses. He should first give some information about the course, such as a *Name*, a *Code* and a *Password*. On the next figure (Figure 24), we can see how a professor fills in the information during the creation of the course .


Please enter the data of the course

Coursename:	Computer Programming
Coursecode:	CE100
Course Enrollment Password:	••••••

Figure 24. Course Creation

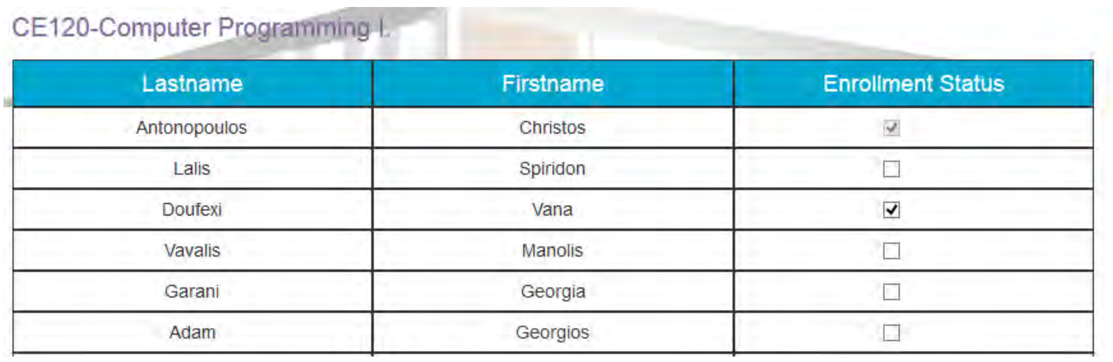
After the creation of the course, the professor will be redirected to the *View Courses* page, where he can see which courses he manages. Furthermore, on this page the professor can edit course information as well as the information of enrolled assistants, professors and students. He is not only able to add new labs or homeworks for the course, but he can also retract the course or even delete it. But if a professor is not the creator of the course, he cannot edit either the course information or the professors who manage it and of course, he cannot delete the course. On the next figure (Figure 25), we can see the courses list of a professor.



Coursename	Coursecode	Enrolled Professors	Options
Computer Programming I	CE120	✓ Αντωνόπουλος...	
Computer Programming III	CE130	✓ Λάλης ✓ Αντω...	
Computer Programming IV	140	✓ Doufexi ✓ Αντ...	
Λειτουργικά Συστήματα	Κωδ101	✓ Αντωνόπουλος...	
Σχεδίαση και Ανάπτυξη Λ...	CE421	✓ Αντωνόπουλος	

Figure 25. Professor's Course List

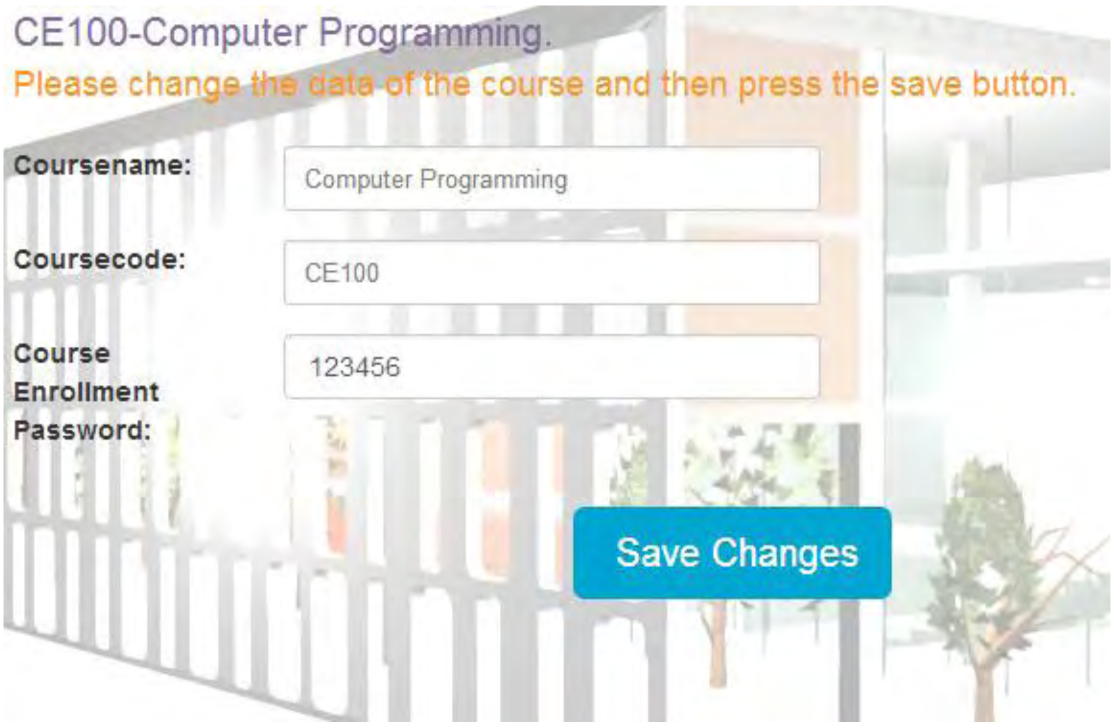
The corresponding options for each course are being presented in each table row. First, as we have previously mentioned, a professor is able to edit the professors who manage the course. On the next figure (Figure 26), we will display this procedure.


CE120-Computer Programming I.

Lastname	Firstname	Enrollment Status
Antonopoulos	Christos	✓
Lalis	Spiridon	☐
Doufexi	Vana	✓
Vavalis	Manolis	☐
Garani	Georgia	☐
Adam	Georgios	☐

Figure 26. Edit Professors

A professor is able to edit in the same way enrolled assistants and students. He is also able to change the course information, as we can see on the next figure (Figure 27).

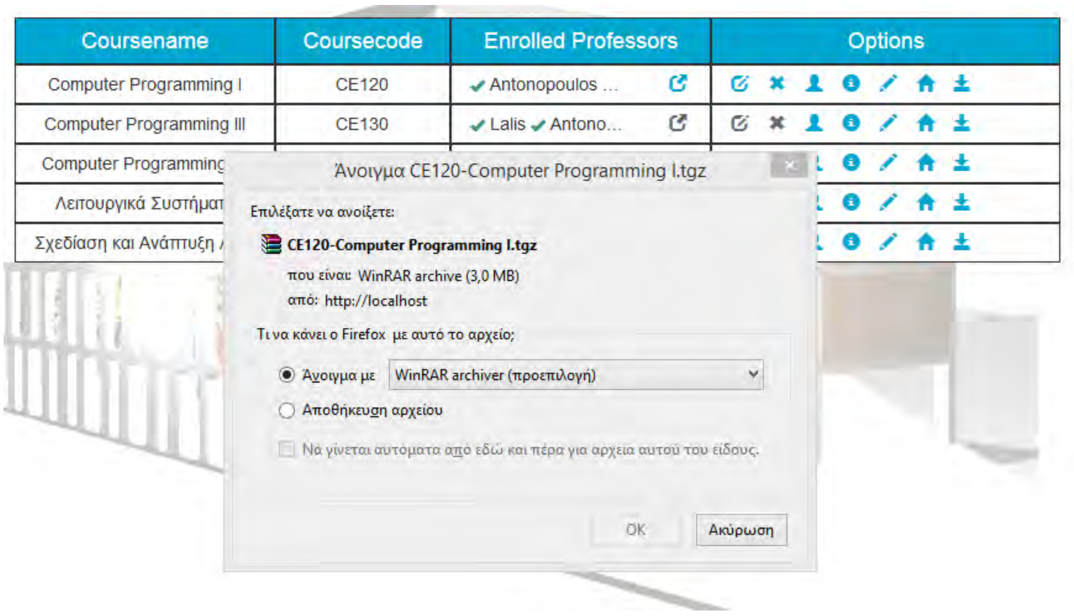

CE100-Computer Programming.

Please change the data of the course and then press the save button.

Coursename:	Computer Programming
Coursecode:	CE100
Course Enrollment Password:	123456

Figure 27. Edit Course Information

Such a user can also delete the course as well as add and edit new labs and homeworks. He can also get a course backup as a zip file, as can be seen on the next figure (Figure 28).



Coursename	Coursecode	Enrolled Professors	Options
Computer Programming I	CE120	✓ Antonopoulos ...	
Computer Programming III	CE130	✓ Lalis ✓ Antono...	
Computer Programming			
Λειτουργικά Συστήματ			
Σχεδίαση και Ανάπτυξη /			

Άνοιγμα CE120-Computer Programming I.tgz

Επιλέξατε να ανοίξετε:

CE120-Computer Programming I.tgz

που είναι: WinRAR archive (3,0 MB)

από: http://localhost

Τι να κάνει ο Firefox με αυτό το αρχείο;

● Άνοιγμα με WinRAR archiver (προεπιλογή)

○ Αποθήκευση αρχείου

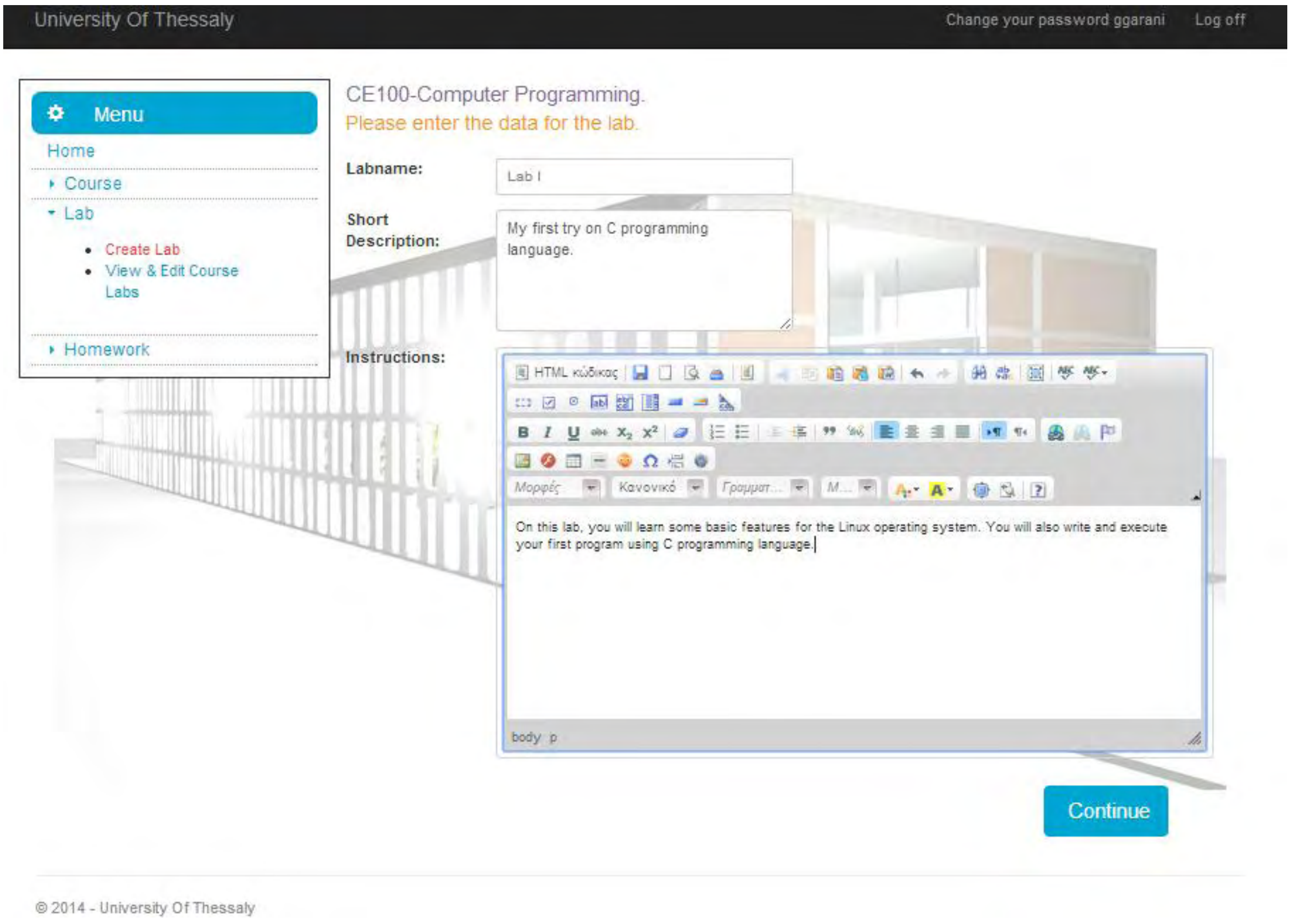
☐ Να γίνεται αυτόματα από εδώ και πέρα για αρχεία αυτού του είδους.

Figure 28. Get Course Back Up

5.4.3 Create Labs and Homeworks

On this section we will present the labs and homeworks administration. Due to the fact that both of them have the same functionality, we will present it at the same time.

First of all, professors should first choose the course and then fill in some information about the lab or the homework. On the next figure (Figure 29), we can see how professors can fill in information about a lab.



CE100-Computer Programming.

Please enter the data for the lab.

Labname: Lab I

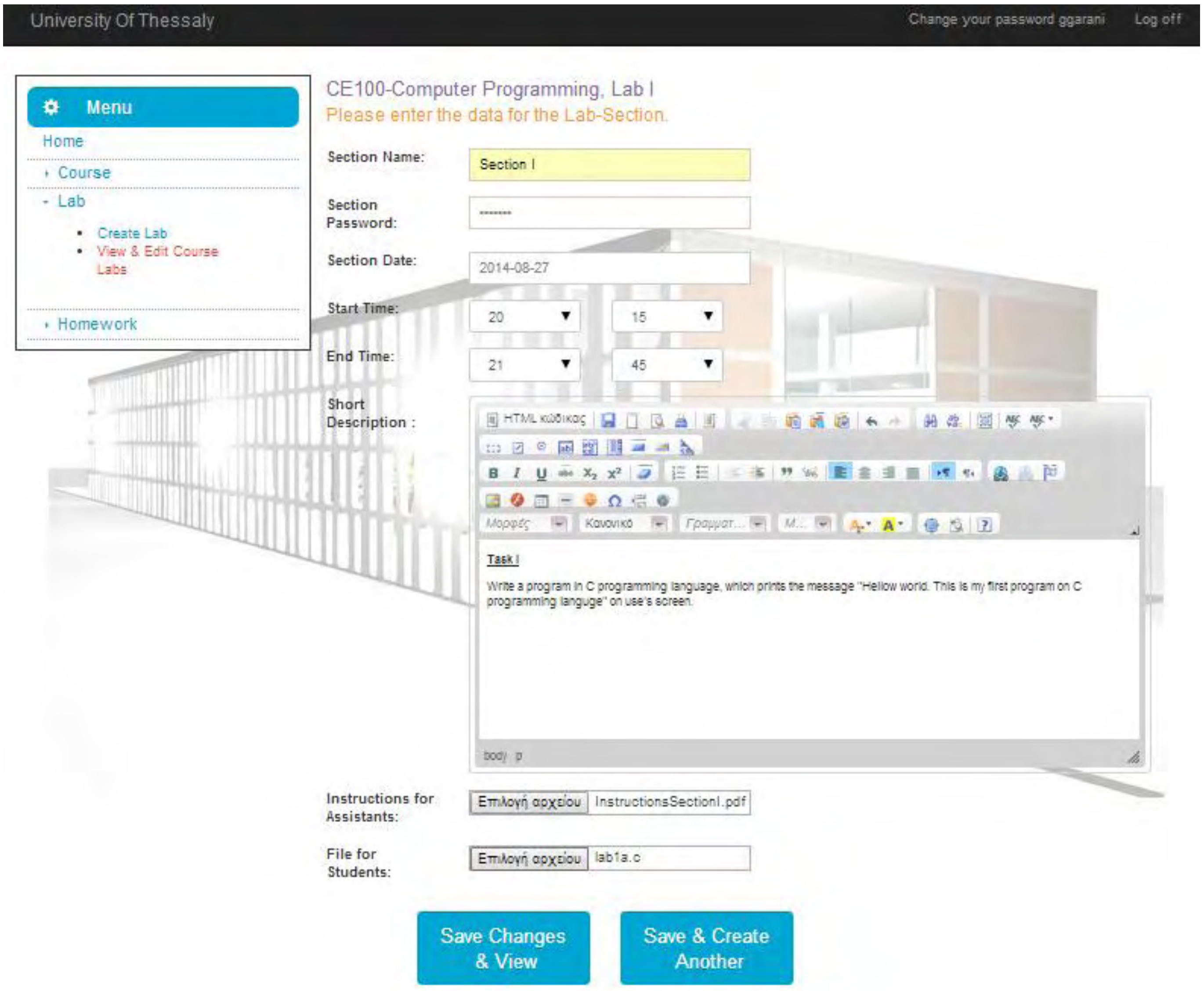
Short Description: My first try on C programming language.

Instructions: On this lab, you will learn some basic features for the Linux operating system. You will also write and execute your first program using C programming language.

Figure 29. Create Course Lab

5.4.3.1 Lab Section Creation

After entering the information concerning the lab, the professor has to create some sections for the week labs. The user is not obliged to do the same when it comes to homeworks. On the next figure (Figure 30), we present the creation of a course lab section.



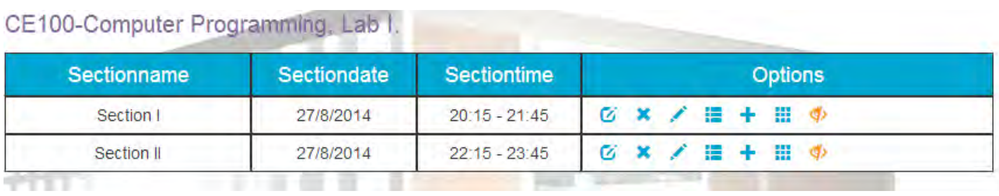
CE100-Computer Programming, Lab I

Please enter the data for the Lab-Section.

Section Name:	Section I
Section Password:	*******
Section Date:	2014-08-27
Start Time:	20 : 15
End Time:	21 : 45
Short Description :	**Task I** Write a program in C programming language, which prints the message "Hellow world. This is my first program on C programming languge" on use's screen.
Instructions for Assistants:	Επιλογή αρχείου InstructionsSectionI.pdf
File for Students:	Επιλογή αρχείου lab1a.c

Figure 30. Lab Section Creation

After saving the changes, the professor has the opportunity to create another lab section and view all of them together. On the next figure (Figure 31), we can view a list of the sections of a lab.



CE100-Computer Programming, Lab I.

Sectionname	Sectiondate	Sectiontime	Options
Section I	27/8/2014	20:15 - 21:45	
Section II	27/8/2014	22:15 - 23:45	

Figure 31. Lab Sections List

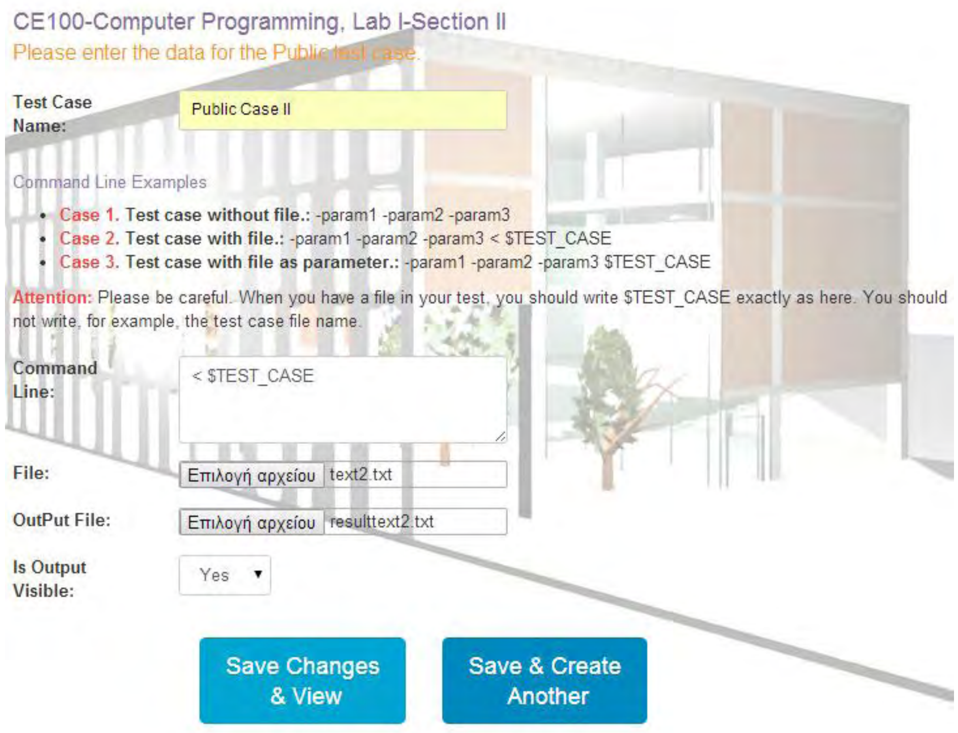
As we can see on the previous figure, we can view, edit, delete, enable the section, as well as add and view private and public test cases. As can also be seen, each section has its own public and private cases. A student is able to view the public cases, only after they are enabled by the professor.

5.4.3.2 Lab Section Enabling

First of all, a Professor has to enable the lab sections of every lab and afterwards the whole lab. Each lab section will be visible to enrolled students, for the time that the professor has defined while creating it, and only if both lab and lab section are also enabled. In order to enable a lab section, the professor has to press the icon that looks like a yellow eye. We can easily notice it on the previous figure (Figure 31). After enabling it, the eye will turn green. A Professor has to follow exactly the same steps for the corresponding lab.

5.4.3.3 Public and Private Test Case Creation

On the next figure (Figure 32), we can see how a professor can add a new public case.


CE100-Computer Programming, Lab I-Section II

Please enter the data for the Public test case.

Test Case Name: Public Case II

Command Line Examples

- Case 1. Test case without file.: -param1 -param2 -param3
- Case 2. Test case with file.: -param1 -param2 -param3 < $TEST_CASE
- Case 3. Test case with file as parameter.: -param1 -param2 -param3 $TEST_CASE

Attention: Please be careful. When you have a file in your test, you should write $TEST_CASE exactly as here. You should not write, for example, the test case file name.

Command Line: < $TEST_CASE

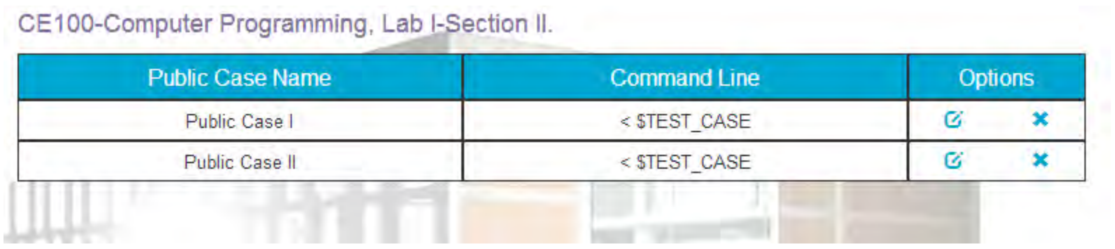
File: Επιλογή αρχείου text2.txt

OutPut File: Επιλογή αρχείου resulttext2.txt

Is Output Visible: Yes

Figure 32. Create Public Test Case

As can be seen, we give instructions to professors on how to write command line and also the ability, whether to make it visible for students to have access on the output file or not. We can create another test case like on lab sections, or view the public test cases for these sections and edit them. Last but not least, we make exactly the same actions, when it comes to a private test case, apart from the visibility of output file. Lab section private cases and its contents, are not visible to students as public cases. On the next figure (Figure 33), we display a list with public test cases and their options.



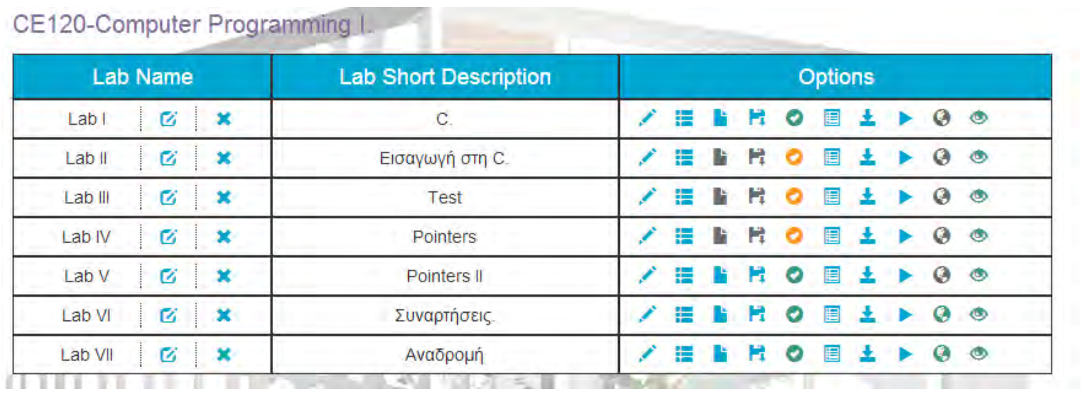
CE100-Computer Programming, Lab I-Section II.

Public Case Name	Command Line	Options
Public Case I	< $TEST_CASE	✎ ✖
Public Case II	< $TEST_CASE	✎ ✖

Figure 33. Public Test Cases List

5.4.4 Lab and Homework Administration

Course professors have the ability to edit and delete labs and homeworks the same way that they edit and delete courses. They can also create and view lab sections and their contents, get grades as .csv and .pdf files, view and get submissions as .zip files, set graders to submissions, publish grades to students and run moss script. On the next figure (Figure 34), we present a list of labs and their actions.



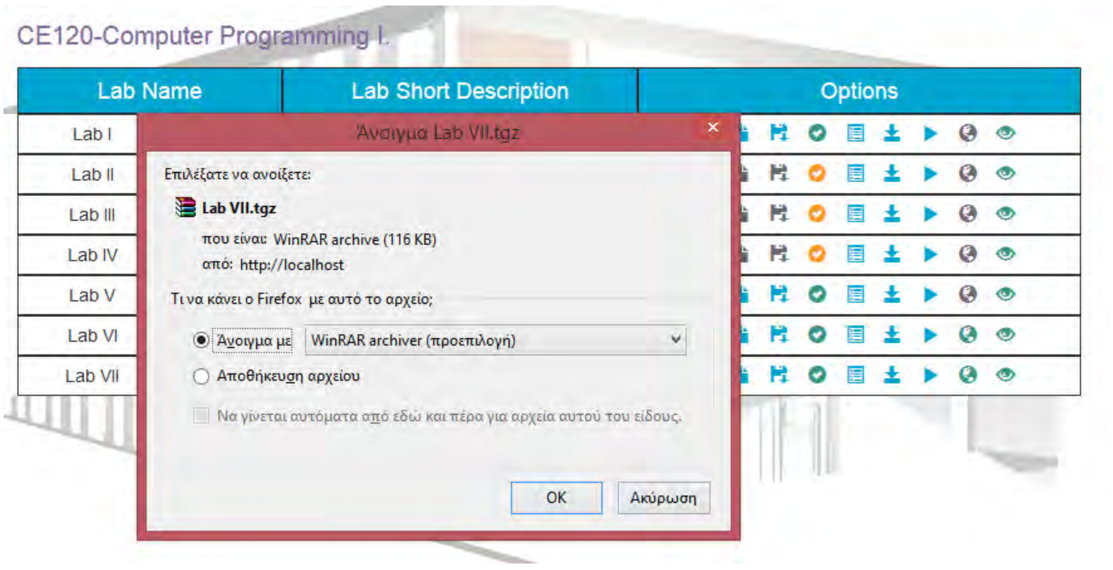
CE120-Computer Programming I.

Lab Name	Lab Short Description	Options
Lab I ✎ ✖	C.	✎ ☰ 🗎 ⛁ 🟢 ▤ ⤓ ▶ 🌐 👁
Lab II ✎ ✖	Εισαγωγή στη C.	✎ ☰ 🗎 ⛁ 🟠 ▤ ⤓ ▶ 🌐 👁
Lab III ✎ ✖	Test	✎ ☰ 🗎 ⛁ 🟠 ▤ ⤓ ▶ 🌐 👁
Lab IV ✎ ✖	Pointers	✎ ☰ 🗎 ⛁ 🟠 ▤ ⤓ ▶ 🌐 👁
Lab V ✎ ✖	Pointers II	✎ ☰ 🗎 ⛁ 🟢 ▤ ⤓ ▶ 🌐 👁
Lab VI ✎ ✖	Συναρτήσεις.	✎ ☰ 🗎 ⛁ 🟢 ▤ ⤓ ▶ 🌐 👁
Lab VII ✎ ✖	Αναδρομή	✎ ☰ 🗎 ⛁ 🟢 ▤ ⤓ ▶ 🌐 👁

Figure 34. Labs List

5.4.4.1 Get Submissions

As we said before, professors are able to get submissions on .zip files. On the next figure (Figure 35), we can see how the user gets submissions, simply by pressing the download button.



CE120-Computer Programming I.

Lab Name	Lab Short Description	Options
Lab I		
Lab II		
Lab III		
Lab IV		
Lab V		
Lab VI		
Lab VII		

Άνοιγμα Lab VII.tgz

Επιλέξατε να ανοίξετε:

Lab VII.tgz

που είναι: WinRAR archive (116 KB)

από: http://localhost

Τι να κάνει ο Firefox με αυτό το αρχείο;

● Άνοιγμα με WinRAR archiver (προεπιλογή)

○ Αποθήκευση αρχείου

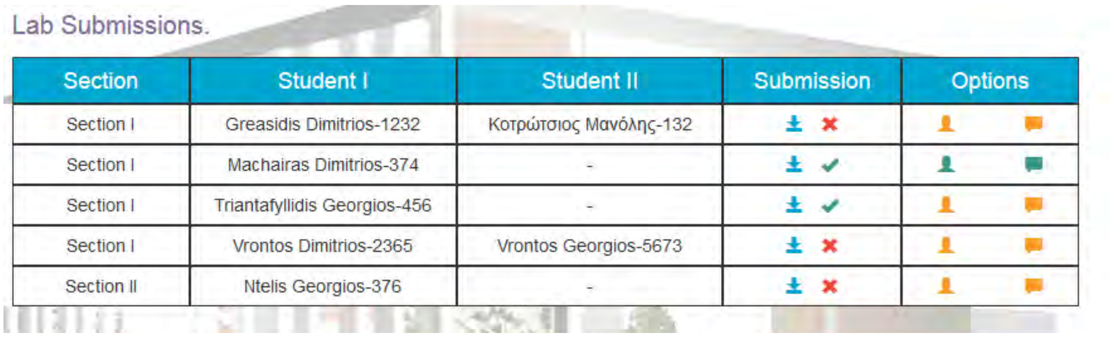
☐ Να γίνεται αυτόματα από εδώ και πέρα για αρχεία αυτού του είδους.

OK Ακύρωση

Figure 35. Get Submissions on .zip File

5.4.4.2 Assign Graders to Submissions

Professor can also view the submissions one by one and set a grader for them. He can achieve it by pressing submissions button. On the next figure (Figure 36), we can see a submission list.

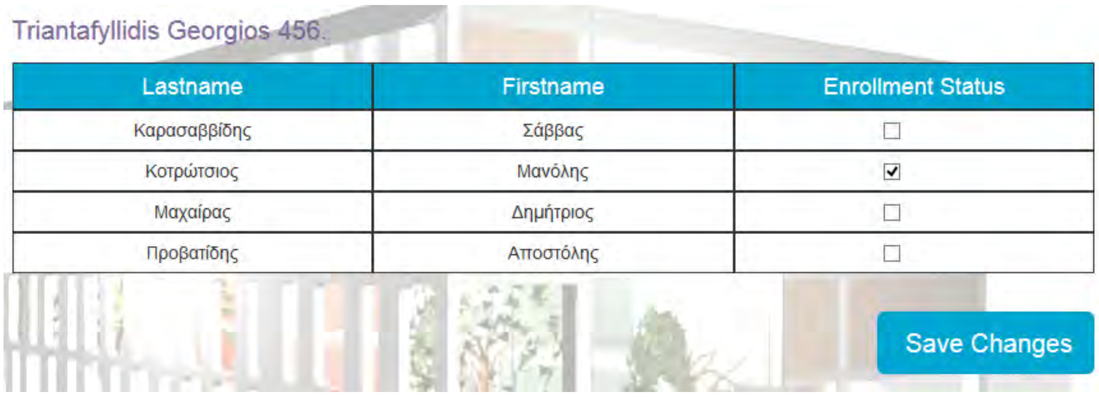


Lab Submissions.

Section	Student I	Student II	Submission	Options
Section I	Greasidis Dimitrios-1232	Κοτρώτσιος Μανόλης-132	✗	
Section I	Machairas Dimitrios-374	-	✓	
Section I	Triantafyllidis Georgios-456	-	✓	
Section I	Vrontos Dimitrios-2365	Vrontos Georgios-5673	✗	
Section II	Ntelis Georgios-376	-	✗	

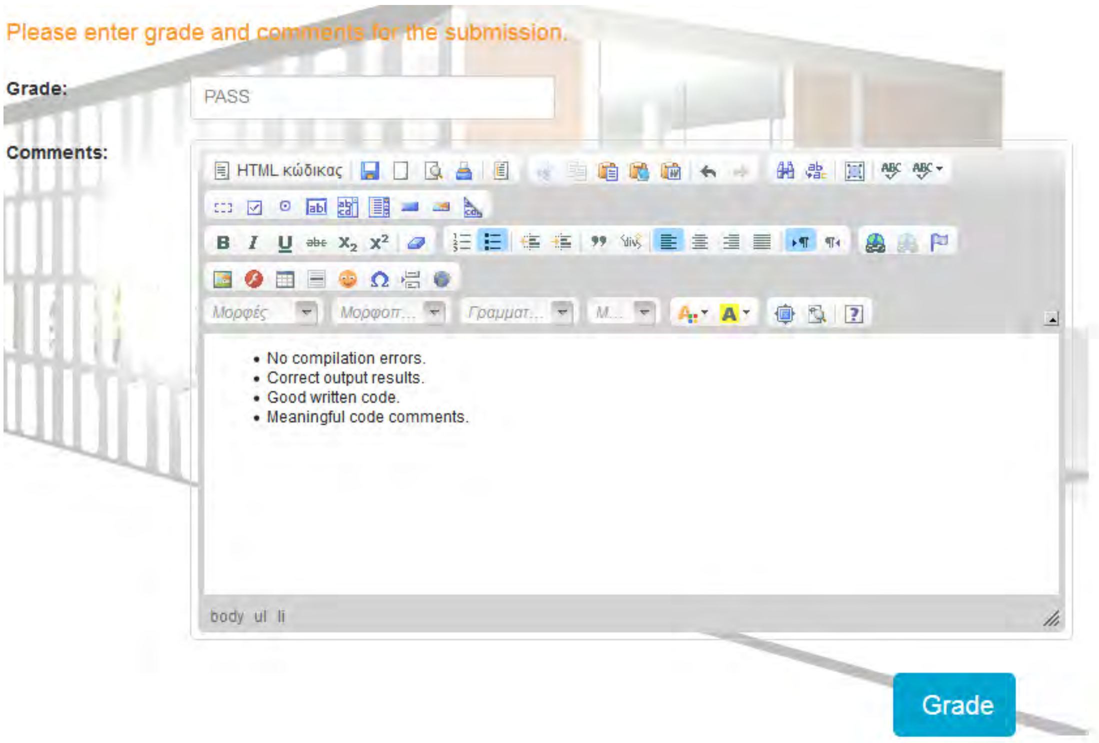
Figure 36. Lab Submissions List

On this list, the professor can see whether the compilation has been successful or not, since the results are being displayed beside the download button. He can also download every submission, as well as set a grader, simply by clicking on the head button. He can also see whether the grader has graded the submission or not. On the next figures (Figure 37 and Figure 38), we can see how a professor chooses a grader, and how he can view grader's grades and comments.


Triantafyllidis Georgios 456.

Lastname	Firstname	Enrollment Status
Καρασαββίδης	Σάββας	☐
Κοτρώτσιος	Μανόλης	☑
Μαχαίρας	Δημήτριος	☐
Προβατίδης	Αποστόλης	☐

Figure 37. Grader Choice


Please enter grade and comments for the submission.

Grade: PASS

Comments:

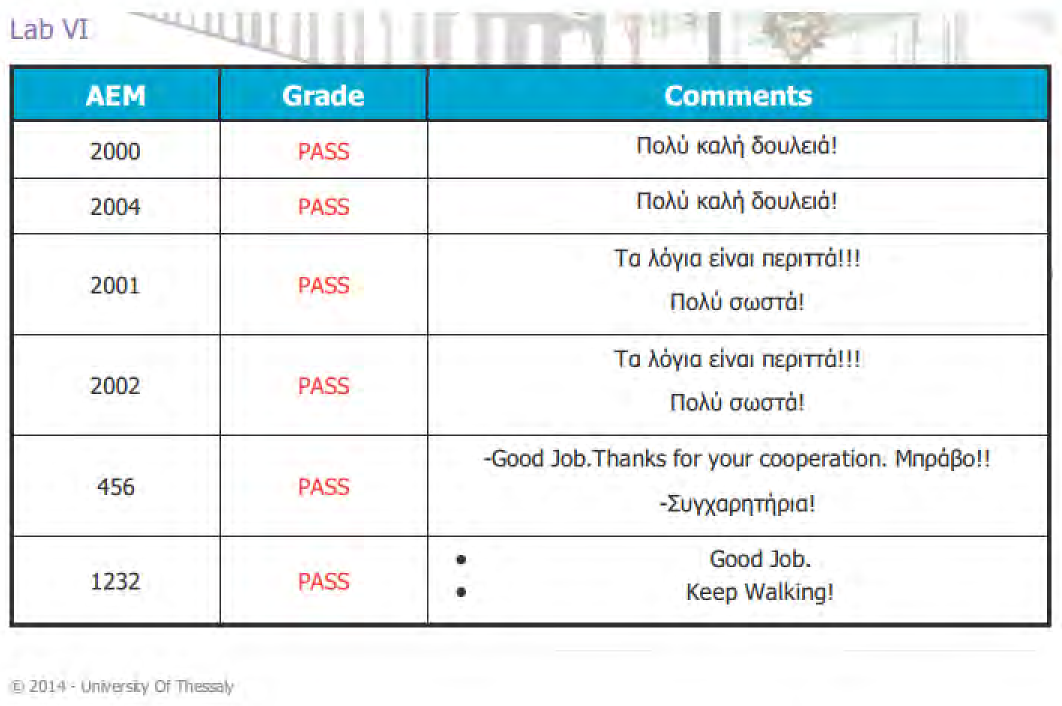
- No compilation errors.
- Correct output results.
- Good written code.
- Meaningful code comments.

Figure 38. Submission Grade and Comments

Last, a professor is also able to edit the submissions grades and comments before publishing them.

5.4.4.3 Publish and Get Grades

Professors must at first, publish lab grades and afterwards, get grades on files. They can publish grades by pressing tick button. When grades are not published, this button is yellow, otherwise is green. On the next figure (Figure 39), we present some lab grades on a .pdf file.



Lab VI

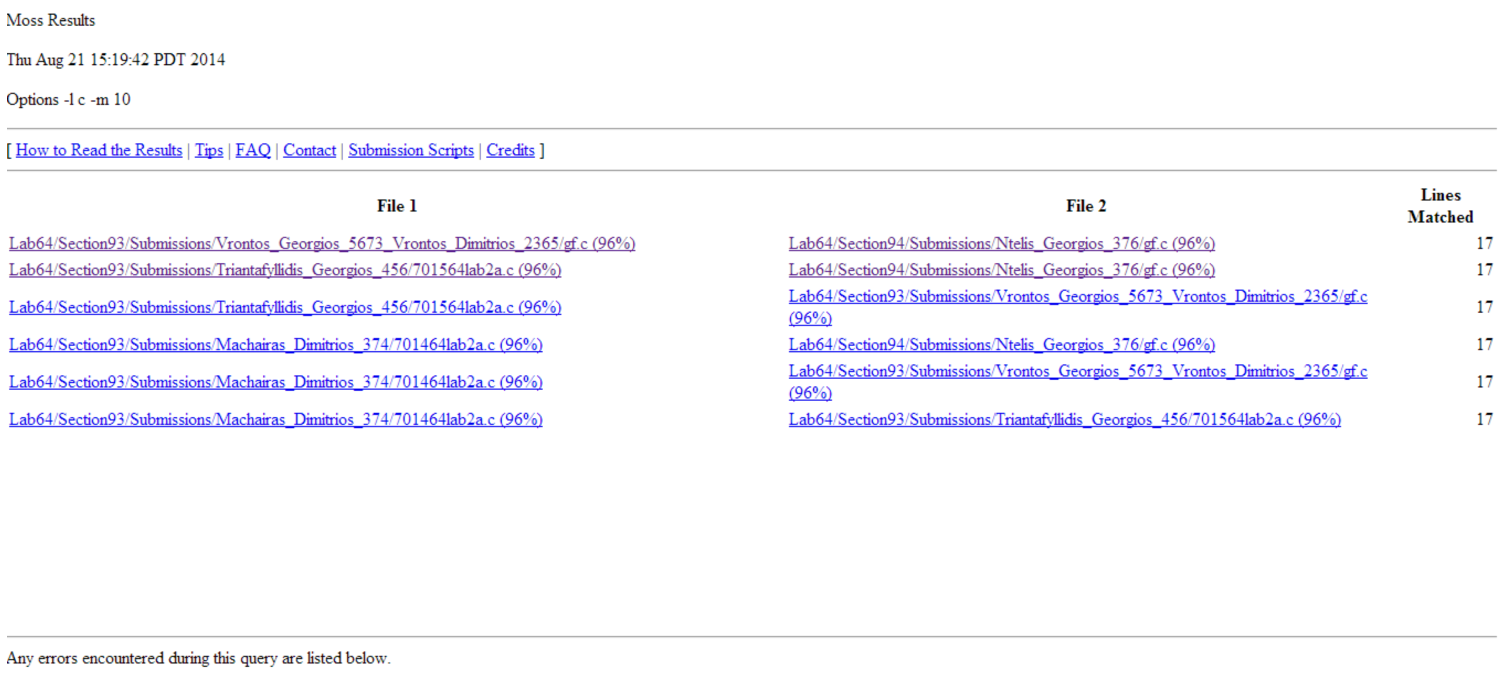
AEM	Grade	Comments
2000	PASS	Πολύ καλή δουλειά!
2004	PASS	Πολύ καλή δουλειά!
2001	PASS	Τα λόγια είναι περιττά!!! Πολύ σωστά!
2002	PASS	Τα λόγια είναι περιττά!!! Πολύ σωστά!
456	PASS	-Good Job.Thanks for your cooperation. Μπράβο!! -Συγχαρητήρια!
1232	PASS	• Good Job. • Keep Walking!

© 2014 - University Of Thessaly

Figure 39. Lab Grades and Comments

5.4.4.4 Moss Plagiarism

As we have analyzed in previous chapters, the professors should also be able to detect any plagiarism, running moss script. After having run it, we save and present the moss results on another browser tab. Moreover, the professors can run moss script simply by pressing the play button, after the corresponding submissions deadline. On the next figure (Figure 40), we can see some moss results of a lab.


Moss Results

Thu Aug 21 15:19:42 PDT 2014

Options -l c -m 10

[How to Read the Results | Tips | FAQ | Contact | Submission Scripts | Credits]

File 1	File 2	Lines Matched
Lab64/Section93/Submissions/Vrontos_Georgios_5673_Vrontos_Dimitrios_2365/gf.c (96%)	Lab64/Section94/Submissions/Ntelis_Georgios_376/gf.c (96%)	17
Lab64/Section93/Submissions/Triantafyllidis_Georgios_456/701564lab2a.c (96%)	Lab64/Section94/Submissions/Ntelis_Georgios_376/gf.c (96%)	17
Lab64/Section93/Submissions/Triantafyllidis_Georgios_456/701564lab2a.c (96%)	Lab64/Section93/Submissions/Vrontos_Georgios_5673_Vrontos_Dimitrios_2365/gf.c (96%)	17
Lab64/Section93/Submissions/Machairas_Dimitrios_374/701464lab2a.c (96%)	Lab64/Section94/Submissions/Ntelis_Georgios_376/gf.c (96%)	17
Lab64/Section93/Submissions/Machairas_Dimitrios_374/701464lab2a.c (96%)	Lab64/Section93/Submissions/Vrontos_Georgios_5673_Vrontos_Dimitrios_2365/gf.c (96%)	17
Lab64/Section93/Submissions/Machairas_Dimitrios_374/701464lab2a.c (96%)	Lab64/Section93/Submissions/Triantafyllidis_Georgios_456/701564lab2a.c (96%)	17

Any errors encountered during this query are listed below.

Figure 40. Moss Results

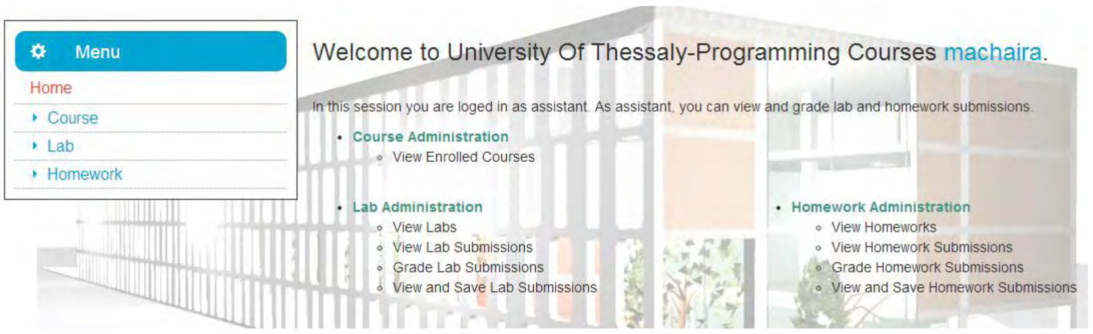
Similarly, professors can perform exactly the same actions for homeworks and get the corresponding results. On the next section, we will describe how graders can administer our platform.

5.5 Assistant's Panel

On this section we will present the assistant's functionality. We start with the presentation of an assistant's home page.

5.5.1 Assistant's Home Page

After an assistant logs in on the web application, he will see the assistant's home page, which is being displayed on the next figure (Figure 41).



Welcome to University Of Thessaly-Programming Courses machaira.

In this session you are loged in as assistant. As assistant, you can view and grade lab and homework submissions.

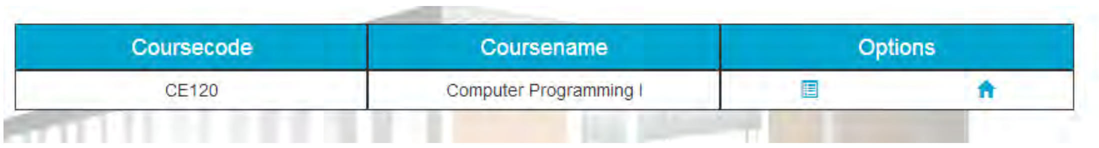
- Course Administration
  - View Enrolled Courses
- Lab Administration
  - View Labs
  - View Lab Submissions
  - Grade Lab Submissions
  - View and Save Lab Submissions
- Homework Administration
  - View Homeworks
  - View Homework Submissions
  - Grade Homework Submissions
  - View and Save Homework Submissions

Figure 41. Assistant's Home Page

As we can see, an assistant can view courses, as well as view and grade lab and homework submissions.

5.5.2 Enrolled Courses

Only the course professors can enroll an assistant in a course or unenroll him. On the next figure (Figure 42), we can see an assistant course list.



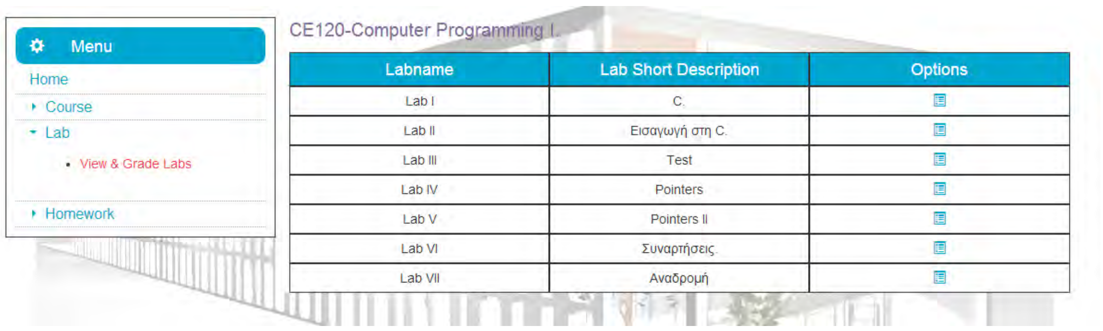
Coursecode	Coursename	Options
CE120	Computer Programming I	

Figure 42. Assistant's Course List

In this list, an assistant can view the lab or homework submissions, which he must grade.

5.5.3 Lab and Homework Administration

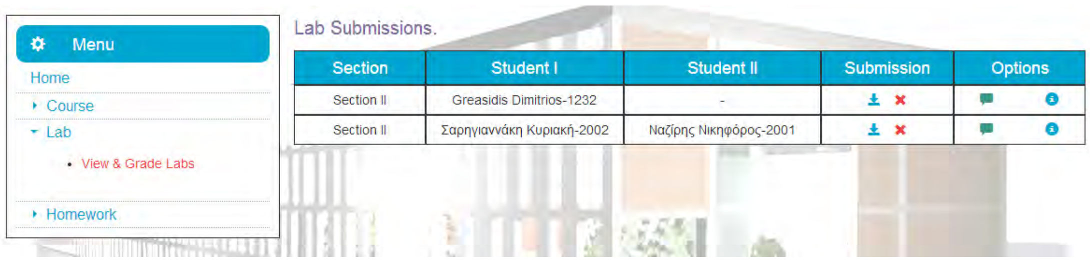
An assistant is able to view only the lists of labs and homeworks of the courses in which he is enrolled. Due to the fact that the lab administration is the same as the homework administration, we have reached the decision to present them together. On the next figure (Figure 43), we can see the list of course labs and on figure (Figure 44) the list of the submissions, that the assistant has to grade. Assistants are also able to see whether the submission has compilation errors or not, to download submission files and to see the instruction file, which the course professors have certainly uploaded while creating the corresponding lab.



CE120-Computer Programming I

Labname	Lab Short Description	Options
Lab I	C	▤
Lab II	Εισαγωγή στη C	▤
Lab III	Test	▤
Lab IV	Pointers	▤
Lab V	Pointers II	▤
Lab VI	Συναρτήσεις	▤
Lab VII	Αναδρομή	▤

Figure 43. Assistant's Labs List



Lab Submissions.

Section	Student I	Student II	Submission	Options
Section II	Greasidis Dimitrios-1232	-	⤓ ✖	💬 ⓘ
Section II	Σαρηγιαννάκη Κυριακή-2002	Ναζίρης Νικηφόρος-2001	⤓ ✖	💬 ⓘ

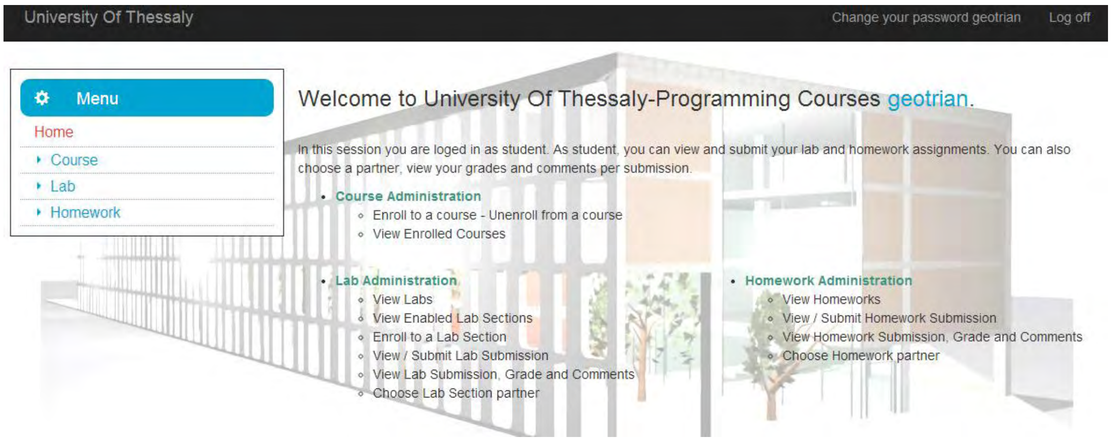
Figure 44. Assistant's Labs Submissions

5.6 Student's Panel

On this section we will present the student's functionality. We start with the student's home page.

5.6.1 Student's Home Page

After a student logs in on the web application, he will see the student's home page, which is being displayed on the next figure (Figure 45).


University Of Thessaly

Change your password geotrian Log off

Welcome to University Of Thessaly-Programming Courses geotrian.

In this session you are loged in as student. As student, you can view and submit your lab and homework assignments. You can also choose a partner, view your grades and comments per submission.

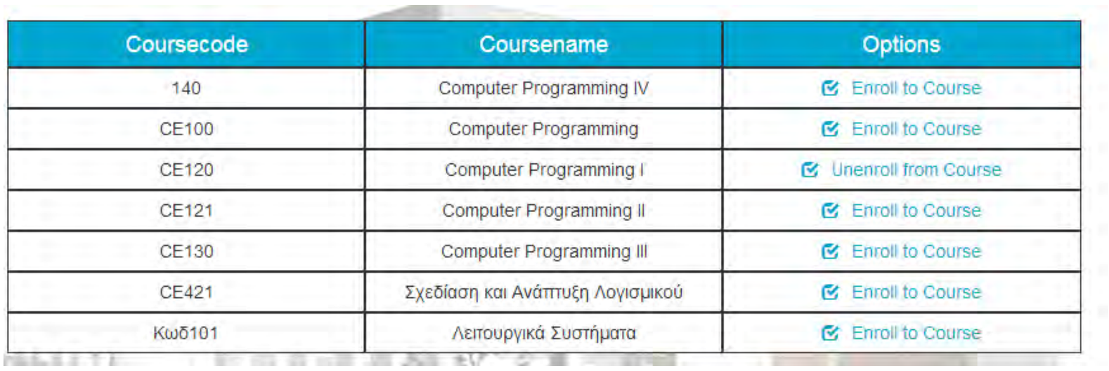
- Course Administration
  - Enroll to a course - Unenroll from a course
  - View Enrolled Courses
- Lab Administration
  - View Labs
  - View Enabled Lab Sections
  - Enroll to a Lab Section
  - View / Submit Lab Submission
  - View Lab Submission, Grade and Comments
  - Choose Lab Section partner
- Homework Administration
  - View Homeworks
  - View / Submit Homework Submission
  - View Homework Submission, Grade and Comments
  - Choose Homework partner

Figure 45. Student's Home Page

We can easily notice that a student can view courses and enroll to them, as well as view and submit lab and homework assignments.

5.6.2 Course Enrollment

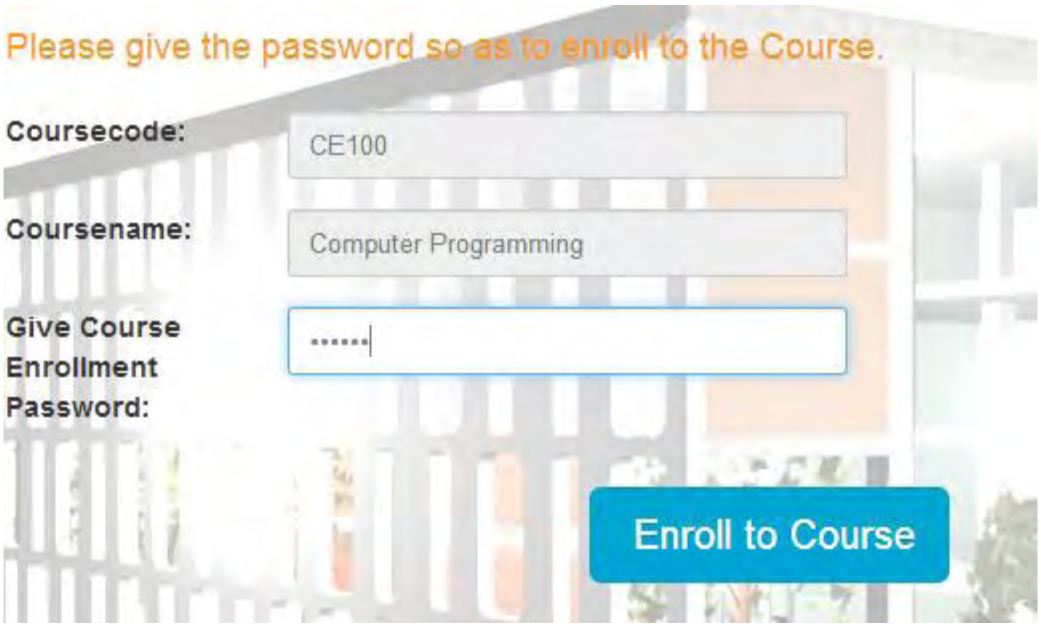
Students are able to enroll to courses and unenroll from them. In order for a student to enroll to a course, he has to know the course password, which is provided by course professors. On the next figure (Figure 46), we can see the list of the courses, in which a student can enroll or unenroll from.



Coursecode	Coursename	Options
140	Computer Programming IV	Enroll to Course
CE100	Computer Programming	Enroll to Course
CE120	Computer Programming I	Unenroll from Course
CE121	Computer Programming II	Enroll to Course
CE130	Computer Programming III	Enroll to Course
CE421	Σχεδίαση και Ανάπτυξη Λογισμικού	Enroll to Course
Κωδ101	Λειτουργικά Συστήματα	Enroll to Course

Figure 46. Student's Course List

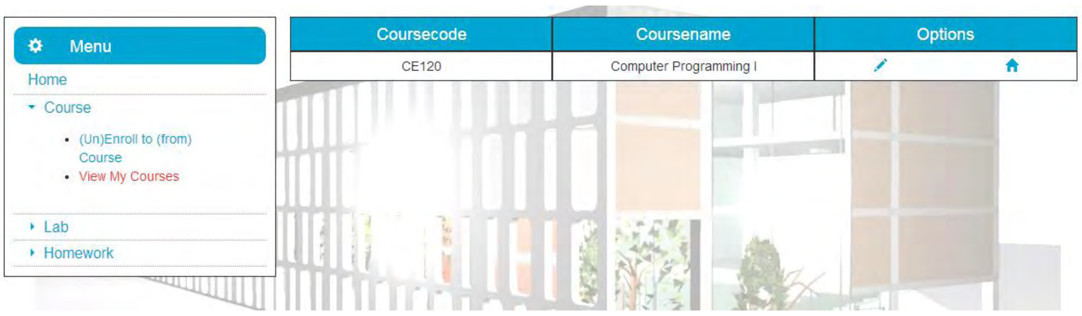
If a student wants to enroll to a course, he should click on the option "Enroll to Course" for the corresponding course. Then he must insert the course password, as we can observe on next figure (Figure 47).


Please give the password so as to enroll to the Course.

Coursecode:	CE100
Coursename:	Computer Programming
Give Course Enrollment Password:	••••••

Figure 47. Course Enrollment

If the student gives a wrong password, an error message will be immediately displayed on his screen. Otherwise, he will be redirected to his *View My Enrolled Courses* page. We can see this page on the next figure (Figure 48).

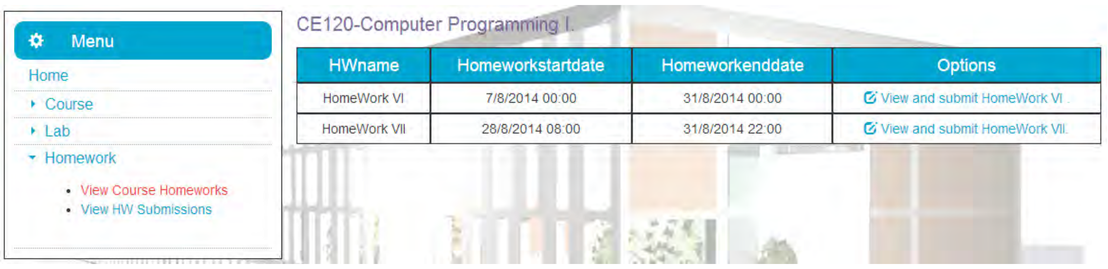


Coursecode	Coursename	Options
CE120	Computer Programming I	

Figure 48. View My Enrolled Courses Page

5.6.3 Assignment Submission

After enrolling in a course, students are able to view the course labs and homeworks. On the next figure (Figure 49), a list of homeworks is being displayed.



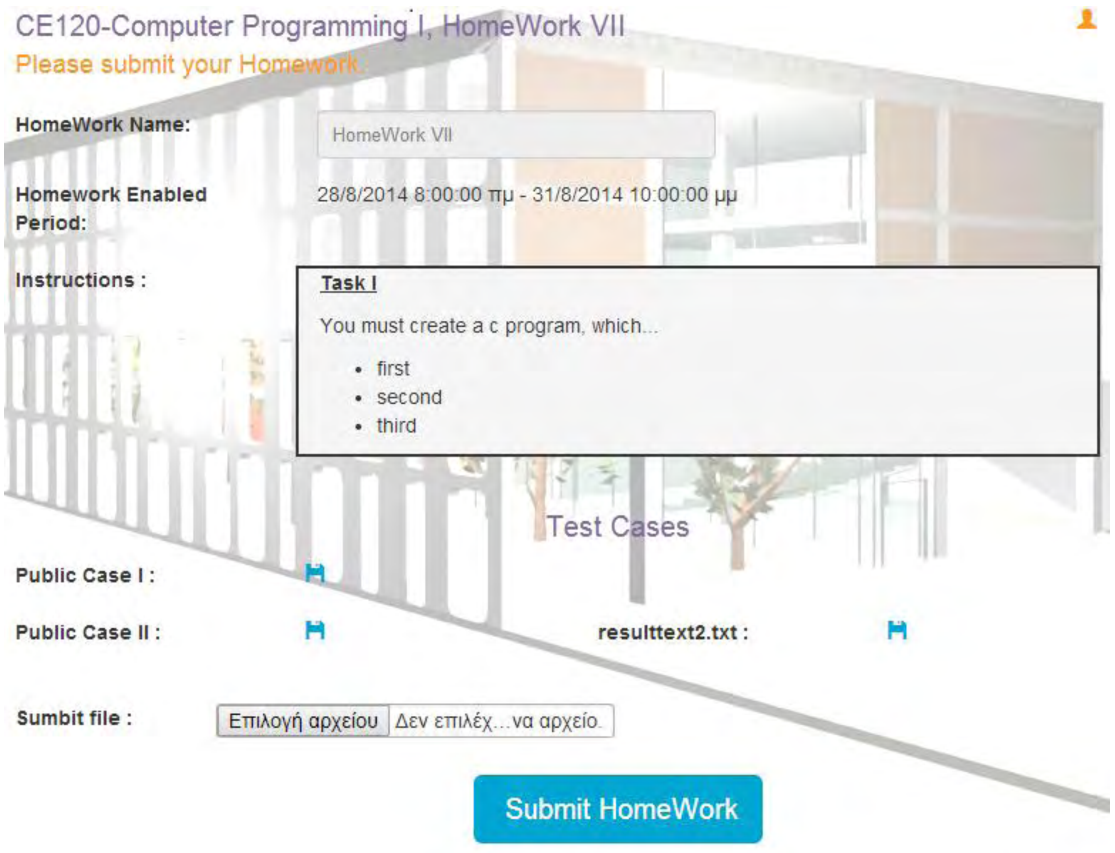
CE120-Computer Programming I

HWname	Homeworkstartdate	Homeworkenddate	Options
HomeWork VI	7/8/2014 00:00	31/8/2014 00:00	View and submit HomeWork VI
HomeWork VII	28/8/2014 08:00	31/8/2014 22:00	View and submit HomeWork VII

Figure 49. List of Homeworks

The students can choose and view one lab or homework from the previous list. It is vital that they enroll to an available section, when it comes to a lab . Even if they simply want to view the available homeworks and the corresponding task, these are options which must be enabled by the professor. Afterwards, if they have to work in pairs, they have to choose a partner, solve and submit their tasks, and view the compilation results of their assignment. Additionally, they certainly have the ability to view the test cases' results.

On the next figure (Figure 50), we can see how a lab or homework task looks like and how a student can get information about its task.


CE120-Computer Programming I, HomeWork VII

Please submit your Homework.

HomeWork Name: HomeWork VII

Homework Enabled Period: 28/8/2014 8:00:00 πμ - 31/8/2014 10:00:00 μμ

Instructions :

Task I

You must create a c program, which...

- first
- second
- third

Test Cases

Public Case I :

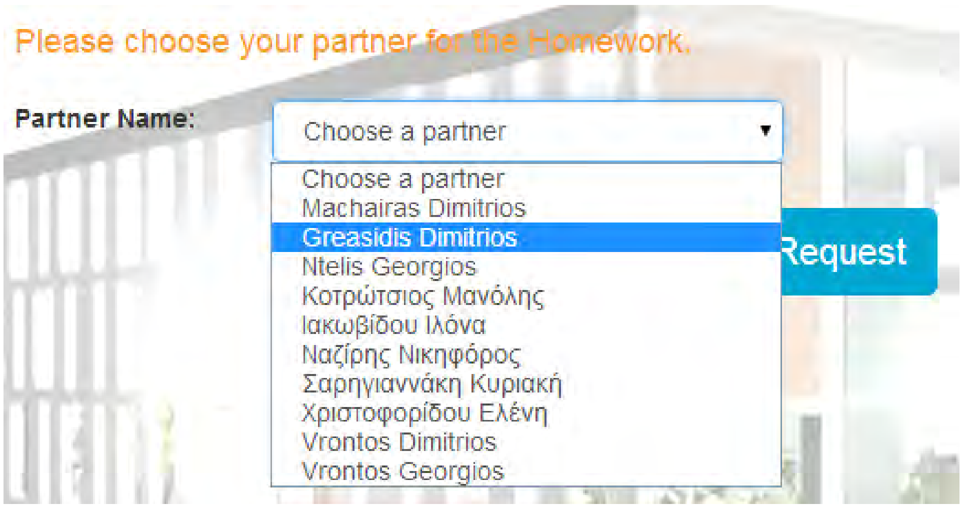
Public Case II : resulttext2.txt :

Sumbit file : Επιλογή αρχείου Δεν επιλέχ...να αρχείο.

Submit HomeWork

Figure 50. Homework Task

As we can see, students can see not only the tasks, but also the available public test cases. By clicking on the face button, they are able to see all the available partners and choose one. On the next figure (Figure 51), we can see how a student can send a request to another enrolled student in order for them to become partners. When it comes to lab sections, both of them, should be enrolled in the same section.


Please choose your partner for the Homework.

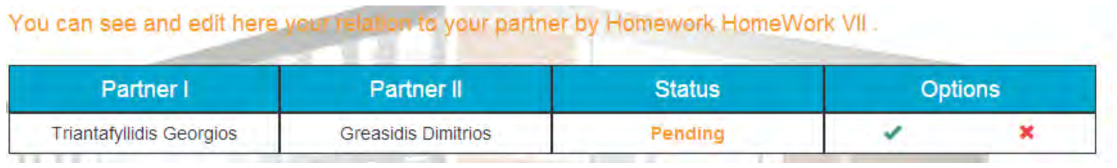
Partner Name: Choose a partner

Choose a partner
Machairas Dimitrios
Greasidis Dimitrios
Ntelis Georgios
Κοτρώτσιος Μανόλης
Ιακωβίδου Ιλόνα
Ναζίρης Νικηφόρος
Σαρηγιαννάκη Κυριακή
Χριστοφορίδου Ελένη
Vrontos Dimitrios
Vrontos Georgios

Request

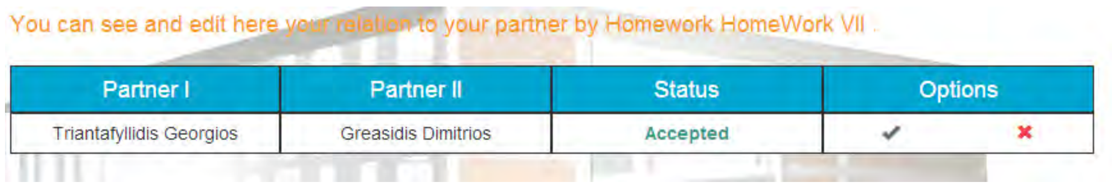
Figure 51. Partner's Choice

The student who receives a request can accept or reject it. We actually have to notice that there are 3 available status for this case: *Accepted*, *Rejected* and *Pending*. The Accepted status means that the latter of the two has accepted the request that the former has sent and that they have become partners. Moreover, the Rejected status means that the students have not become partners, since the receiver has rejected the request. Furthermore, the Pending status means that the student has not yet accepted or rejected the request. On the next figures (Figure 52 and Figure 53), we can see a Pending status and how the status changes when the student accepts the request.


You can see and edit here your relation to your partner by Homework HomeWork VII .

Partner I	Partner II	Status	Options	
Triantafyllidis Georgios	Greasidis Dimitrios	Pending	✓	✗

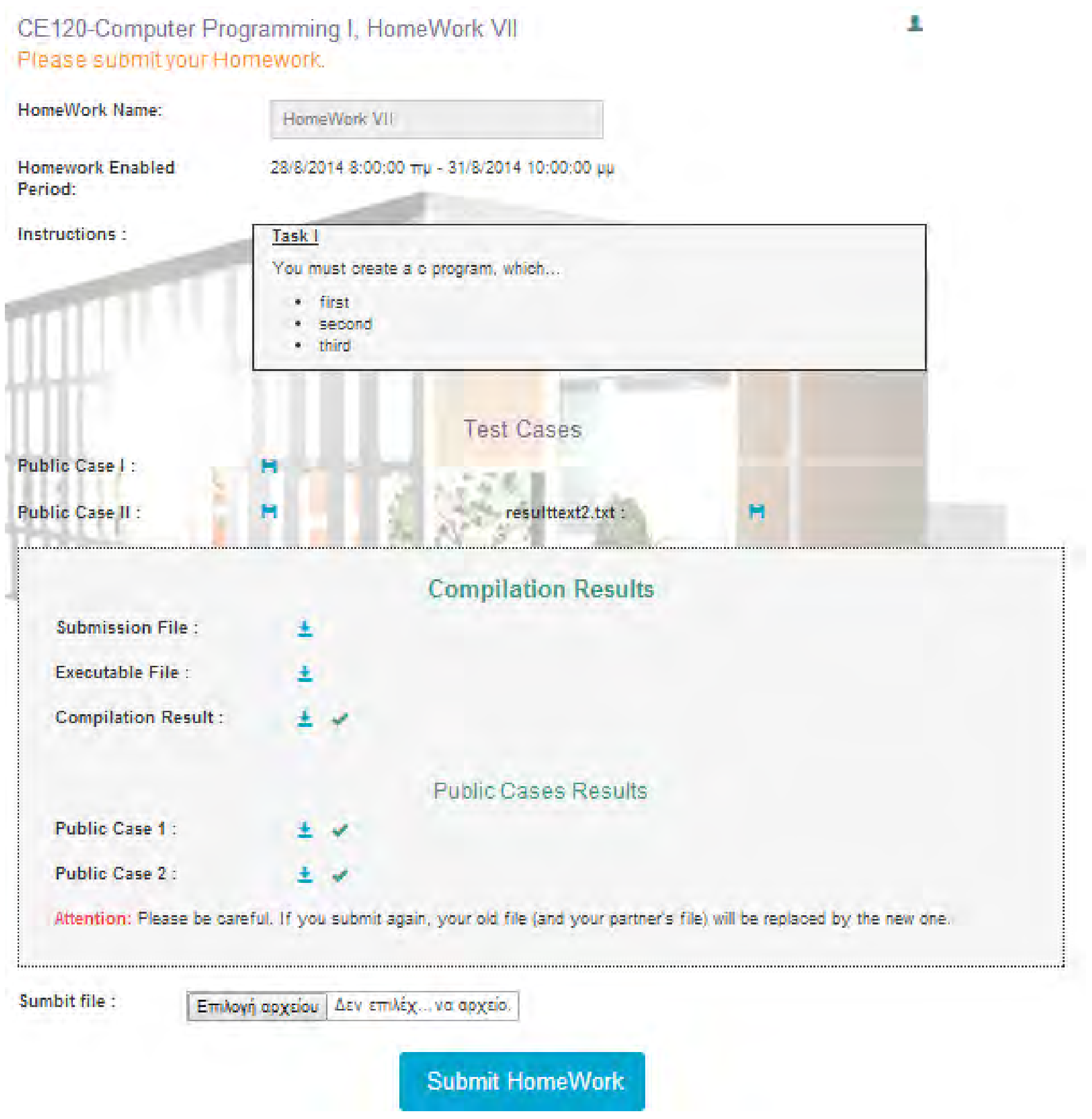
Figure 52. Pending Status


You can see and edit here your relation to your partner by Homework HomeWork VII .

Partner I	Partner II	Status	Options	
Triantafyllidis Georgios	Greasidis Dimitrios	Accepted	✓	✗

Figure 53. Accepted Status

After accepting the request, one of the two partners must submit their assignment. Moreover, the students can replace the task that they have submitted simply by submitting a new one on the deadline. Last but not least, each time that they submit their assignment, the old one is deleted. On the next figure (Figure 54), we can see a submission page with results, after the submission of an assignment.


CE120-Computer Programming I, HomeWork VII
Please submit your Homework.

HomeWork Name:	HomeWork VII
Homework Enabled Period:	28/8/2014 8:00:00 πμ - 31/8/2014 10:00:00 μμ
Instructions :	Task I You must create a c program, which... • first • second • third

Test Cases

Public Case I :
Public Case II : resulttext2.txt :

Compilation Results

Submission File :
Executable File :
Compilation Result : ✓

Public Cases Results

Public Case 1 : ✓
Public Case 2 : ✓

Attention: Please be careful. If you submit again, your old file (and your partner's file) will be replaced by the new one.

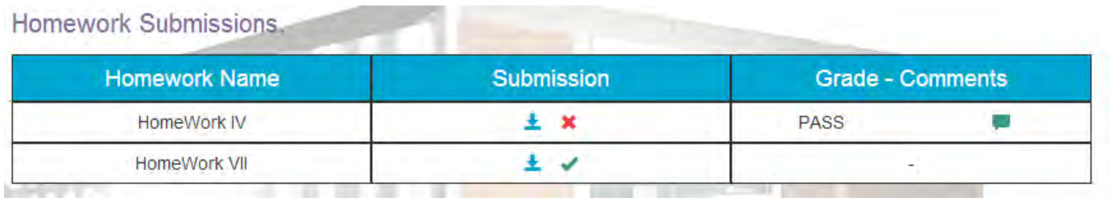
Sumbit file : Επιλογή αρχείου Δεν επιλέχ...να αρχείο.

Figure 54. Compilation Results

After submitting their assignments, the students are able to see and download the results of their submission and the results of the test cases.

5.6.4 Submission Grade and Comments

When the deadline is over, the graders grade and comment on the students submissions. Then the professors publish grades and as a result, the students can view their grades and comments. Due to the fact that they have no access to the submission page, since the deadline is already over, they can see the submission file and the results on another page, which is called *Homework* or *Lab Submissions*. On the next figure (Figure 55), we view how students can see their grades. By clicking on the comments button, they can also see the submission comments.



Homework Submissions

Homework Name	Submission	Grade - Comments
HomeWork IV	✗	PASS
HomeWork VII	✓	-

Figure 55. Submissions List of Homeworks

6 Related Work

In this chapter, we present some indicative works, which are related to ours. We divide these works into 4 main categories: *e-learning platforms*, *plagiarism detection systems online, compilation tools* and *automated testing systems*.

6.1 E-learning Platforms

Universities often use e-learning platforms, which are complete course management systems that support asynchronous eLearning services using a simple web browser. The goal of such platforms is the incorporation and the constructive use of the Internet and web technologies in the teaching and learning process. They support the electronic management, storage and presentation of teaching material. Such systems operate independently of the spatial and time limiting factors of conventional teaching and as a result they create the necessary conditions for a dynamic teaching environment. [15]

Despite the fact that such applications provide a variety of functionality, they are not customized for programming assignments. On such platforms, for example, professors are able to create new courses and assign new homeworks or laboratories to their students, but they cannot upload programming tasks, whose submissions will be compiled online. As a result, when it comes to programming assignments, in order for professors to test them, they have to compile them one by one. Furthermore, these e-learning platforms do not allow the professors to view any plagiarism reports concerning similar or same submissions. Consequently, professors must collect all the submissions and save them in a local folder, which they then have to check for any same or similar code parts, by running, for example, moss script. Such platforms are, for example, *e-class* [15] and *Blackboard* [16]. On both of them, professors can grade students' submissions and write some feedback about them, which the students can directly view. Even though the professors have the ability to easily grade and comment, these platforms do not provide them the option of choosing for each submission a grader, who would grade and comment the submission. As a consequence, they have to grade all submissions by themselves.

6.2 Plagiarism Detection Systems

A great number of Plagiarism Detection tools are available online. Some of them are free but the most of them are commercial. These systems are able to detect plagiarism among uploaded files, as well as between uploaded files and the web. The most of them provide a plagiarism percentage number.

First of all, *Chimpsky* [17], is a system, which can detect either plagiarism between text documents (DOC, DOCX, WPD, RTF, ODT e.t.c.) and content which is available online on the web, or among uploaded documents, which can be not only documents, but also programs in C, Java, Pascal, Modula-2, Miranda, and Lisp. This tool uses *SIM*, which is a similarity tester software for uploaded documents, and Google ajax api, when it comes to content which is available online on the web. This tool is free and a user can use it after creating a free account. The user is also able to upload a template file, whose content will not be checked for plagiarism. Although Chimpsky allows the users to upload all files as a .zip file, some browsers do not support it. Furthermore, this tool does not support files, that contain more than 5000 words. We should also mention that the number of tasks for each user is also limited.

Another plagiarism detection system is *seesources.com* [18]. This tool detects plagiarism between text documents and content which is available online on the web. Users are able to use it for free, and in addition, the program does not need to be installed in their computer. The maximum number of words per document is 1000 and the size of the document must be less than 300MB. The users can only upload one document at a time, and if it is a bit big, the functionality of the system is too slow. *The Plagiarism Checker* has a functionality quite similar to seesources.com.

PlagTracker [19] is another significant plagiarism system. It is actually a commercial tool which has 4 available user roles: *Students*, *Teachers*, *Publishers* and Site *Owners*. This system is quite fast and easy to use, but a user can use it only after obtaining a license. First, students are able to detect cases of plagiarism, so as for their tasks to be safe. They can also check if there are any syntax and grammar mistakes so they can correct them. Moreover, teachers are able to find out whether there are any problems concerning their students' tasks or not while publishers can check if their files are ready and safe before publishing them. Additionally, the site owners can check if the written material on their web site is unique or if somebody have copied them.

Another very useful commercial tool is *Turnitin* [20]. This tool not only manages submissions but also evaluates the students' papers online. It also provides its users the ability to detect plagiarism. What is more, users are also able to get an originality report, that is being immediately generated, and to grade online. As we have already mentioned, due to the fact that it comes to a commercial tool, a user must always have an account in order to use it.

Last but not least, *Ephorus* [21] is another commercial web system for plagiarism detection, that requires no installation. This system allows its user to upload and check a great number of documents at the same time. Afterwards, it sends a report to the user via email. Ephorus can rapidly detect any similarities between uploaded files and content which is available online on the web or works previously submitted on the system. It can also detect similarities between the user's files and files that other enrolled users have submitted. This tool is also suitable for teachers who want to check their students' submissions for plagiarism however they have no free time to check them one by one. Additionally, Ephorus can also be used for examination boards, directors and ICT workers.

6.3 Online Compilation Tools

Apart from e-learning platforms, there is a also plenty of tools, which provide the ability to their users to check whether their program has any compilation warnings and errors or not. Needless to say on these tools, users have to write their code on an editor and then press the compilation button in order to view whether there are any compilation warnings and errors or not. Such tools, for example, are compileonline.com [22], onlinecompiler.net [23], ideone.com [24] and codepad [25]. Most of them provide their users the ability to choose from a wide variety of programming languages, the one that they used while programming. As a result, the program will be compiled according to the data of the programming language. It should be mentioned that most of these systems enable their users to write some input test cases, as well as to get the source code file and the executable one.

6.4 Automated Testing Systems

Online automated testing systems check whether the results between the output files of the test cases and the output files of the user test cases results are the desirable or not. One significant automated testing system is ATS [26]. ATS is a Python-based tool which is used for automating the running of tests of an application. This tool runs the available tests, and afterwards, it creates a log file, which describes the corresponding results. Moreover, this system is free and a user is able to download and use it from its web page.

7 Conclusions

In this thesis we have tried to develop an interactive web platform, which can support programming assignments (in C programming language) in university courses. As the number of university students is rapidly increasing, the whole procedure of the submissions management is getting much more difficult. The already existing old-fashioned tools such as email, are not so efficient any more. Moreover, since it is a rather common phenomenon that many students try to cheat and copy from each other, it is vital that professors use a kind of plagiarism system, which will detect the same code parts. This procedure is very difficult when it comes to a large number of submissions.

We have spent a lot of time using moodle [7], however, we have found out that this software was not actually helpful as it was quite away from the customization of our platform. As a result, and since our first try failed, we had to start the web application development from scratch. This means, that we had to be much more cautious, while we were searching for an appropriate development software for our application. We finally chose the MVC 5 architecture [2] [1]. MVC 5 has actually been proved much more efficient and helped us develop our platform, as it has much more benefits in comparison to moodle software.

So, to cut a long story short, we managed to develop a web platform, where there are 4 kinds of users: *Administrator*, *Professor*, *Assistant* or *Grader* and *Student*. Each kind of user can administer courses, labs and homeworks in his own way, apart from the administrator, who can also administer users. In this version, we have developed the platform so as to run only for C programming language. Moreover, by using our platform, not only professors but also graders will have less workload concerning submissions management, as they can check their students submissions much easier. Furthermore, professors can also view any cases of plagiarism by pressing only one button, and without even organizing the students submissions in folders. Additionally, graders can see only their assigned submissions without wasting their time searching among all submissions. Students can view compilation and test cases results right after submitting their assignments and as a result, they are able to check whether they have sent the right submission or not. Last but not least, a future plan is to make this platform also run and for other programming languages, such as C++, Java or C#, which a professor will select when he creates a new course.

Bibliography

[1] "A brief history of Asp.Net MVC framework," Dot Net Tricks, [Online]. Available: http://www.dotnet-tricks.com/Tutorial/mvc/XWX7210713-A-brief-history-of-Asp.Net-MVC-framework.html.

[2] A. Leff and J. Rayfield, "Web-application development using the Model/View/Controller design pattern," in *Proceedings of the EDOC '01. Fifth IEEE International in Enterprise Distributed Object Computing Conference*, Seattle, WA, 2001.

[3] "Bootstrap," Mark Otto (mdo) & Jacov (fat), [Online]. Available: http://getbootstrap.com/.

[4] F. Ju, L. Guoliang and Z. Lizhu, "Interactive SQL query suggestion: Making databases user-friendly," in *Proceedings of the 2011 IEEE 27th International Conference in Data Engineering (ICDE)*, Hannover, 2011.

[5] A. Yoshizaki and N. Shimizu, "Development of educational material for real-time operating system," in *Proceedings of the 2012 International Conference in Computing, Electronics and Electrical Technologies (ICCEET)*, Kumaracoil, 2012.

[6] "A System for Detecting Software Plagiarism," Moss - Stanford University, [Online]. Available: http://theory.stanford.edu/~aiken/moss/.

[7] "About Moodle," Moodle, [Online]. Available: https://docs.moodle.org/27/en/About_Moodle.

[8] "Benefits of MVC pattern," CareerRide.COM, [Online]. Available: http://www.careerride.com/MVC-benefits.aspx.

[9] S. Schleimer, D. Wilkerson and A. Aiken, "Winnowing: Local Algorithms for Document Fingerprinting," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, San Diego, 2003.

[10] "Microsoft SQL Server Management Studio Express," Microsoft Download Center, [Online]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=8961.

[11] "How to install SQL Server 2012 Management Studio Express on Windows 7 PC and Create a Local SQL Server Database," Perficient, [Online]. Available: http://blogs.perficient.com/ibm/2012/11/02/how-to-install-sql-server-2012-

management-studio-express-on-windows-7-pc-and-create-a-local-sql-server-database/.

[12] "Welcome to Visual Studio 2013.," Microsoft Developer Network, [Online]. Available: http://msdn.microsoft.com/en-us/library/dd831853.aspx.

[13] "Deploy a Secure ASP.NET MVC 5 app with Membership, OAuth, and SQL Database to an Azure Website.," Microsoft Azure, [Online]. Available: http://azure.microsoft.com/en-us/documentation/articles/web-sites-dotnet-deploy-aspnet-mvc-app-membership-oauth-sql-database/.

[14] R. Pranav, R. Anderson, T. Dykstra and J. Galloway, "Introduction to ASP.NET Identity.," 2013.

[15] "Open eClass - Asynchronous eLearning Platform," [Online]. Available: http://eclass.chios.aegean.gr/.

[16] "Blackboard," Blackboard, [Online]. Available: http://uki.blackboard.com/sites/international/globalmaster/.

[17] M. Palmer, "Chimpsky plagiarism detection," University of Waterloo, [Online]. Available: http://chimpsky.uwaterloo.ca/.

[18] "Seesources.com," PlagScan, [Online]. Available: http://www.plagscan.com/seesources/.

[19] "PLAG TRACKER," [Online]. Available: http://www.plagtracker.com/.

[20] "Turnitin," [Online]. Available: http://turnitin.com/.

[21] "Ephorus," [Online]. Available: https://www.ephorus.com/.

[22] "compile online com," [Online]. Available: http://compileonline.com/.

[23] "Online Compiler .net," [Online]. Available: http://www.onlinecompiler.net/.

[24] "ideone.com," [Online]. Available: http://ideone.com/.

[25] "codepad," [Online]. Available: http://codepad.org/.

[26] "Automated Testing System (ATS)," Google Project Hosting, [Online]. Available: https://code.google.com/p/ats/.

Appendices

Appendix A

On this appendix, we present and describe a table (Table 7) which presents a view.

```
@model UniversityOfThessaly.Models.Grade

@{
    ViewBag.Title = "Add Grade";
}


<div class="row">
    <div class="col-sm-3">
        @Html.Partial("_Menus")


    </div>
    <div class="col-sm-9">
        @if (string.IsNullOrEmpty(ViewBag.Success))
        {
            <span class="myYellow fs18 m">  Please enter grade and comments for the
submission.</span>
            using (Html.BeginForm("AddGrade", "labs", null, FormMethod.Post, new {
id = "AddGrade" }))
            {
                @Html.Hidden("gradeid", (int)ViewBag.Gradeid)
                @Html.Hidden("labid", (int)ViewBag.LabID)

                <div class="row form-group marTop20">
                    <div class="col-sm-12 row form-group">

                        <label for="LabName" class="col-md-2 control-
label">Grade:</label>

                        <div class="col-md-10">
                            @Html.TextBoxFor(m => m.GradeValue, new { style =
"width:100%", @class = "GradeValue form-control" })
                        </div>

                    </div>

                    <div class="col-sm-12 row form-group">

                        <label for="Instructions" class="col-md-2 control-
label">Comments:</label>

                        <div class="col-md-10">
                            @Html.TextAreaFor(m => m.Comments, new { style =
"width:100%", @class = "Comments form-control ckeditor", rows = 10 })
                        </div>

                    </div>

                    <div class="col-sm-11 marTop10 text-right">
                        <button type="submit" class="btn btn-blue btn-lg clearfix">
                            <div class="fs20 text-center"><p class="fs20 marBottom0
marLeft5">Grade</p></div>
                        </button>
                    </div>
                </div>
                <div class="row">
                    <div class="col-sm-12">
                        @if (!string.IsNullOrEmpty(ViewBag.Error))
                        {
                            <span class="myRed text-left">@ViewBag.Error</span>
                        }
                    </div>
                </div>


            }
        }
        else
        {
            <div class="row">
                <div class="col-sm-12">
                    <span class="myGreen">@ViewBag.Success <br /></span>
                    @if (User.IsInRole("ASSISTANT"))
                    {
                    <span class="myGray">Please click <a
href="@Url.Action("viewmylabassigments", "labs", new { labid = (int)ViewBag.LabID
})"> here </a> to see your assignments.</span>
                    }
                    @if (User.IsInRole("PROFESSOR"))
                    {
                        <span class="myGray">Please click <a
href="@Url.Action("viewmylabsubmissions", "labs", new { labid = (int)ViewBag.LabID
})"> here </a> to see your submissions.</span>

                    }
                </div>
            </div>
        }
    </div>
</div>
```

Table 7. Add Grade View

On the previous table, we can see the *View* code, which is, in fact, a combination between Html and C# programming language. Needless to say, we declare on the first line, the model that we get as returning parameter from the controller. We also use some C# features, for example, @Hmtl.Partial() function, which is used so as to call a Partial View. Such views, contain code which is used on many places on the web platform, and they help, in order not to repeat the same code on different places on the web platform.

Appendix B

On the next figure (Figure 56), we present the whole database schema, which we have described on 4.1.1 Database Schema. Needless to say, we have created this database schema, using SQL Server Management Studio 2012.

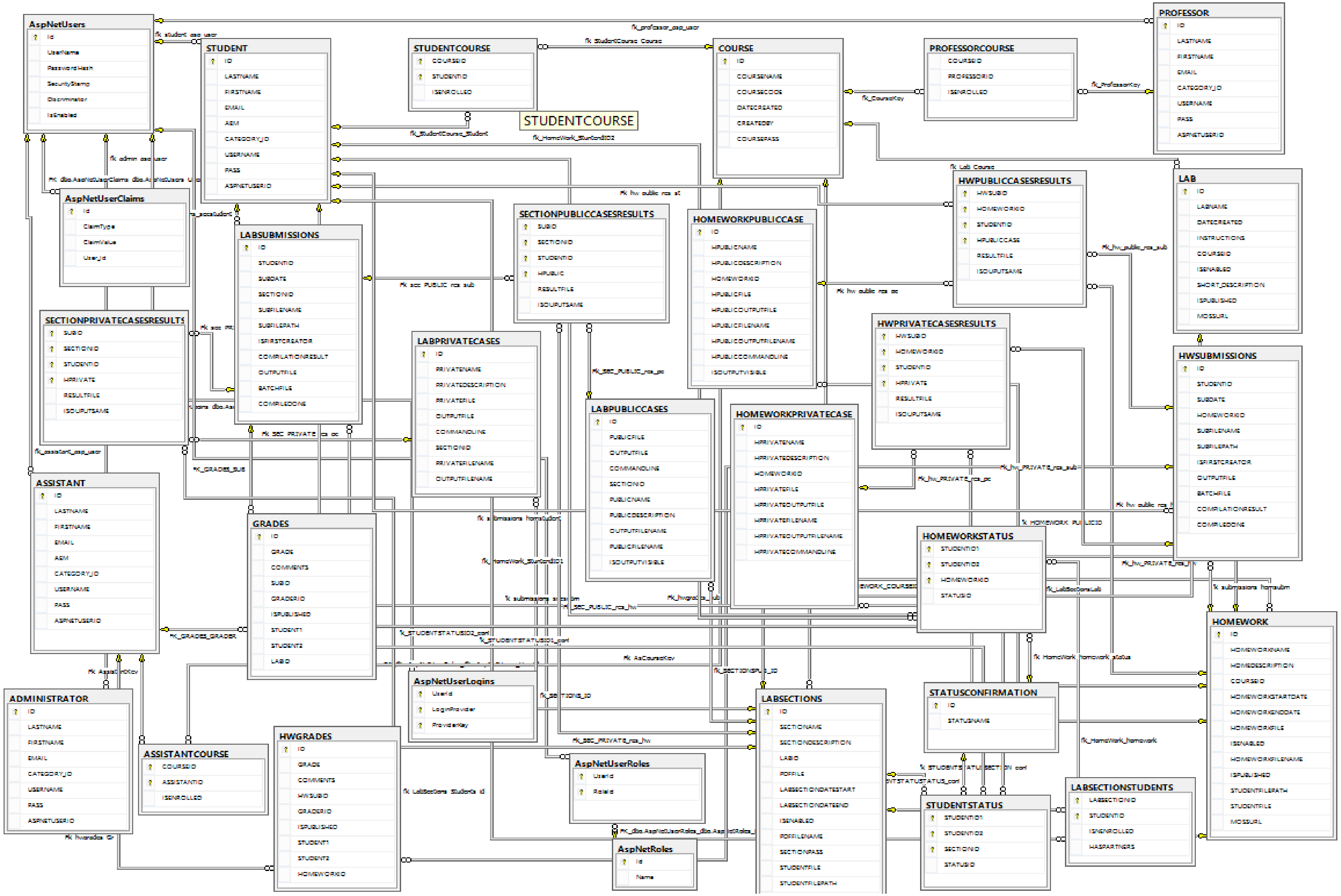


Figure 56. Database Schema