

People's Democratic Republic of Algeria

Abu Bakr Belkaid University - Tlemcen

Faculty of Sciences

Computer Science department

Graduation memoir

For obtaining Master's degree in Computer Science

Option: Network and Distributed System

Theme:

Computer-Based Assessment System Applied to Programming Education

By: Ilyes Skender

Presented on September 18th 2019 before the jury of:

- Mrs. Illes (President)
- Ilyas Bambrik (Supervisor)
- Mr. Belhoucine (Examiner)

2018/2019

Abstract

Academic institutions around the globe have been adopting E-learning systems. The main advantage of these systems is to give tutors a better “monitoring” of the learning process , hence diminishing the distance between them and the student through this system, The impact of this system on programming education is significant. It ‘s vitally crucial for I.T. students to solve as many as possible programming challenges in order to grasp complex concepts and algorithms as well as working with different programming languages.

The combination of a considerable amount of exercises with a high number of students is putting a great pressure on teaching staff in many Universities. Nevertheless, this process of assessing and monitoring can be aided by using a specific kind of e-Learning systems, the Computer-Based Assessment (CBA) systems.

This work proposes a new CBA system for programming courses in the University of Tlemcen. The system was specified to support five different programming languages C, C++, Csharp, Java and Python, set up on a Windows 10 machine that acts as a server, which other students (clients) connect to.

In order to achieve the proposed goals, a number of steps were followed and the results are detailed in this document. A study of state-of-the-art CBA systems was performed on four systems. Then a requirement analysis was made specifically for the University of Tlemcen, based on that and the study of the state of the art, the first C.B.A system prototype was developed and the proposed product was later discussed.

ملخص

عدة مؤسسات تعليمية حول العالم تستعمل أنظمة التعليم الإلكتروني, الميزة الأساسية لهاته الأنظمة هي, والميزة الرئيسية لهذه الأنظمة هي إعطاء المعلمين "مراقبة" أفضل لعملية التعلم, والتي يمكن تحقيقها من خلال تقليص المسافة بينهم وبين الطالب باستخدام هذا النظام, تأثير هذا النظام على تعليم البرمجة مهم جدا, من الأهمية بالنسبة لطلاب علوم الكمبيوتر حل أكبر عدد ممكن من تحديات البرمجة من أجل فهم المفاهيم والخوارزميات المعقدة, بالإضافة إلى العمل باستخدام لغات برمجة مختلفة. إن توفير كمية كبيرة من التمارين لعدد كبير من الطلاب يفرض ضغطاً كبيراً على أعضاء هيئة التدريس في الجامعات, ومع ذلك, يمكن فان عمليات التقييم والإشراف على التمارين البرمجية يمكن تحسينها باستخدام نوع معين من أنظمة التعليم الإلكتروني, و هو نظام التقويم المبني على الحاسوب (سي.بي.أي)

تقترح هذه الرسالة نظام (سي.بي.أي) جديد لتعليم البرمجة في جامعة تلمسان, صمم هذا النظام لدعم خمس لغات برمجة مختلفة, يكون الخادم مبني على نظام ويندوز 10, حيث يتصل التلاميذ (العملاء).

من أجل تحقيق الأهداف المقترحة, تم اتباع عدد من الخطوات وترد النتائج بالتفصيل في هذه الوثيقة, تم اجراء دراسة على اربع أنظمة (سي.بي.أي) متنوعة بمقارنة لخواص هته الأنظمة, ثم تم إجراء تحليل لمتطلبات جامعة تلمسان, بناءً على ذلك ودراسة الأنظمة الأخرى, تم تطوير أول نموذج أولي لنظام (سي.بي.أي), وتم تقييم نتائج المشروع لاحقاً.

Abstrait

Les institutions universitaires du monde entier ont adopté des systèmes d'apprentissage en ligne. Le principal avantage de ces systèmes est de donner aux tuteurs un meilleur «suivi» du processus d'apprentissage, réduisant ainsi la distance qui les sépare de l'élève grâce à ce système. L'impact de ce système sur la programmation de l'enseignement est considérable. Il est d'une importance cruciale que les étudiants de la Technologie de l'Information résolvent autant de problèmes de programmation que possible afin de saisir des concepts et des algorithmes complexes, ainsi que de travailler avec différents langages de programmation.

La combinaison d'un nombre considérable d'exercices et d'un nombre élevé d'étudiants exerce de fortes pressions sur le personnel enseignant de nombreuses universités. Néanmoins, ce processus d'évaluation et de surveillance peut être facilité en utilisant un type spécifique de systèmes d'apprentissage en ligne, les systèmes d'évaluation informatisée (CBA).

Ce travail propose un nouveau système CBA pour la programmation de cours à l'Université de Tlemcen. Le système a été spécifié pour prendre en charge cinq langages de programmation différents, C, C ++, Csharp, Java et Python, configurés sur un ordinateur Windows 10 faisant office de serveur, auquel se connectent d'autres étudiants (clients).

Afin d'atteindre les objectifs proposés, plusieurs étapes ont été suivies et les résultats sont détaillés dans ce document. Une étude des systèmes CBA a été réalisée sur quatre systèmes.

Ensuite, une analyse des besoins a été réalisée spécifiquement pour l'Université de Tlemcen et sur la base de cette étude et de l'état de la technique, le premier prototype de système C.B.A a été développé et le produit proposé a ensuite été examiné.

Acknowledgements

First I would like to thank everyone who contributed in some way to this work. I would like to thank my work advisor Mr.Ilyas Bambrik, for all enthusiasm, efforts and good ideas. He was always open whenever I ran into a trouble spot or had a question about my programming or writing. He consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought I needed it.

I would also like to acknowledge the entire jury of the computer science department at University of Tlemcen as the second reader of this work, and I am gratefully indebted to them for their very valuable comments on this work.

Finally, I must express my very profound gratitude to my parents, friends and family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this memoir. This accomplishment would not have been possible without them. Thank you.

Ilyes Skender.

Abbreviations

CS	Computer Science
CLI	Command Line Interface
API	Application Programming Interface
CBA	Computer-Based Assessment
CGI	Common Gateway Interface
CPU	Central Processing Unit
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
ICT	Information and Communication Technology
IDE	Integrated Development Environment
RMI	Remote Method Invocation
SQL	Structured Query Language
SSL	Secure Sockets Layer
VPN	Virtual Private Network
XSLT	Extensible Stylesheet Language Transformations

Contents

General Introduction	9
Chapter I: State of the art review.....	12
1.1 CBA systems.....	12
1.2 CourseMarker	12
1.3 BOSS	18
1.4 XLx.....	24
1.5 Mooshak	27
1.6 Conclusion:	30
Chapter II Implementation	32
2.1 Introduction:	32
2.2 Development tools.....	32
2.3 Requirements analysis	34
2.4 Design and architecture	35
2.5 Design diagrams.....	39
2.6 Use-Case scenario and experimentation.....	41
2.7 Conclusion	52
General conclusion.....	54
References	55

General Introduction

Every semester, an increasing number of students join CS related University courses. Academic institutions face the challenge of providing their students with better quality teaching while minimizing the amount of additional work for their staff.

Professors are facing a real tricky challenge when it comes to managing their classes for a considerable amount of time [I1], [I2], [I3],[I4], as these classes have been growing in proportion at an excessive pace. Whereas old-fashioned training methods can be improved by audio/visual means and by enhancements in the field of remote learning. However, from a homework assessment perspective, teachers still face the same old problems. Moreover, the usual methods we practice in the assessment field are insufficient or inapplicable, especially in disproportionate classrooms.

A Computer-Based Assessment software (CBA) has become the long-awaited solution for teaching staff facing these problems and has become an increasingly important tool of teaching over the years. With the use of CBA software, the marking and distribution of grades is homogenous and fair, feedback to its users is instant, feedback quality increases, and hundreds of hours of student coursework evaluation can be saved. Thus, freeing the time of the academic personnel for teaching and research.

➤ Problem statement

Universities that offer CS related majors have reported a difficulty in teaching programming skills or assessing students', which mostly consists of student that do not have access to the appropriate platform to test and execute their code. Another major obstacle that faces academic institutions is the lack of staff per each class. For starters, a teacher cannot manage to evaluate each student assignment individually in large classes. Lastly, during coding exams, teachers find it very challenging to fairly asses every student, as students usually do not submit their code simultaneously. Besides, it is usually difficult to verify the solution correctness from a semantics point of view.

➤ Motivations for research and development

The two essential motivations behind this research is to study and discuss what a CBA is, by presenting current CBA systems that are in use such as (CourseMaker, BOSS, xLx, Mooshack....), and going through details about each one of them, as well as the development of our own CBA software, that caters to the needs of instructors during practical exercises classes(T.P), that are involved in teaching programming courses (object oriented programming, algorithms, Python, Java, C#, C++ and C) in UABT.

➤ Document Structure

This work is structured in three chapters, the first one being this introduction.

Chapter 1 gives an overview of the features available in four of the CBA systems currently in use (state of the art). These features are divided in topics for a better comprehension. Subsequently, the chapter ends with a conclusion.

The second chapter goes into implementation details of the prototype that has been developed based on the needs of University of Tlemcen

Chapter I: State of the art review

1.1 CBA systems

CBA systems have been in demand since the earliest CS related courses in universities. Incidentally, one of the first examples of such system is the Hollingsworth's system for assessing assembly code written on punch cards in the early days of programming. In the 80's and 90's, several CBA systems based on command-line interfaces and manual operation of scripts were developed [II1]. Afterwards, new CBA systems such as CourseMarker, BOSS, TRAKLA and RoboProf were web-based platforms that emerged.

A recent example is the EduJudge project, developed by the University of Valladolid (UVa) which is taking the concept of CBA systems one step further, by integrating on-line judge into effective e-learning [II2].

1.2 CourseMarker

CourseMarker was implemented at the University of Nottingham in 1998 and tested for the first time as a replacement for Ceilidh in the academic year 1998–1999. The first courses that were authored covered two Java programming modules containing 35 exercises. CourseMarker was made available to other academic institutions in February 2000 and is in use in more than 20 academic institutions where it supports the automation of coursework in classes that have as many as 1500 students[II3]. As of 2005, about 50 universities worldwide have taken copies of CourseMarker for trial and approximately 15 institutes purchased the system for actual use [II4].

1.2.1. The Automatic Assessment

CourseMarker has a set of marking tools to automatically assess the quality of students' submissions. Three tools are used for assessing programming exercises [II5]:

1-The Typographic Tool Checks text for solution's typographic features.

2-The Dynamic Tool Tests solution's behavior via interaction (e.g. runs it against test data)

3-The Feature Tool Inspects for features specific to the exercise (for example, an exercise set on a unit that teaches the difference between while and for statements would typically include a feature test to ensure the appropriate use of each construct).

1.2.2. Setup of New Exercises

In CourseMarker, every exercise is parameterized using a property file, each property is used to specify a behavioral aspect of the exercise's assessment in a controlled environment. For example, an exercise's skeleton filename, the maximum allowed number of submissions and its availability status (e.g., open/closed/late), the maximum allowed number of lines that a student program is able to output, the maximum CPU time allocated to run of a student's solution, are all examples of such properties[II6]. In addition, the following files need to be specified:

- *mark.java* - contains Java code that can access CourseMarker's state or call external tools, like compilers, to help with the assessment;
- *properties.txt* - defines parameters for the exercise settings, such as: exercise's filename, maximum number of submissions, whether the exercise is open or closed to students, maximum size of a solution's output, maximum running time, etc;
- *mark.scale*- if required, this file can be used to specify information on how to scale marks.[II7]

1.2.3. Solving Exercises

The workflow followed by students for solving an exercise can be defined as follows [II8]:

1. Logging into the system and reading the exercise description which explains the specifications that the solution has to comply with, and the formats of any input and output files;
2. Obtain a skeleton solution along with header files and/or testing tools;
3. Solve and test the exercise locally, in their workspace;
4. Submit their solution for assessment, by uploading it to the system, and receive grades and feedback. Solutions can be resubmitted as many times as allowed by the exercise developer.

1.2.4. Feedback System

CourseMarker provides a sophisticated feedback mechanism and presents the students with their results in detail. CourseMarker's GUI client displays information using a tree component. The students can navigate the tree and easily identify the reasons for their lost grades. Comments on how to improve their solution or links with further reading material can be given.

In CourseMarker, the amount of feedback can be regulated to match the needs of the classroom. In addition, the exercise author can choose whether the student's mark is to be displayed in a numeric or in an alphabetic scale. The association between numeric values, letters, colors and shapes can be customized. [II9]

1.2.5. Management of Submissions

When using CourseMarker, course tutors have access to a set of functionalities related to the management and analysis of students' solutions. Among them are: view students' solutions and marks, view statistics of marks for particular exercises, students that submitted their work and those who did not, search for and view plagiarism results.[II10], [III1]:

1.2.6. Architecture

In order to satisfy the deployment requirements arising from the wide range of computing configurations that exists in academic institutions, CourseMarker was designed using the Java language. Java facilitates platform independence and offers a convenient distribution mechanism with its the Remote Method Invocation (RMI) mechanism. RMI is effective and relatively easy to use as it makes the network transparent to the programmer. The latter feature has been especially sought as in the early stages of development. [III1].

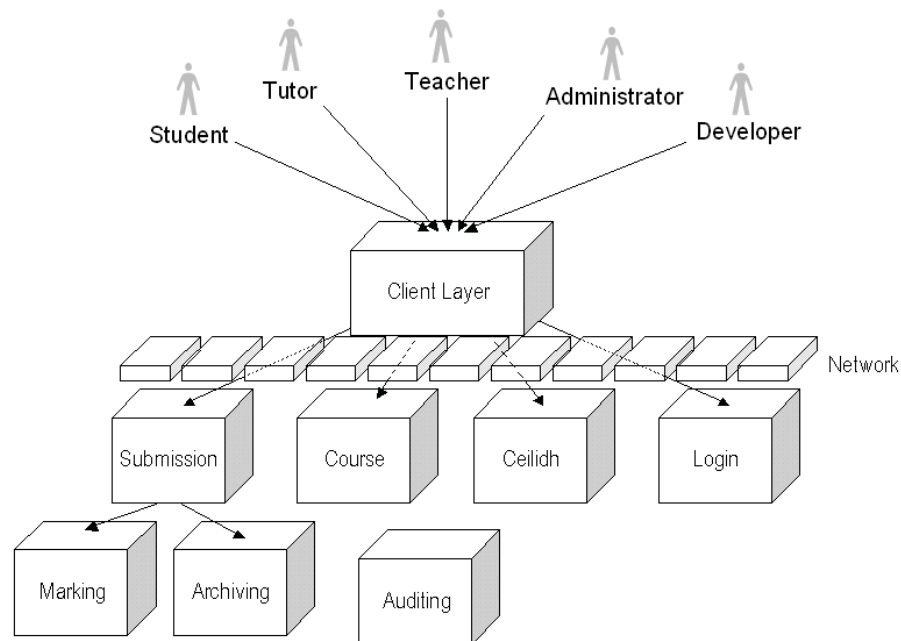


Figure 1A high level view of CourseMarker main parts.

Server name	Function
Login	Authorizes users to perform the requested tasks
Course	Manages course material and responds to requests
Submission	Decides between accepting and rejecting a submission
Marking	Marks the submission and produces analytical feedback
Archiving	Archives student work and issues unique receipts
Auditing	Logs audit trails for various configurations
Ceilidh	Provides course related information to its clients

Table 1. CourseMarker’s servers and their responsibilities

1.2.7. Security

Security has been a primary design concern for CourseMarker. For programming courses, CourseMarker uses the SUID and GUID security mechanisms when running under Unix. Alternatively, the “runas” command can be used when running under Windows 2000. CourseMarker also uses encryption for the sensitive information exchanged between servers.[III1]

1.2.8. Graphical Interface

The GUI of CourseMarker was developed with Java Swing library. In Figure 2, the tree component on the lower left represents the available courses, units and exercises and allows browsing through the course’s material.

The options available to the student are context dependent and sensitive to the state of the exercise. For example, students that have not compiled their programs cannot select the option to

submit. Options appear as menu items, toolbar buttons and shortcuts. Students can personalize their view in a number of functional and presentational fashions.

The area where the notes are presented at bottom right can render either text or HTML documents. The students can also access a graphical representation of their marks as well as the feedback received from the different assessments(figure3).[III1]

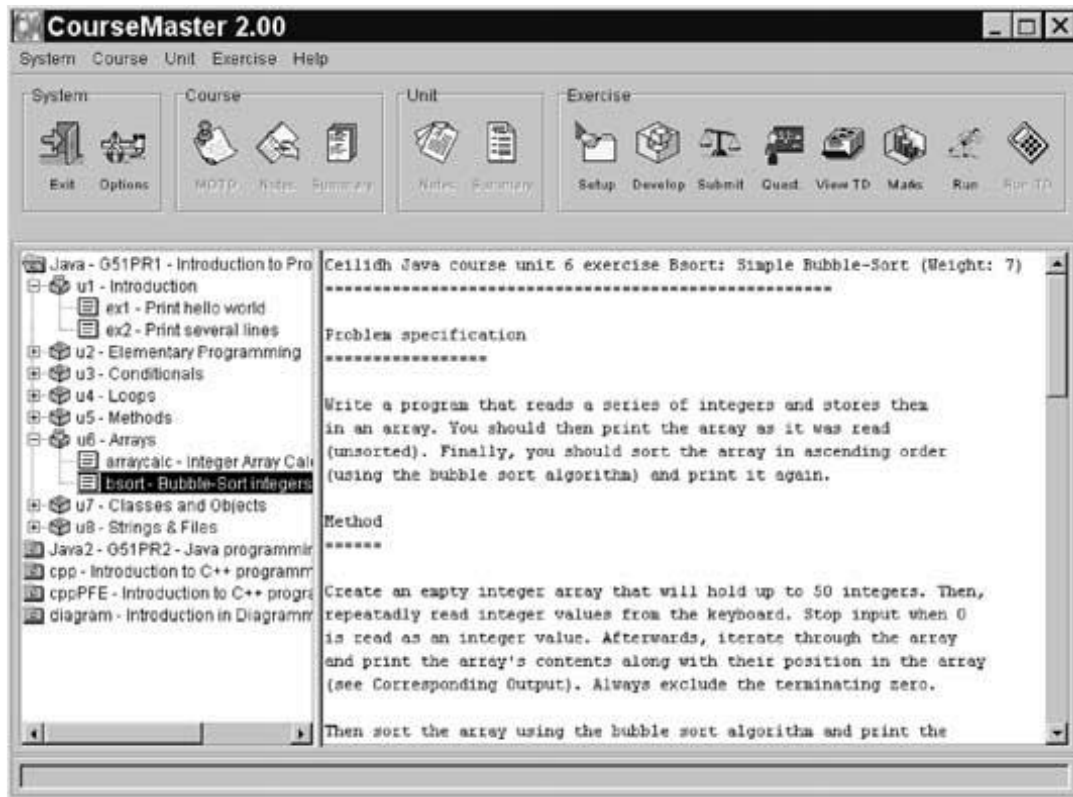


Figure 2. CourseMarker student interface: the student view for a course.



Figure 3. CourseMarker student interface: exercise results for a student.

1.3 BOSS

BOSS is a CBA system developed by the University of Warwick. It started back in the mid 90's with only LUI (Line User Interface) [II12]. However, it was improved subsequently as a web based application and later as java GUI. Thus giving more options to the users [II13, II14]. Since BOSS is an open source product, the authors do not have accurate information as to which other institutions have deployed it and what local changes they have made to the software [II14].

1.3.1. The Automatic Assessment

There is no single correct approach to the problem of assessing programming assignments. Different practitioners may adopt different strategies, depending on the specific aims and objectives of the course they are teaching and on their own style and preferences. BOSS is a tool for the assessment of programming assignments, which supports a variety of assessment styles

and strategies whilst providing maximum support to both teachers and students. Within this framework, the teacher has access to automatic tools to assist in the assessment process, which can be used as much (or as little) as the teacher deems appropriate.[II15]

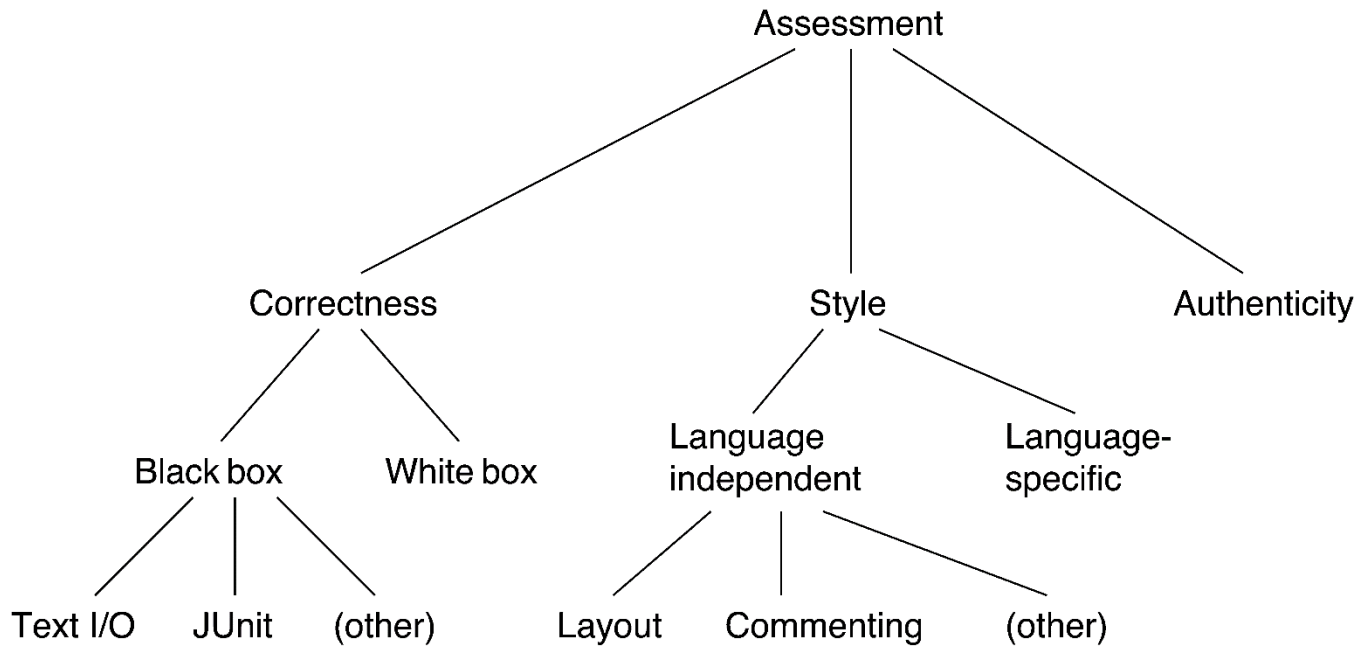


Figure 4. The assessment process.

1.3.2. Setup of New Exercises

In order to setup new exercises, exercise developers use a data model available in BOSS: the *component model*. This model is intended to support arbitrarily complex assessment structures, composed by one or more problems [II11]. Then, for each problem, the developers have to use one of the methodologies, presented in the previous section, for defining the test cases to assess students' solutions. Part of these tests can be made available to students, so they can use them to check their solutions before submitting.

1.3.3. Solving Exercises

The workflow followed by students to solve an exercise on BOSS are the following:[II16]

1. Login into the system and read the exercise description, which explains the specifications that the solution has to comply with, and the formats of any input and output files;
2. Solve and test the exercise locally, in their workplace;
3. Submit their solution for assessment, by uploading it to the system. Before submitting, students can run a set of automatic tests (which are a subset of the final tests) on their programs. There is no limit for the maximum number of submissions (within the prescribed deadline).

1.3.4. Feedback System

When running the set of automatic tests available before submitting their solution, students receive immediate feedback about whether or not their program succeeds in producing the right outputs. However, the feedback of their final mark is not immediate.

After the deadline for the submissions has passed, course staff can run the final automatic tests and grade the solutions. [II15]

1.3.5. Management of Submissions

The BOSS software permits staff to perform five principle tasks [II15]:

1. Automatic tests can be run on the set of student submissions and as part of the grading process.
2. Plagiarism detection software assists in the identification of potential source-code plagiarism.

3. Staff authorized by the module organizer can moderate the marks given to students' work by other markers.
4. Associate manual feedback to students' submissions.

Some other features are available from BOSS' interface: consult student details, penalize late submissions, save submission, submit on the behalf of a student, publish marks, etc.

1.3.6. Architecture

An overview of the system architecture can be seen in Figure II4, showing its primary components. There are four data repositories (represented by grey rounded boxes), which store marks, information about students and their submissions, results of automated tests, the results of plagiarism detection, and other necessary data.

Technologies like SSL and RMI are used to support the communication between the different components.[II15]

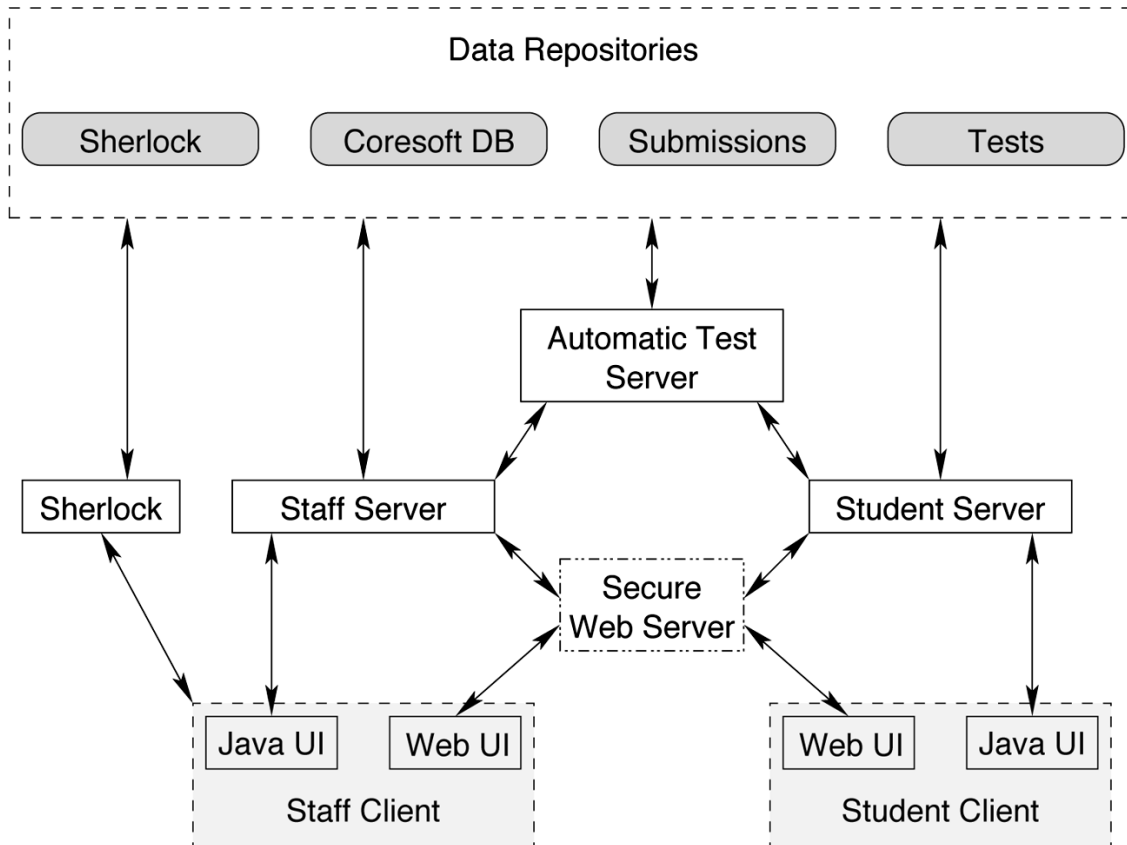


Figure 5. BOSS architecture

1.3.7. Graphical Interface

Since its release, BOSS provides two types of GUI's: one designed for desktops in the university's network which were developed with Java Swing library, the second one is a hosted online for remote users [II15]. The Figures below present screenshots of the student interfaces respectively, the Java client and the web client.

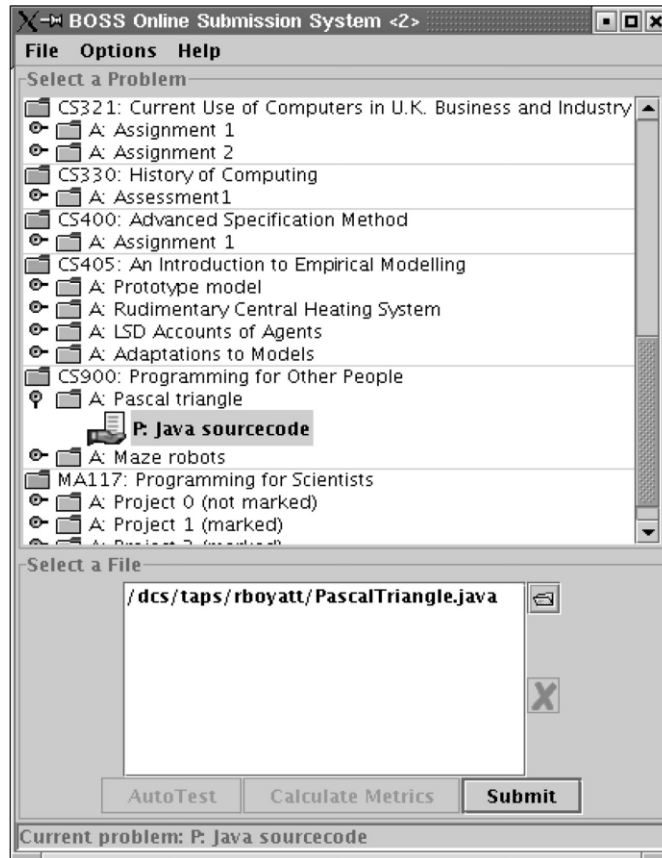


Figure 6. BOSS student interface - Java client

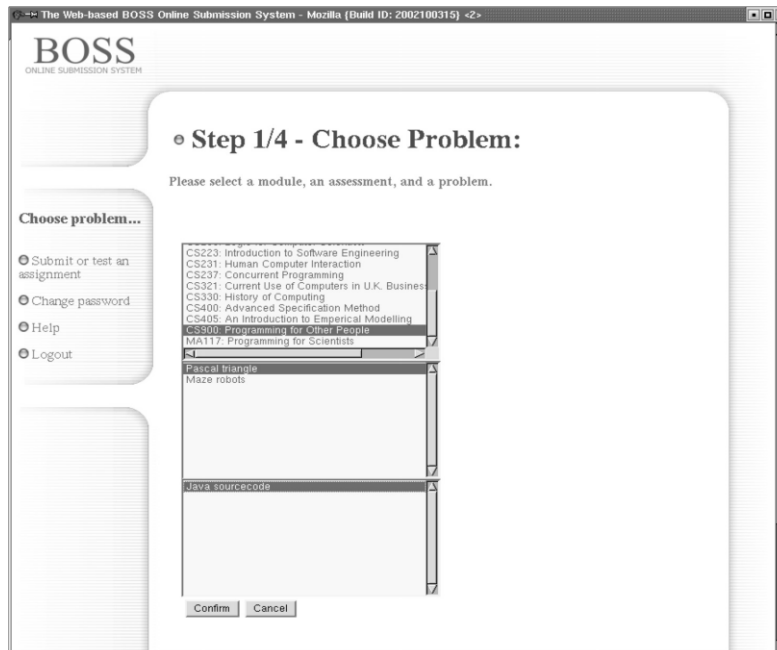


Figure 7. BOSS web interface

1.4 XLx

XLx is an abbreviation for eXtreme e-Learning eXperience, a web based online learning platform developed at the University of Muenster and can either be used in university or commercial contexts. Its main objective is to support the exercise portion of technically oriented university courses [III17]. Initially, it supported the training of skills related to SQL queries, object-relational features of SQL and transformation of XML documents with XSLT or XQuery [III18]. Later on, the possibility of automatic assessment of Java programming exercises was also incorporated. Quite recently, xLx was redesigned and rewritten over a 5-month period by a 12-students team and a xLx2 system is now available.

1.4.1. The Automatic Assessment

The xLx system automatically assesses the exercise solutions with two types of tests: static and dynamic. Only one static test is performed and simply consists in verifying whether or not the code compiles. The dynamic tests are specified with the JUnit test framework. To better handle the compilation and test process, Apache Ant is used to both compile the submitted solutions and execute the JUnit test cases [III17].

1.4.2. Setup of New Exercises

When configuring a new exercise, tutors have to set up the followings:

- **Section** - the exercise can be assigned to an existing section comprising associated exercises;
- **Level** –to define the level of difficulty in the exercise;
- **Exercise type** –XLx supports only Java written programs;

- **Exercise text** –the description of the exercise

In addition, the files containing the dynamic JUnit test cases need to be indicated.

When creating the tests (which as to be done with an external IDE or text editor), tutors can declare them either as *public* or *hidden*. The result of *public* tests is presented to students, but does not count for the grading of the exercise, while *hidden* tests results are not presented and count for the grading.

The system also provides the tutors with the facility of uploading sample solutions that can be used by the correctors of the exercise. [III17]

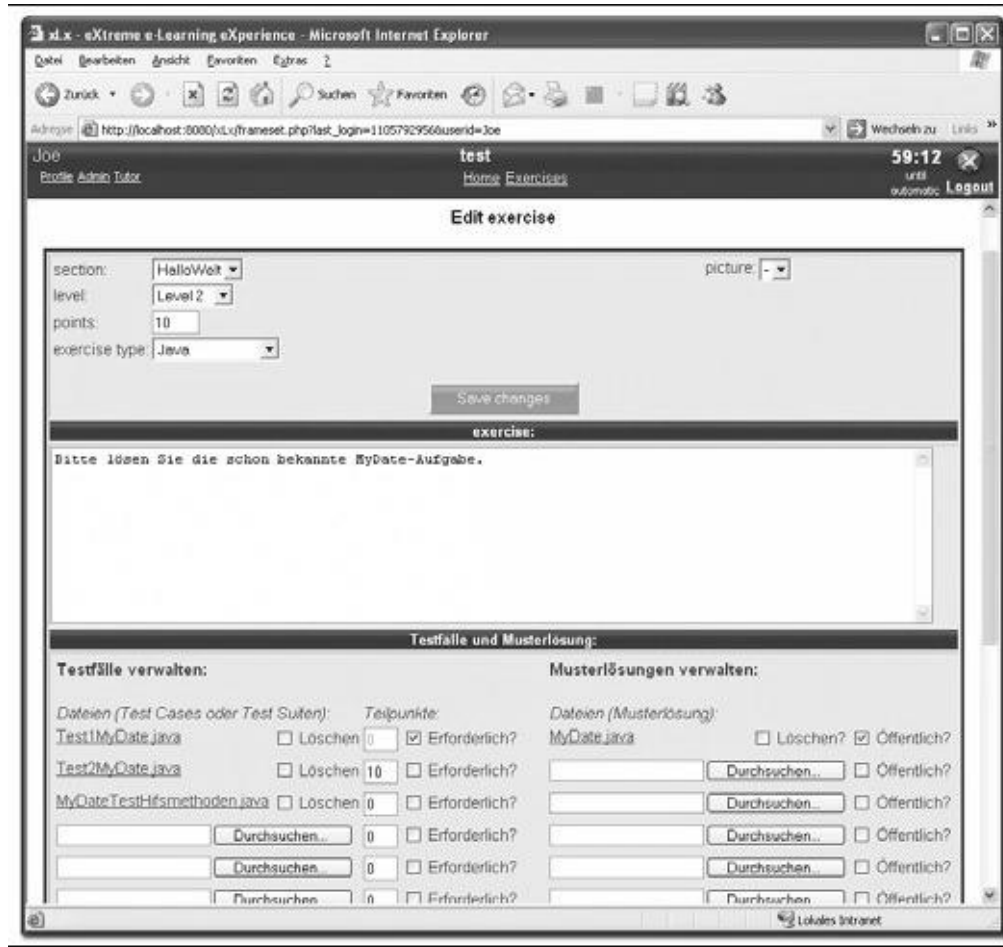


Figure 8. xLx GUI for a tutor to configure new exercises.

1.4.3. Solving Exercises

The workflow followed by students for solving an exercise can be defined as follows:

- Login into the system and read the exercise description which explains the specifications that the solution has to comply with;
- Solve and test the exercise locally, in their workplace;
- Submit their solution for assessment, by uploading it to the system.

After a submission is made, i.e. the code compiles, the solutions cannot be changed anymore.

1.4.4. Feedback System

Due to the static assessment, as soon as the system receives students' submissions, an immediate feedback about whether or not their code compiles is sent back to the students. In the case of an unsuccessful compilation, students get the chance to submit the code again. After a successful compilation, students receive immediate feedback about the performance on the *public* test cases. The system distinguishes between two kinds of test case failure: the program output does not match the expected results or an unhandled exception occurred during the execution.

The system provides the tutors with a feature to easily annotate students' source code. For these annotations, a special Java syntax-highlighting scheme is used to make the reading of the source code more comfortable [II17].

1.4.5. Architecture

In terms of architecture, xLx is a web based application implemented in typical three-tier client-server architecture, which makes it accessible via a standard web browser. The platform is layered above an Apache Web server and a mySQL data base running on Linux [II18].

1.4.6. Graphical Interface

The Web pages that compose xLx's graphical interface are generated dynamically by PHP4 scripts and Java Servlets. Figure 8 presents the graphical interface used by a tutor to configure new exercises, while Figure 9 illustrates the window where students can consult the annotations added by their tutors.



Figure 9. xLx consultation window.

1.5 Mooshak

Mooshak is a CBA system that was developed by the University of Porto, with the main purpose of conducting programming contests. However it is increasingly being used in programming education, to give instant feedback on practical classes, to receive and validate assignments submissions, to pre-evaluate and mark assignments, etc[II19].

1.5.1. The Automatic Assessment

Mooshak automatic assessment consists of two different types of analysis [II20]:

- **Static** - checks for integrity of the submission. For instance, the size of the program is verified to prevent a denial of service attack by a submission of a very large program. Also, the system tries to compile the code. If the compiler produces errors or warnings, the assessment is aborted and an error message is presented to the student. Otherwise, an executable program is produced to be used in the dynamic analysis;
- **Dynamic** - in the dynamic analysis, the solution is tested against a set of test cases, defined by input and output files. Depending on the behavior of the submitted program, the system produces different classifications which have different levels of severity [LS03]. From the most to the least severe, the classifications may be: Requires re-evaluation, Time-limit exceeded, Output too long, Run-time error, Wrong answer, Presentation error and Accepted;

1.5.2. Setup of New Exercises

When setting up a new exercise, and after defining the textual description, exercise developers must specify a set of input and output files for testing the submissions. They also need to upload a sample solution to the problem which is used by the system to automatically determine the maximum allowed running time for the submissions [II20].

1.5.3. Solving Exercises

The workflow for solving an exercise can be defined as follows:

- Login into the system and read the exercise description which explains the problem and the formats of the expected input and output;

- Solve and test the exercise in a local workplace;
- Submit the solution for assessment, by uploading it to the system. The system shows

the classification resultant from the static and dynamic analyses.

1.5.4. Feedback System

Immediately after a submission has been made, Mooshak produces feedback which consists in the classifications produced by the static and dynamic analysis [II20]. In case of a compilation error or warning, an additional message containing the compiler output may be presented.

1.5.5. Management of Submissions

Mooshak provides some features related to the management of students' submissions. For instance, it is possible to re-evaluate submissions, by re-running the automatic assessment process. This is useful when mistakes in the input or output files are detected. It is also possible to consult information related to the submissions: source code, running time, memory used, produced output, etc.

1.5.6. Graphical Interface

Figure 10 presents a screenshot of Mooshak's HTML interface. The screenshot illustrates the tutor view of the list of submissions [II20].

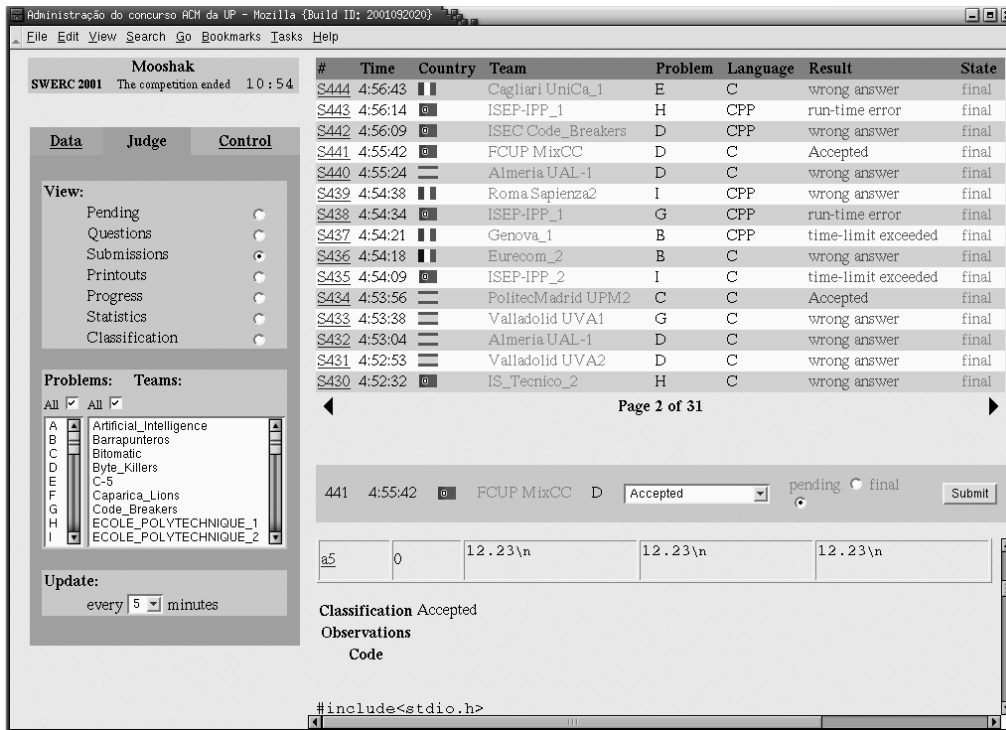


Figure 10. Mooshak graphical interface tutor's view

1.6 Conclusion:

To conclude, this chapter went through four different CBA systems (CourseMaker, Boss, Xlx and Mooshack) the subjects were analyzed by different criterion, such as the architecture, the GUI and the assessment, another point worth mentioning is that these CBA systems have slightly different uses, for instance Mooshack can be used in coding competitions rather than education as in other systems.

Chapter II Implementation

2.1 Introduction:

Using the requirements of the practical courses on programming education in the University of Tlemcen as guidelines, a prototype of the new CBA system was implemented. In this prototype, the assessment server is the instructor's computer, which other students connect to. Using their GUI, students can upload, execute and delete their files from the server. Each student connect to the server. The latter creates a thread which is responsible of all the exchanges between the server and the client. It starts with the authentication procedure before enabling the user to perform all the available operations.

This CBA system was developed with Python 3.7 whereas PYQT5 is used for developing GUI on the client side. In this chapter the different steps of development as well as, an intensive explanation of all the features that were built into the code on both sides (server and client) are represented.

2.2 Development tools

Python 3

Python is a dynamically typed and high-level, general-purpose programming language. Guido van Rossum released it in 1991, as it is an interpreted language; its design philosophy supports code readability with its remarkable use of whitespace for mandatory code formatting. It's also an object oriented programming language, which helps to write logical code for small and large-scale projects. [III1]

Python 3.0 was released in 2008. Although this version is supposed to be backward incompatible, later on many of its important features have been backported to be compatible with version 2.7.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. As a non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.[III2][III3] [III4]

PYQT5

PyQt is a set of Python bindings for the cross-platform application framework that combines all the advantages of Qt and Python. With PyQt, you can include Qt libraries in Python code, enabling you to write GUI applications in Python. In other words, PyQt allows you to access all the facilities provided by Qt through Python code. Since PyQt depends on the Qt libraries to run, a version of Qt is also installed automatically on the host. [III5]

Qt designer

Qt Designer is the Qt tool for designing and building graphical user interfaces

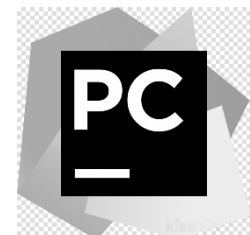
(GUIs) with Qt Widgets. You can compose and customize your windows or dialogs in a what-you-see-is-what-you-get (WYSIWYG) manner, and test them using

different styles and resolutions.[III6]



Pycharm IDE

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger



, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.[III7]

Sqlite3

SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows to access the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It's also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.[III8]



2.3 Requirements analysis

The first step in developing our CBA prototype is the requirements analysis, which is the process of determining user expectations for a new or modified software product. These features, called requirements, must be quantifiable, relevant and detailed.

In our case, the analysis was done from two aspects, the students' and the tutors'. Each one will be discussed in details below.

➤ the tutors:

In most of my discussions about the subject with teaching staffs from several institutions, such as the University of Tlemcen, It was reported that instructors find it very challenging to oversee and direct all the students solving their exercises in practical work classes, as well as reading every student submission and make recommendations to the latter based on that.

Also during programming examinations and quizzes, one teacher cannot carry out the surveillance when assessing some student's code on his machine.

➤ the students’:

As a former student in both graduate and undergraduate courses, in two different institutions, we often as students face a difficulty in running our code. The problem can be caused by, not having the appropriate compiler, and if so sometimes it’s required to have certain libraries pre-installed on every machine in the class. Adding to all that, not all students know how to properly configure their machines, and finally there are numerous programming languages, that require special operating systems to run on, and of course not every machine has it installed. All the aforementioned issues can take up to two sessions to setup, which is a waste of time during the start of every semester.

From what has been analyzed above, it’s concluded that, in order to cater for all the needs of each group, a central system that acts as a server has to be installed in each class. The tutor has full access to the server machine whilst students use their machines to connect as clients to the server. Each client machine has to install a client interface application.

2.4 Design and architecture

The proposed system adheres to a client/server architecture where the machines are connected via a network. The servers are setup to handle every client simultaneously whereas the clients can perform a number of operations when connected to the server.

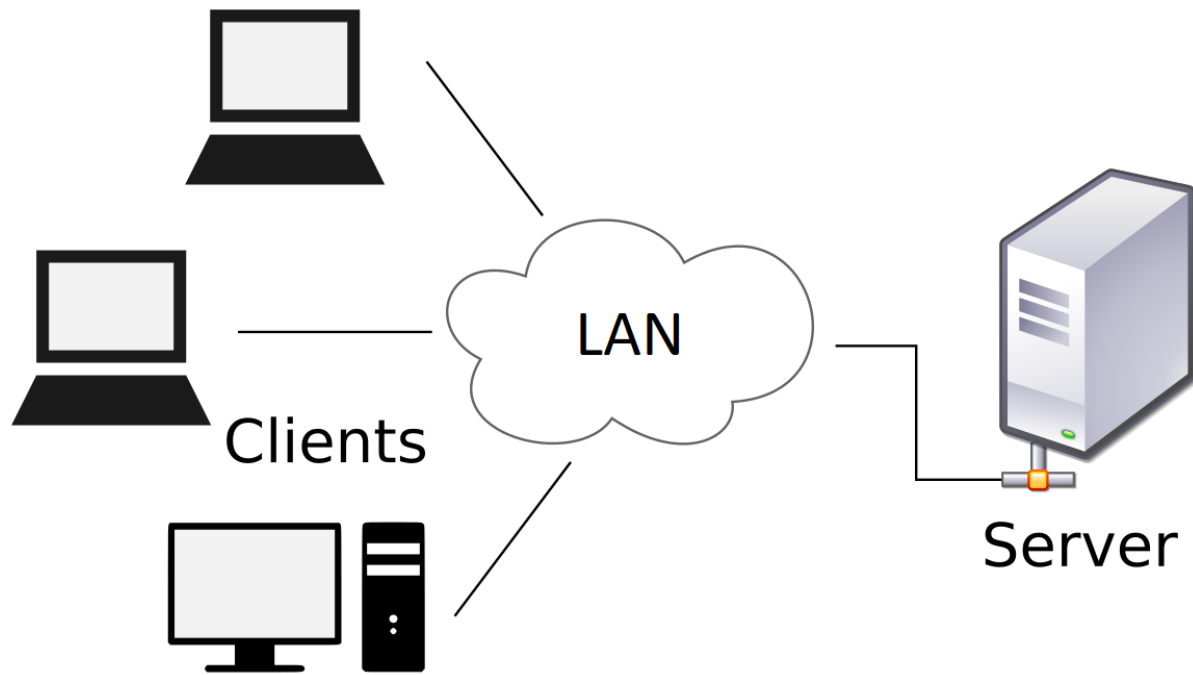


Figure 11. Client/Server architecture.

The server:

The main concept of designing the server is providing the class with a central execution platform, that can handle different programming languages, and help tutors in the assessment process.

The server application is responsible for:

- Credentials checking - a client's user name and password are checked against a database in the server machine.
- Connection management - establishes and close socket connections with clients.
- Listening to the clients applications – a client can send different types of requests
- Create a directory for each connected client

- Download a file from the client machine and store it in his directory
- Deleting a file from the directory following a client's request
- Receiving an execution request followed by a file name from the client
- Checking what language that file is written in
- Checking for malicious code inside the file and deciding whether to execute it or not
- Executing the file and storing the results or sending an error message
- Sending the output of the execution to the client

On top of that, the server has a CLI that provides the tutor with access to the status of the clients. Also, the application gives full access to student's directories, and inside them, the tutor can open code files as well as results files, which can be used for assessment.

A client's database is accessible through a special CLI, with which the tutor can add or delete users.

The client:

The aim of the design of the client application was to provide an environment in which the student can focus on developing the right code for their assignments rather than installing, configuring compilers and libraries, compatibility issues and operating system requirements.

The client application is responsible for:

- Attempting to connect to the server from the address provided by the user
- Send the credentials to the server to check for the authenticity of the client

- Uploading code files to the server
- Sending a deleting request to delete files from the clients directory
- Sending an execution request to the server
- Displaying the results from the server

2.5 Design diagrams

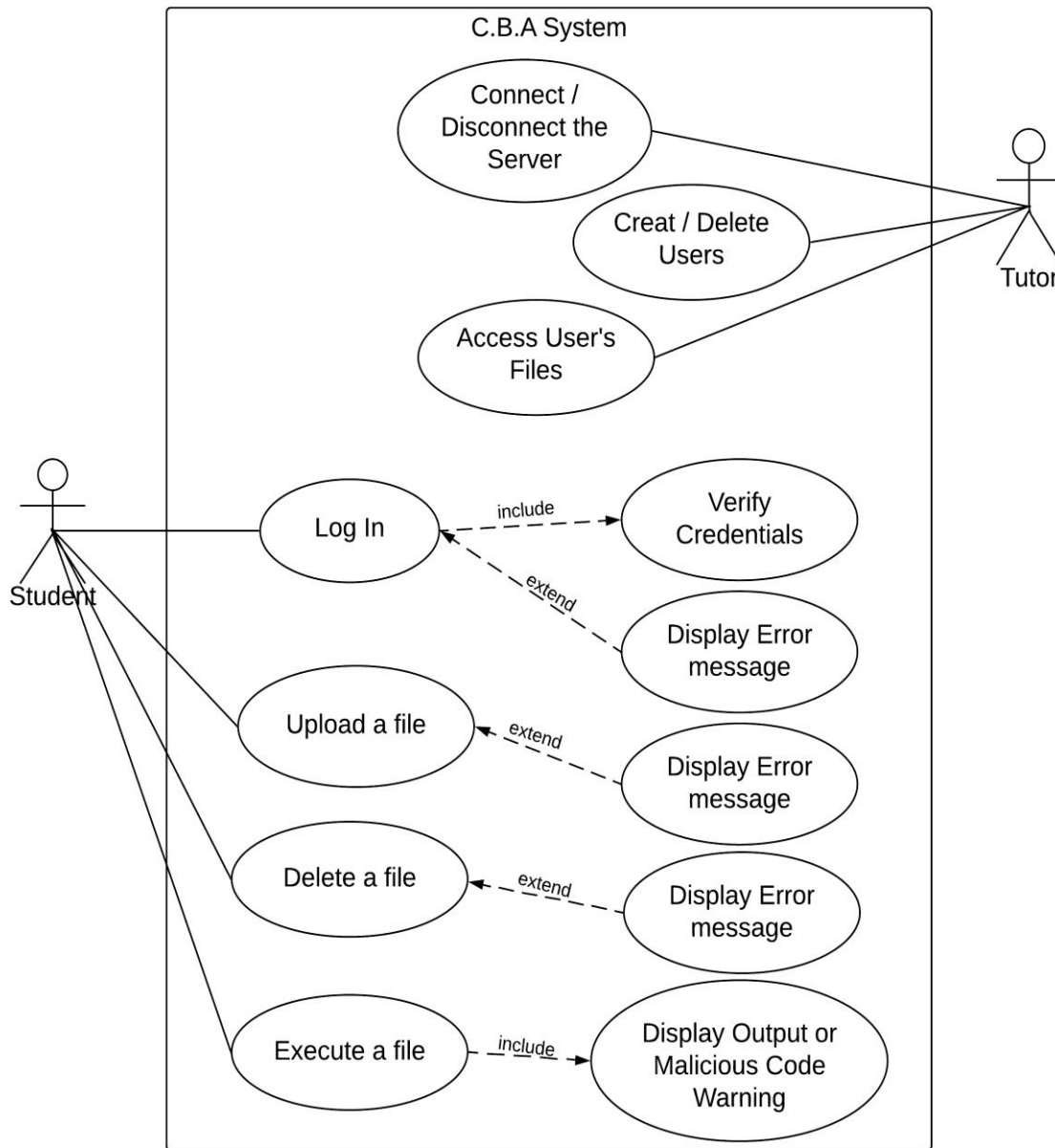


Figure 12. Use-case diagram

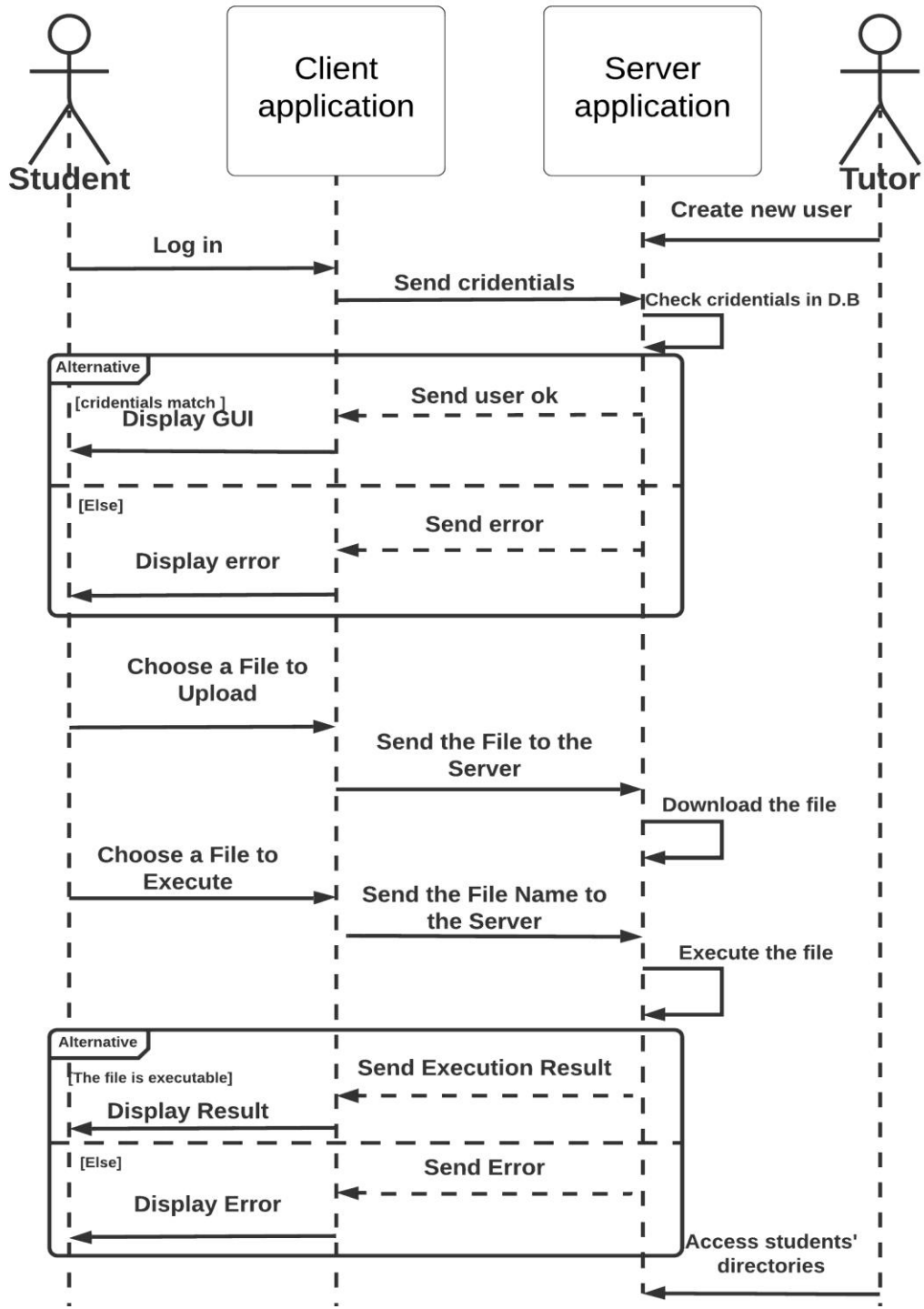


Figure 13. Sequence diagram

2.6 Use-Case scenario and experimentation

2.6.3 Setting-up the server

Before the server can execute any students' files, the languages that are supported by the CBA system (Java, Python, C#, C++, C) need to be pre-installed, and added to the PATH environment variable. In this case, the server application is installed on a Windows 10 machine.

One way to access environment variables in Windows 10 is (Control Panel\System and Security\System), then “Advanced system settings”, the system properties window will pop up, and then the environment variables can be edited.

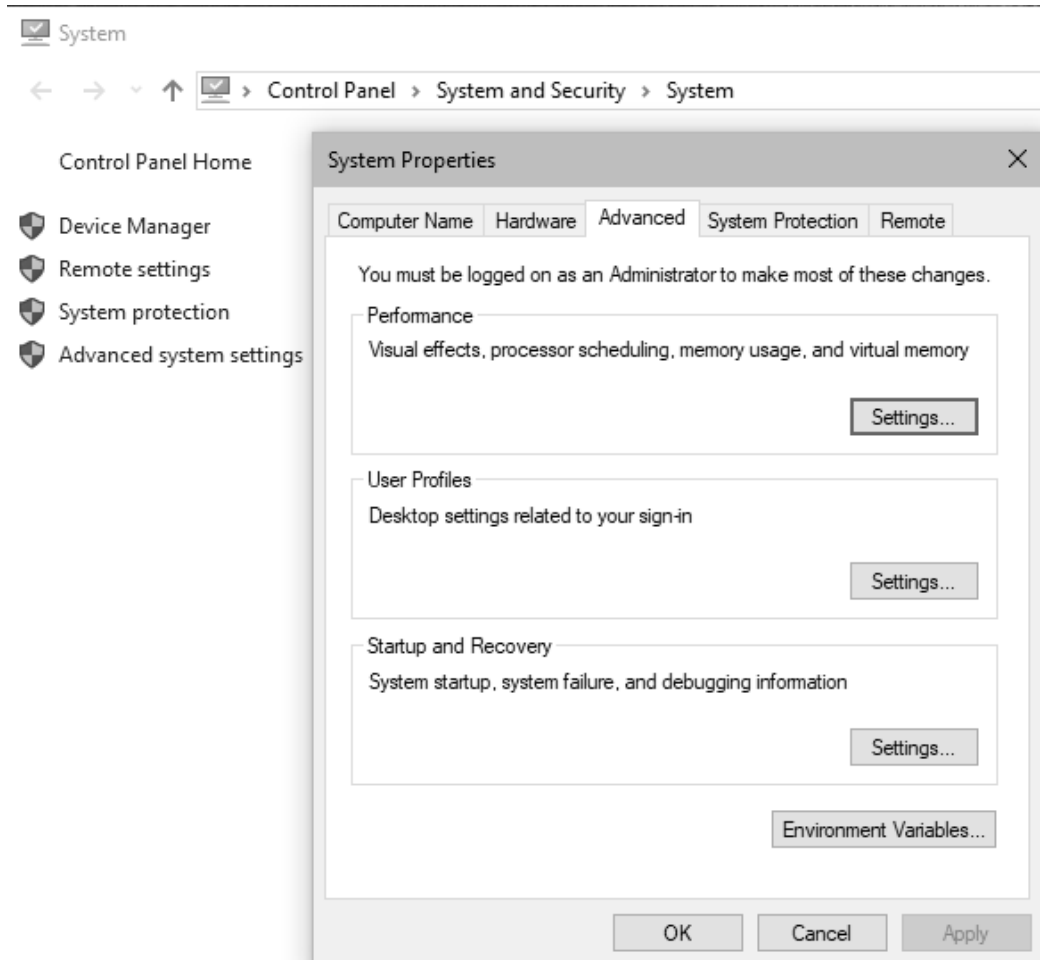


Figure 14. Windows 10 advanced system properties

After accessing the environment variables, the user's path has to contain the compilers of C, C++, C#, Java and the interpreter of Python 3.7.

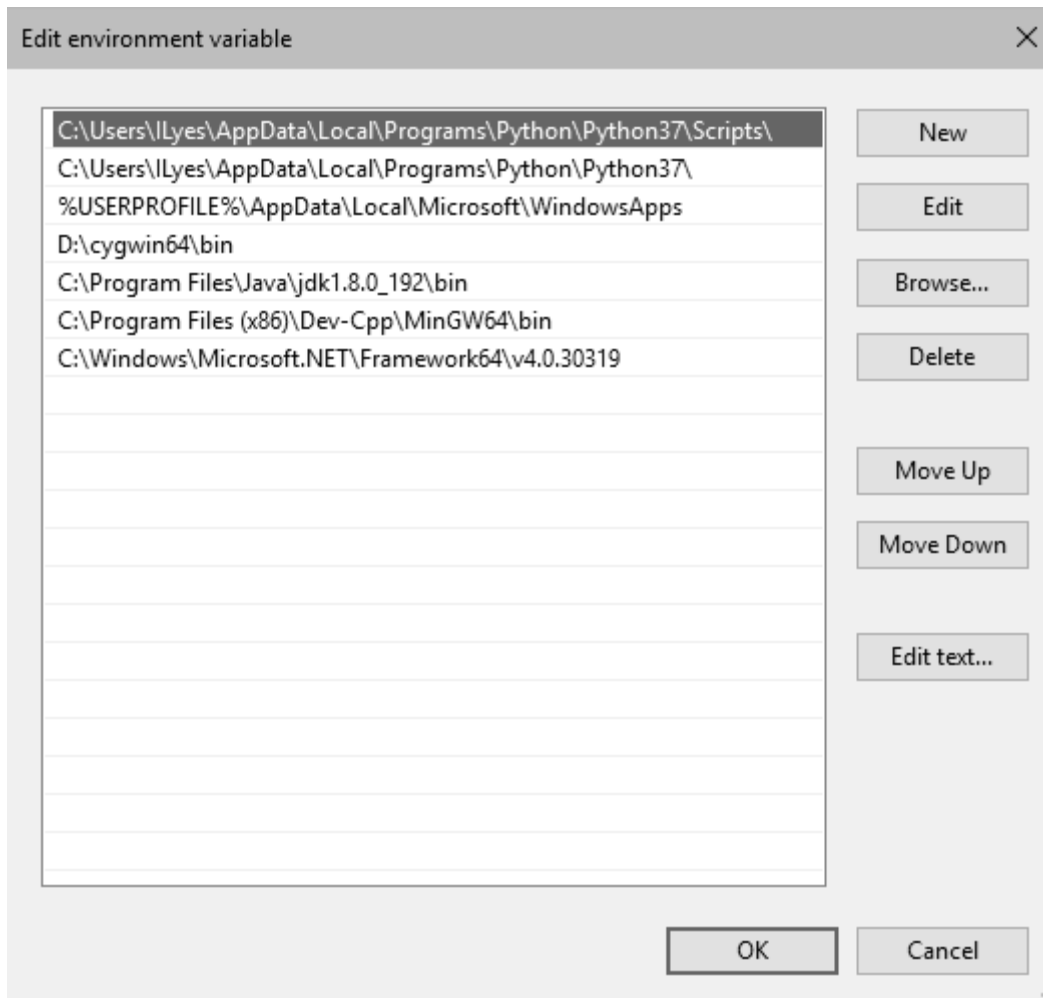
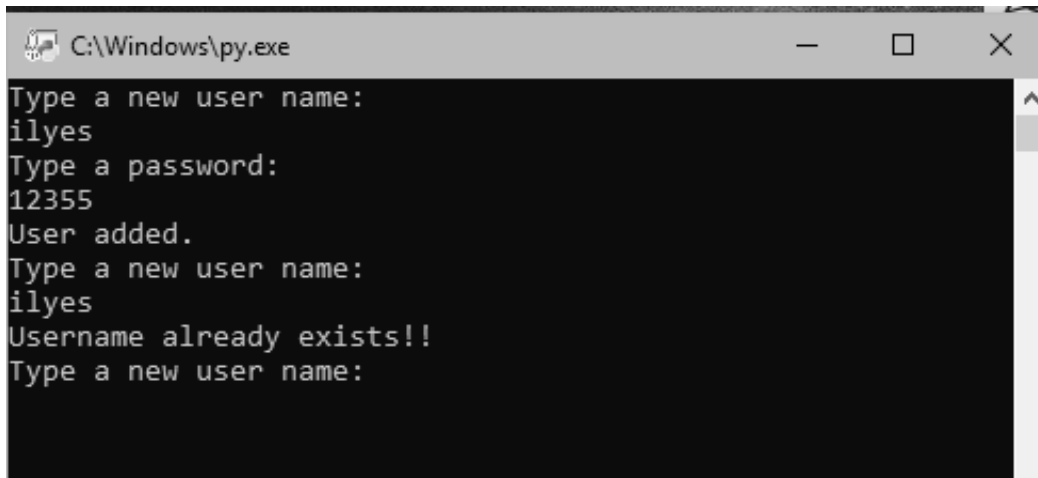


Figure 15. Environment variables

2.6.4 Adding students

The tutor can add students from a standalone CLI application. It creates a database if the latter does not exist already, then it asks for a new username and checks if it exists in the table, if not it stores the username and password in the table.

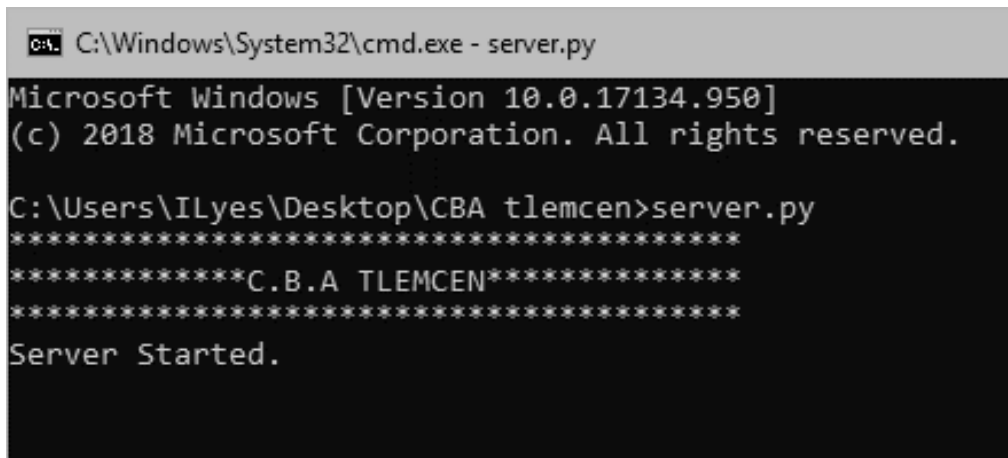


```
C:\Windows\py.exe
Type a new user name:
ilyes
Type a password:
12355
User added.
Type a new user name:
ilyes
Username already exists!!
Type a new user name:
```

Figure 16. Adding a student

2.6.5 Starting the server

As soon as the students are added, the tutor starts the server; the server is set to wait for incoming connections.



```
C:\Windows\System32\cmd.exe - server.py
Microsoft Windows [Version 10.0.17134.950]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ILyes\Desktop\CBA tlemcen>server.py
*****
*****C.B.A TLEMCEN*****
*****
Server Started.
```

Figure 17. Starting the server

2.6.6 Student connection

With the availability of a network between the server and the students' machines, students can open the client app to connect and log in.

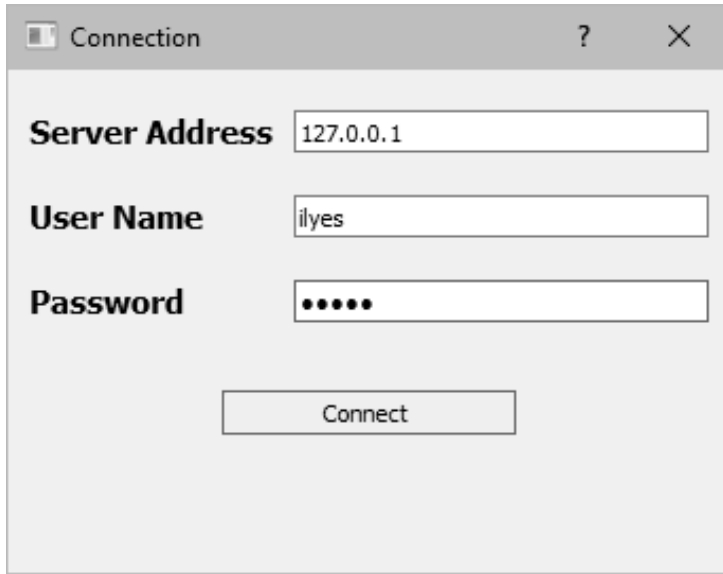


Figure 18. Student connection GUI

After clicking connect, the username and password are sent to the server address, after accepting the new connection, the server verify the username and password, if they match the server notify the tutor of a new user.

```
C:\Windows\System32\cmd.exe - server.py
Microsoft Windows [Version 10.0.17134.950]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ILyes\Desktop\CBA tlemcen>server.py
*****
*****C.B.A TLEMCEN*****
*****
Server Started.
Connection from ('127.0.0.1', 10790) has been established.
user ilyes has connected.
```

Figure 19. Server notification (user connected)

2.6.7 The student's GUI

If the user is registered, a GUI on the student's side will pop up. It gives the ability to upload, delete and execute files "safely" on the server machine, and subsequently the output is displayed on the output text widget.

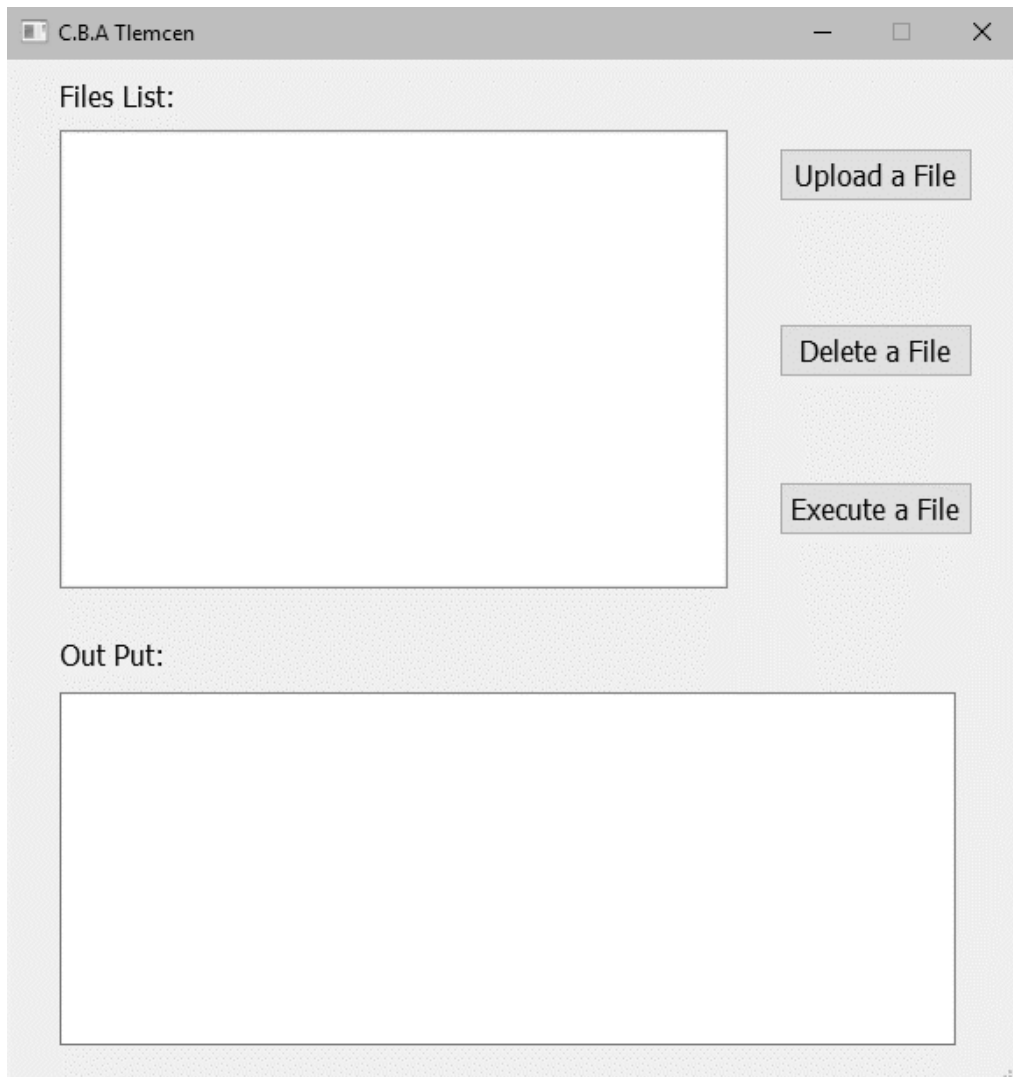


Figure 20. CBA Tlemcen GUI

2.6.8 Uploading files

The student can click on “Upload a File” button to send files to the server. A file dialog will appear and lets the student choose a file. The student has the ability to upload any file regardless of the extension.

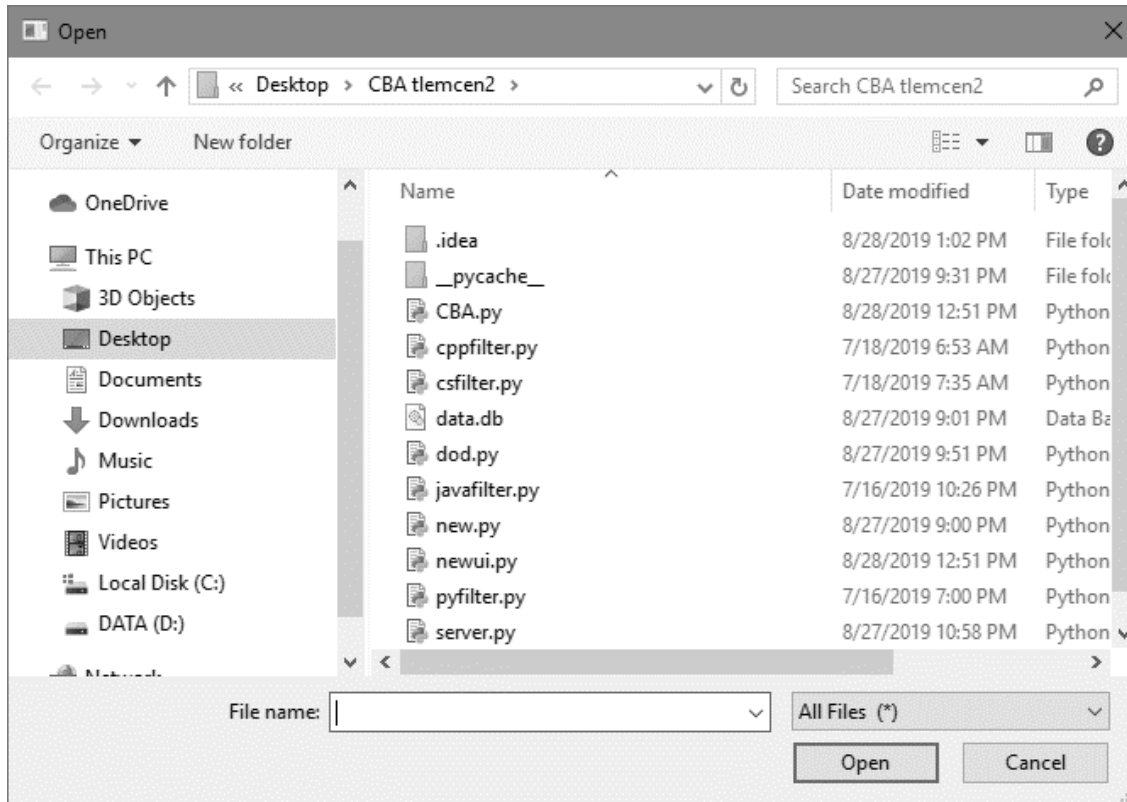


Figure 21. Choosing a file

When the file is successfully uploaded to the server, the files path will appear in the files list, and on the server side. Afterwards, the tutor is notified by a message.

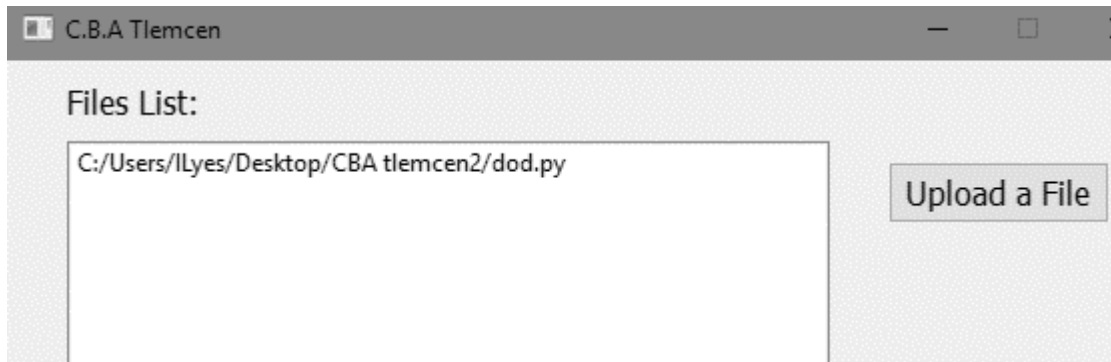


Figure 22. File added to the file list

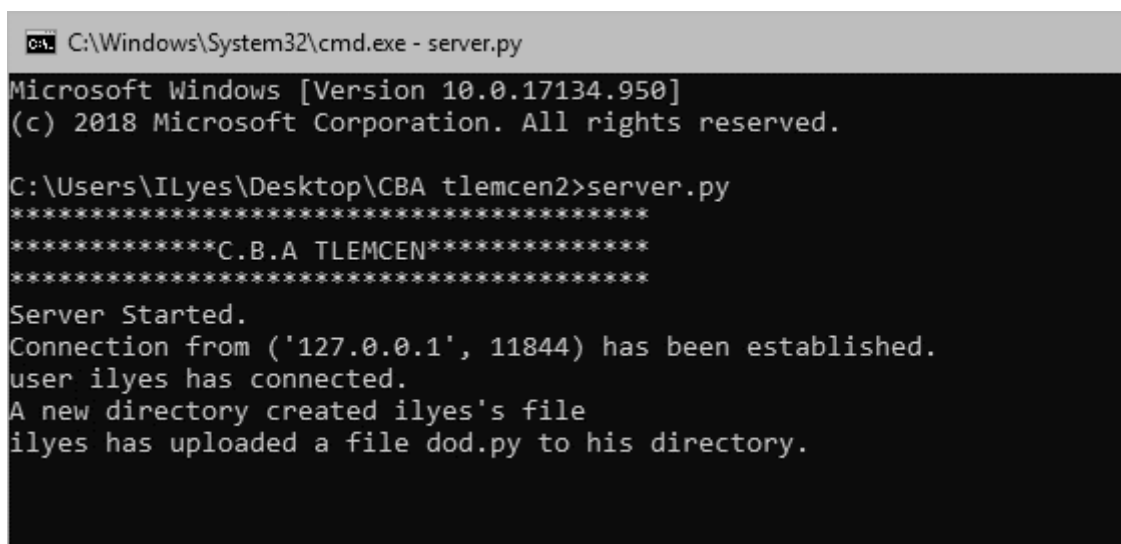


Figure 23. Server notification (file uploaded)

2.6.9 Executing a file

This CBA system supports five programming languages. Consequently, the students can execute five types of files extensions, which are (.PY, .C, .CPP, .CS, and .JAVA). Other files can be libraries, auxiliaries, etc. To execute a file, the student simply needs to select a file from the files list, and click “Execute a File” button. A series of internal consecutive actions will take place at the server, after that an output message will appear on the student GUI. Lastly, the tutor is notified by a message describing the operation done by the student.

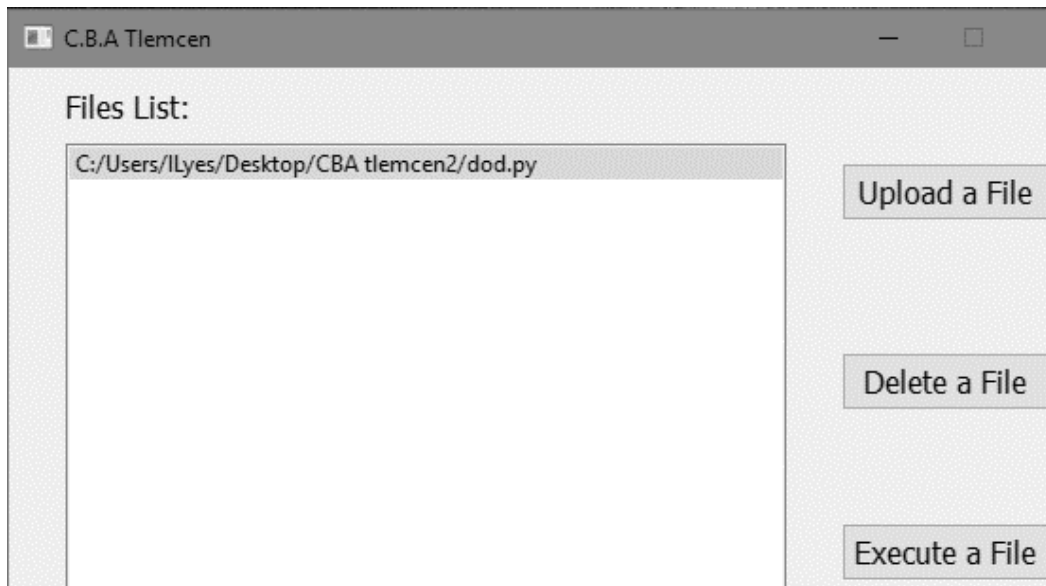


Figure 24. Executing a file

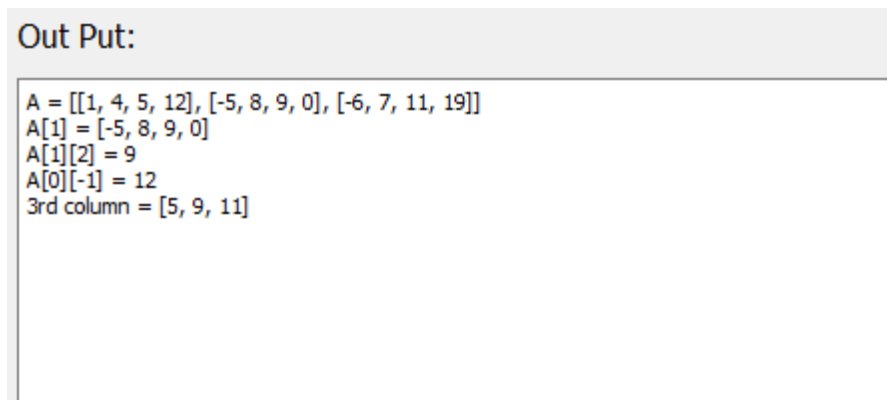


Figure 25. Example of output

```
C:\Windows\System32\cmd.exe - server.py
Microsoft Windows [Version 10.0.17134.950]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Ilyes\Desktop\CBA tlemcen2>server.py
*****
*****C.B.A TLEMCEN*****
*****
Server Started.
Connection from ('127.0.0.1', 12202) has been established.
user ilyes has connected.
A new directory created ilyes's file
ilyes has uploaded a file dod.py to his directory.
ilyes has executed dod.pythe output file is stored in his directory
delete
A new directory created ilyes's file
ilyes has uploaded a file dod.py to his directory.
ilyes has executed dod.pythe output file is stored in his directory
```

Figure 26. Server notification (executing a file)

2.6.10 Malicious code check

Remote execution protocols often face security issues. Most of these issues are malicious code injection. As an example, a student will upload and execute a file that can compromise the server. Fortunately, this case was discovered early in the development phase. Hence, the proposed CBA system checks every file against a filter before executing it. The filters are specified for each extension and contains a list of dangerous instructions for each language. For instance, students program should not access to the file system in the server, system commands, networking, etc.

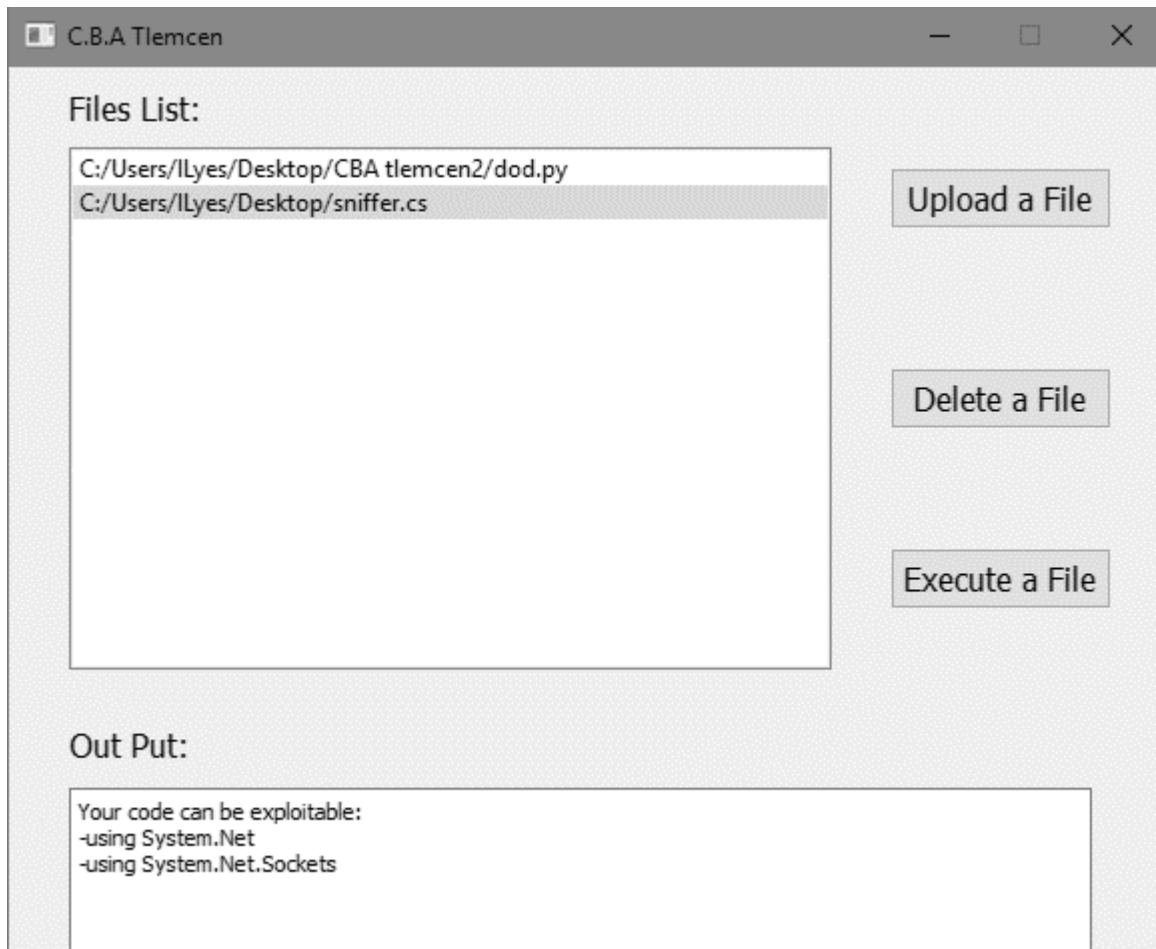


Figure 27. Malicious code detected

2.6.11 Deleting files

The student can delete a file by selecting it and pressing “Delete a File” button, the file will be deleted from the server in the student’s directory, and will disappear from the files list.

```

C:\Windows\System32\cmd.exe - server.py
Microsoft Windows [Version 10.0.17134.950]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ILyes\Desktop\CBA tlemcen2>server.py
*****
*****C.B.A TLEMCEN*****
*****
Server Started.
Connection from ('127.0.0.1', 2306) has been established.
user ilyes has connected.
A new directory created ilyes's file
ilyes has uploaded a file new.py to his directory.
ilyes has deleted new.py from his directory.

```

Figure 28. Server notification (deleting a file)

2.6.12 Assessment

The tutor can access at any given time the students' directories, as soon as a file is executed. An output text file is saved with it and the tutor can access and monitor all the students' work.

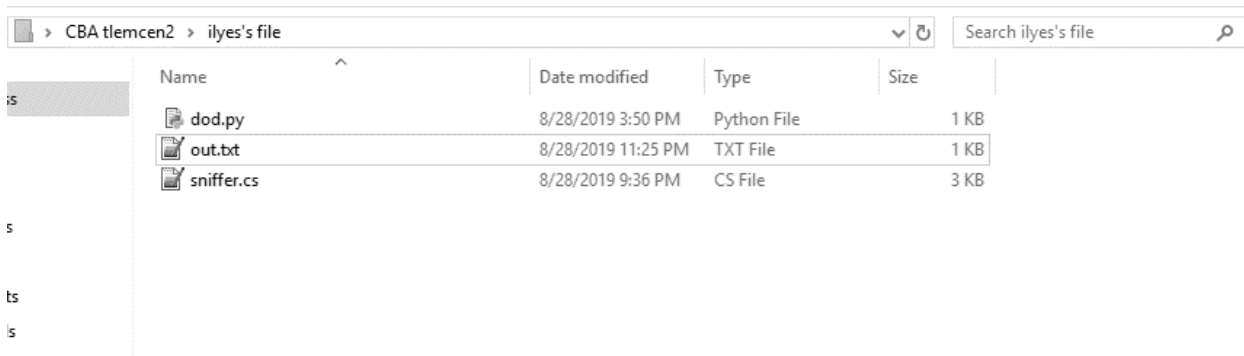


Figure 29. Student's files

2.7 Conclusion

To summarize, this prototype has handled several issues related to using a CBA system for programming education. However, improvement could be done on the server deployment. A safe clustered virtual environment for running each student's code is certainly more interactive and evolved solution. The student's GUI has only the essential components whilst a real time chat box with the tutor seems an appealing way to provide better guidance.

General conclusion

The goals of this work were to specify and develop a prototype of a CBA system to support the programming courses in the University of Tlemcen. The system should be able to help reduce the amount of work needed for marking and grading programming assignments, as well as supervising students' work in the classroom, which was made possible by making a remote file execution protocol. It has a two core components (client/server), behaving as an independent services, responsible for file exchange and running then returning the results to the end user. In order to achieve the proposed goals, the following steps were followed:

- A study of four state-of-the-art CBA systems was performed, with the goal of understanding what features are available. The features were grouped in five/six different topics for a better comprehension. The studied systems were firstly presented one by one and then their features were summarized.
- The learning needs of UABT's teaching staff were analyzed ,also needs from students' prospective were looked into in details.
- Then, a specification for the new CBA system was developed. It has a proposal containing use case diagrams and a sequence diagram.
- Having the complete system specification, a prototype of the CBA system was developed. It implements a subset of the specified functionalities and consists of a server that can download and execute students' files in many programming languages, plus a client side to send requests to the server. The server supports the execution of most programming languages used in academic institutions (Java, C, C++, python, C#).
- Several improvements can be done to this work, such as the fully automatic assessment of students' code, which should be taken into account in a future work.

References

[I1] :Summons P., Coldwell J., Bruff C., Henskens F., Automated assessment and marking of spreadsheet concepts, Proceedings of the Second Australasian Conference on Computer Science Education, Sydney, Australia, 3-5 July, 1996, Pages 178-184

[I2] :Canup M., Shackelford R., Using software to solve problems in large computing courses, Proceedings of the 29th SIGCSE, Technical Symposium on Computer Science Education, Atlanta, USA, 1998, Pages 135-139

[I3] : Pfleeger S. L., An Investigation of Cost and Productivity for Object-Oriented Development, Ph.D. dissertation, Georgia Mason University, Fairfax, Virginia, 1989

[I4] : Greening T., Pedagogically sound responses to economic rationalism, Proceedings of the 31st SIGCSE technical symposium on Computer Science Education, Austin, TX USA, 7-12 March, 2000, Pages 149-156

[I5] : Lee S, Development of instructional strategy of computer application software for group instruction, Computers and Education, Volume 37, Issue 1, 2001

[II1] : Christopher Douce, David Livingstone, and James Orwell. Automatic test-based assessment of programming: A review. *Journal on Educational Resources in Computing (JERIC)*, v.5 n.3, p.4-es, September 2005.

[II2] : UVA Online Judge, 2010. <http://uva.onlinejudge.org/>. Last accessed: August 2nd, 2019.

[II3] : COLIN HIGGINS *, TAREK HEGAZY, PAVLOS SYMEONIDIS and ATHANASIOS TSINTSIFAS School of Computer Science and Information Technology, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK

[II4] : Colin A. Higgins, Geoffrey Gray, Pavlos Symeonidis, and Athanasios Tsintsifas. Automated assessment and experiences of teaching programming. *Journal on Educational Resources in Computing (JERIC)*, v.5 n.3, p.5-es, September 2005.

[I15] : Colin Higgins, Pavlos Symeonidis, and Athanasios Tsintsifas. The marking system for CourseMaster. *Proceedings of the 7th annual conference on Innovation and technology in computer science education*, Aarhus, Denmark, June 24-28, 2002.

[I16] : COLIN HIGGINS *, TAREK HEGAZY, PAVLOS SYMEONIDIS and ATHANASIOS TSINTSIFAS School of Computer Science and Information Technology, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK- page 298 3.4. Exercise parameterization

[I17] : Pedro Xavier Pacheco, Computer-Based Assessment System for e-Learning applied to Programming Education, FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO page 7

[I18] : Colin Higgins, Pavlos Symeonidis, and Athanasios Tsintsifas. The marking system for CourseMaster. *Proceedings of the 7th annual conference on Innovation and technology in computer science education*, Aarhus, Denmark, June 24-28, 2002.

[I19] : COLIN HIGGINS *, TAREK HEGAZY, PAVLOS SYMEONIDIS and ATHANASIOS TSINTSIFAS School of Computer Science and Information Technology, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK- page 298 3.3. Feedback detail and grading styles

[I110] : Colin Higgins, Tarek Hegazy, Pavlos Symeonidis, and Athanasios Tsintsifas. The CourseMarker CBA System: Improvements over Ceilidh. *Education and Information Technologies*, v.8 n.3, p.287-304, September 2003.

[I111] : Colin A. Higgins, Geoffrey Gray, Pavlos Symeonidis, and Athanasios Tsintsifas. Automated assessment and experiences of teaching programming. *Journal on Educational Resources in Computing (JERIC)*, v.5 n.3, p.5-es, September 2005.

[I112] : Mike Joy and Michael Luck. On-line submission and testing of programming assignments. *Innovations in Computing Teaching*, SEDA, 1995.

[I113] : M. Joy and M. Luck. Plagiarism in Programming Assignments. *IEEE Transactions on Education*, 42(1):129–133, 1999.

[II14] : Mike Joy, Nathan Griffiths, and Russell Boyatt. The boss online submission and assessment system. *Journal on Educational Resources in Computing*,5(3):2, 2005.

[II15] : ACM Journal on Educational Resources in Computing, Vol. 5, No. 3, September 2005.

[II16] : Computer-Based Assessment System for e-Learning applied to Programming Education

Pedro Xavier Pacheco July 2010 p:10

[II17] : Joachim Schwierien, Gottfried Vossen, and Peter Westerkamp. Using Software Testing Techniques for Efficient Handling of Programming Exercises in an E-Learning Platform. *Electronic Journal of e-Learning*, 4(1):88, March 2006.

[II18] : Bodo Hüsemann, Jens Lechtenböcker, Gottfried Vossen, and Peter Westerkamp. XLX - A Platform for Graduate-Level Exercises. Proceedings of the International Conference on Computers in Education, p.1262, December 03-06, 2002.

[II19] : José Paulo Leal. Mooshak, 26 May 2009 2009. <http://mooshak.dcc.fc.up.pt/>. Last accessed: August 5, 2019.

[II20] : José Paulo Leal and Fernando Silva. Mooshak: a Web-based multi-site programming contest system. *Software—Practice and Experience*, 33(6):567– 581, 2003.

[III1] : Kuhlman, Dave. "A Python Book: Beginning Python, Advanced Python, and Python Exercises".

[III2] : "Guido van Rossum Stepping Down from Role as Python's Benevolent Dictator For Life | Linux Journal". linuxjournal.com. July 4th 2019

[III3] : "Python boss Guido van Rossum steps down after 30 years". The Inquirer.

[III4] : " python.org/dev/peps/pep-8100/" Python Software Foundation. July 4th 2019.

[III5] : B.M. Harwani "Qt5 Python GUI Programming Cookbook" ISBN 978-1-78883-100-0

[III6] : "doc.qt.io" QT designer manual July 4th 2019

[III7] : Haagsman, Ernst (4 April 2019). "Collaboration with Anaconda, Inc". PyCharm Blog. Last accessed July 4th 2019.

[III8] : Python software foundation "docs.python.org/3/library/sqlite3.html" last accessed July 4th 2019

List of figures

Figure 1A high level view of CourseMarker main parts.....	15
Figure 2CourseMarker student interface: the student view for a course.	17
Figure 3CourseMarker student interface: exercise results for a student.....	18
Figure 4The assessment process.....	19
Figure 5BOSS architecture	22
Figure 6BOSS student interface - Java client	23
Figure 7BOSS web interface.....	23
Figure 8xLx GUI for a tutor to configure new exercises.	25
Figure 9 xlx consultation window.....	27
Figure 10Mooshak graphical interface tutor's view.....	30
Figure 11 client/server architecture.....	36
Figure 12 Use-case diagram.....	39
Figure 13 Sequence diagram.....	40
Figure 14 Windows 10 advanced system properties	42
Figure 15 Environment variables	43
Figure 16 Adding a student.....	44
Figure 17 Starting the server.....	44
Figure 18 Student connection GUI	45
Figure 19 Server notification (user connected).....	45
Figure 20 CBA Tlemcen GUI	46
Figure 21choosing a file	47
Figure 22 File added to the file list.....	48
Figure 23 Server notification (file uploaded)	48
Figure 24 Executing a file	49
Figure 25 Example of output.....	49
Figure 26 Server notification (executing a file).....	50
Figure 27 Malicious code detected	51
Figure 28 Server notification (deleting a file)	52
Figure 29 Student's files.....	52