UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

PROJECT TITLE: AN ONLINE CODE ASSESSMENT SYSTEM FOR VISUAL BASIC.NET PROGRAMS

CASE STUDY OF A LEARNING INSTITUTION IN KENYA

BY

MUKUNGA CATHERINE WAMBUI

P58/76360/2012

SUPERVISOR: DR. KAHONGE MWAURA ANDREW

A RESEARCH PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT OF MSC COMPUTER SCIENCE

DECLARATION

This project as presented in this report is my original work and has not been presented for any other University award.

STUDENT:

DATE:

SIGNATURE:

This work has been submitted as part of the fulfilment of the requirements for Master of Science in Computer Science at the University of Nairobi with the approval of the University Supervisor.

SUPERVISOR: Dr.Andrew Mwaura

DATE:

SIGNATURE:

ACKNOWLEDGEMENT

I take this opportunity to express my heartfelt gratitude to all those who participated in one way or another in making this project a success.

The list is long but I would like to single out the following that played a major role in aiding this work: My supervisor Dr.Andrew Mwaura for his guidance in all stages of the project, panel members Dr.chepken, Dr.Tonny Omwansa, Ms.Pauline Wangunyu and my colleagues in the class of 2012.

Thank you and may God bless you.

DEDICATION.

I dedicate this project to my husband Anthony and my daughter Lauryn who have been my source of encouragement. Thank you for believing in me and being so understanding. To my parents Mr.and Mrs Mukunga and my sister Faith Mukunga for your prayers and words of encouragement.

ABSTRACT

Many Kenyan learning institutions offer ICT training and computer programming is one of the key courses. The programming course with the highest number of students in this institution is Visual basic.net.Currently; the instructors in the institution are forced to set questions in multiple choice format to make their work easier when it comes to marking. This applies to programming examinations and has greatly affected the students' performance negatively. The multiple choice questions do not test the coding skills of the student and neither is the student's programming skill improved because most of them guess the answers.

The main objective of this project was to develop an online code assessment system capable of assessing correctness of visual basic.net programs and providing instant feedback. This was implemented at the learning institution by the programming students taking vb.net course at basic, intermediate and expert levels. The software development life cycle (SDLC) methodology was used in the development of the proposed system and case study research design was used to conduct research.

The online code assessment system was tested using various testing strategies to ensure that it was working correctly. System effectiveness testing results showed that over 80% of the students and instructors found the system to be effective on exam marking, score computation and feedback. Usability testing was conducted and 93.1% of the students, 100% of the instructors and 66% of the administrators accepted to use the system. Exam marking was carried out using character matching strategy which is one of the assessment methods under static analysis. Students' answers were marked manually and the results compared to those generated by the system. The results were analysed and the difference was less that 0.5%.The conclusion was that the system was reliable and had acceptable accuracy levels in code assessment since the difference in the manual results and system results was very minimal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DEFINITION OF TERMS

TRAKLA – A system for teaching algorithms using email and a graphical editor (1993).

JPlag-JPlag is a plagiarism detection tool aiming to detect similarities among source code files

JUnit- A unit testing framework for the Java programming language.

Flash Mx tool. –Tool for designing motion graphics or building data-driven applications.

JXXX- A remote compiling service for the Java programming language.

LIST OF ABBREVIATIONS

AA-Automatic Assessment

MCQs-Multiple Choice Questions

DJGPP-DJ's GNU Programming Platform

XML-Extensible Markup Language

RKR-GST-Running Karp-Rabin Greedy String Tiling

VHDL-Very High-speed integrated circuit Hardware Description Language

VM-Virtual Machines

APIs-Application Programming Interfaces

JDBC-Java Database Connectivity

SML-Standard Meta Language

LCS- Longest Common Subsequence

YAP- Yet another Plague

SUS-System Usability Scale

APAS-Automatic Programming Assessment System

CHAPTER ONE: INTRODUCTION

Assessment provides the teacher with a feedback channel that shows how learning goals are being met. It also ensures for an outside observer that students achieve those learning goals. Assessment provides both means to guide student's learning and feedback for both the learner and the teacher about the learning process from the level of a whole course down to a single student on the specific topic being assessed. Students often direct their efforts based on what is assessed and how it affects the final course grade. Continuous assessment during a programming course ensures that students get enough practice as well as feedback on the quality of their solutions. Providing quality assessment manually for even a small class means that feedback cannot be as instant as in one-to-one tutoring. When the class size grows, the amount of assessed work has to be cut down or rationalized in some other way. Automatic assessment (AA), however, allows instant feedback without the need to reduce exercises.

In recent years, developments in information and communication technologies (ICT) have led to growth in the range of Internet tools that can be used for learning and research. Some have gained wide-scale acceptance (e.g., the ease with which e-mail has been adopted); others seem to find either niche applications or are less pervasive than one might at first have imagined (e.g., videoconferencing). One application that is becoming more common is computer-assisted assessment.

The term computer-assisted assessment can cover any kind of computer use in the process of assessing knowledge, skills, and abilities of individuals.

For many reasons, the use of computer-assisted assessment (CAA) is increasing. Assessment is a critical exercise at the end of every learning process. Formative assessment is a valuable tool in helping students understand what they have achieved and in guiding them to further achievement.

The role of assessment in higher institutions of learning has become far more diverse and open to scrutiny in recent years.

Earlier research has shown a range of motivations for implementing Computer assisted assessment in a course, and often a combination of factors result in Computer assisted assessment being used (Bull & McKenna, 2001).

Some of the key reasons cited include: To increase the frequency of assessment, motivating students to learn and encouraging skills practice, broadening the range of knowledge assessed, extending the range of assessment methods, increasing objectivity and consistency and reduce marking loads and to aid administrative efficiency.

1.1 Background of the problem

The number of students enrolling in programming courses at the learning institution has increased substantially over the past few years, leading to large class sizes and increased student-staff ratios.

A specific problem arising from this is the substantial resources required to manage the assessment of practical exercises, so that students receive accurate and timely feedback which can benefit their progress.

Although a lot of effort has been put into the study of varying forms of online assessment, very little has been done on environments that support automated assessment of the ability to write and debug programs. The most common mode of assessment is through assignment work where demonstrators and tutors give students the opportunity to improve their programming skills and knowledge in a supportive environment.In courses where one of the primary goals is to produce competent programmers, then it is reasonable to expect examinations to contain programming tasks that can be solved in an environment that is similar to the "normal" program development environment.

Previous studies indicate that multiple choice examinations are not the best way to assess programming skills as most students can guess the right answer. In addition, most systems assess basic programming skills using multiple choice questions and this leaves advanced programmers out. Java and C++ are the programming Languages mostly supported by existing online assessment systems.

1.2 Problem statement

At the learning institution, the number of students has grown tremendously hence the instructors are not able to go through student's written code when marking examinations. The instructors set the questions using multiple choice format which is easier for them to mark and give results on time. This has resulted in the students having poor programming skills. The instructor is not able to assess how well the students understand the programming language in terms of practical programming skills. Whenever programming students are sent out for internship by the institution, there have been many complaints from employers expressing their disappointment in those students because they cannot develop good systems.

This research addresses this problem through the development of an online programming assessment system that can assess Visual basic.net programs in all student levels (basic, intermediate and expert).

1.3 Objectives

1. To conduct a research on existing online assessment programming systems and analyse their features and functionality.
2. To develop an online assessment system that is able to assess correctness of visual basic.net programs and provide instant feedback.

3. To use a set of test cases to test the developed online assessment system to determine whether it works correctly.

1.4 Research questions

The research was guided by the following research questions.

1. What are the existing types of online assessment systems for programming exams?
2. What are the limitations of multiple choice format questions in programming examinations?
3. What are the effects of implementing a code assessment system in a learning institution that offers software development training?
4. Which methods of assessment exist in current code assessment systems?

1.5 Justification

The online assessment system for visual basic programs is very useful to the programming students. Students' deep understanding of the programming language can be assessed by the system. This has reduced the number of complaints from employers who are disappointed by the student's poor programming skills.

1.6 Project scope

This study is limited to three specific objectives that focus on determining the advantages of using an online code assessment system for assessing programs as opposed to using multiple choice format questions during programming examinations. The development of this online code assessment system for visual basic.net programs has been of great benefit to the institution. The system can assess student's code and provide immediate feedback to the students.

The system has been tested using test cases to evaluate its performance.

1.7 Assumptions

The assumption is that every person using the system has basic programming knowledge.

1.8 Limitations

The system is restricted to assessment of one programming language (visual basic.net).

CHAPTER TWO: LITERATURE REVIEW

This chapter gives a brief introduction of programming and online assessment concepts and review of various features of existing online assessment systems for programming assessments.

2.1 Computer Programming

Computer programming is a creative skill, requiring "deep" learning. The student is required to practice in order to master. Existing generic tools do not address such skills, and although there is substantial literature defining best practice for use of such tools (Bull and McKenna 2001), it has been argued that simple questions cannot be used to measure deep learning (Entwistle 2001). Computer programs are, in principle, ideal subjects for automated assessment. Not only can the correctness of a program be measured, but also its quality, by the application of metrics. Due to the regularity of program code, techniques for plagiarism detection can be easily incorporated into the automated process.

Programming comprises a broad scientific field that demands not just theoretical knowledge, but also deep understanding of the framework of Structured Programming. Additionally, students need to have a deep understanding of the syntax of the language they are called upon to learn, in order to practice. People involved in Programming appreciate that the Science of Programming requires perfect handling of the logic behind the idea, rather than ability of memorizing the syntax of different languages.

It is not unusual that several students, upon completing a year of study on Programming, exhibit serious shortcomings on basic Programming knowledge (McCracken et al., 2001). It was found that students with little or no practical work were able to produce a piece of code in the final traditional way of assessment through memorization and achieve a "good" grade in the course (Woit & Mason, 2003). It is difficult to closely observe the progress of a particular student, especially in large classes. This happens because there is not enough available time for the teacher to interact personally with every student.

Teaching and learning Programming has created significant difficulties to both teachers and students (Wang & Wong, 2008). Innovative ways are needed in order to improve the effectiveness of teaching Programming. Assessing the students' programming knowledge using computers in a regular and continuous basis could help. The assessment results could be used for continuous improvement of teaching effectiveness and learning quality (Khamis et al, 2008).

2.2 The Assessment Process.

The process of marking a programming assignment includes three principle components. The first component, correctness, relates to the extent to which a program's functionality matches that of its specification. The second, which we refer to as style, describes those attributes of a program that a

student's submission is expected to display, but which are unlikely to be explicit in the program specification, and allow for a limited amount of interpretation. The final component, authenticity, covers administrative tasks including verification of the student's identity and checks for plagiarism. These components are conceptual rather than definitive. There are categories for marking which can be included in the program specification, or can be regarded as stylistic. For example, the performance characteristics of a program may be formally specified, and can thus be checked for correctness, but may alternatively be considered as optional (but desirable) program attributes.

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.

An assessment system is needed to support academics in the assessment of student submissions through collecting submissions, performing automatic tests on them, and providing an interface for marking and delivering feedback.

2.3 Concept of Online Assessment

Online assessment is the process used to measure certain aspects of information for a set purpose where the assessment is delivered via a computer connected to a network. Most often the assessment is some type of educational test. Different types of online assessments contain elements of one or more of the following components, depending on the assessment's purpose: formative, diagnostic, or summative. Instant and detailed feedback, as well as flexibility of location and time, are just two of the many benefits associated with online assessments. There are many resources available that provide online assessments, some free of charge and others that charge fees or require a membership.

The assessment should be carefully designed according to pedagogical theories. (Lister&Leaney, 2003a) encouraged teachers to design assignments according to the cognitive levels defined in the Taxonomy of Educational Objectives (Bloom, 1956).

These levels are the following (from lowest to highest): Recall of data, Comprehension, Application, Analysis, Synthesis and Evaluation.

It is difficult to categorize a question into the proper cognitive level (Thomson et al., 2008). Bloom's Taxonomy can be also used in the course design (Scott, 2003). (Oliver & Dobele, 2007) argued that the lower cognitive levels (Recall of data, Comprehension, and Application) should be gained during the first year of studies. Subsequently, the students could become able to move onto assessments that require higher cognitive levels (Analysis, Synthesis and Evaluation). Otherwise, the assessment will have a negative effect on students to make "upward progress".

2.4 Factors contributing to successful assessment of programming assignments.

2.4.1 Quality assignments

The quality of the assignments is equally important for any course and massive open online courses (MOOCs) are no exception. (Feldman & Zelenski, 1996) state that first-rate homework assignments are integral to the success of courses. (Hundley & Britt, 2009) remark that an important part of a successful course is good assignments. When assignments are assessed manually, it is usually possible to compensate for poor assignment design by giving credit for creativity of solutions while assessing. This is, however, not always the case when automatic assessment is applied. According to (Ala-Mutka, 2005), the use of automatic tools increases the need for careful pedagogical design of the assignment and assessment settings. Thus, the importance of quality assignments is very important.

2.4.2 Clear formulation of tasks

(Hollingsworth, 1960) already realised that assignments need to be properly formulated in order for automatic assessment to be effective. (Douce et al, 2005) remarks that the specification of requirements for an automated assessment always needs to be more precise than for the equivalent manually assessed assignment. Additional care has to be taken to avoid ambiguities. If the specification is ambiguous it allows for different interpretations. Consequently, it is likely that some valid solutions by students may be rejected by the automatic assessment program simply because the assessment instructions may not be configured to recognise it. This observation is a consistent theme mentioned by most authors who have used automatic assessment tools. (Ala-Mutka, 2005) cautions that no ambiguities are allowed in the problem specification, especially when considering input/output formats. Formulating the requirements for assignments that are subjected to automatic assessment has to be thorough, as (Cerioli and Cinelli, 2008) point out.

2.4.3 Well-chosen test data

In most systems applying automated assessment, the functionality of a program is tested by running the program against several test data sets. (Ala-Mutka, 2005) points out that the coverage of the assessment depends on the test case design. The accuracy, and consequently the formative value of the assessment, is highly dependent on the design of the test cases it uses.(Vujoˇsevi´c-Janiˇci´c et al,2012) remark that the grading is directly influenced by the choice of test cases. (Montoya-Dato et al, 2009) point out that the set of test cases needs to be “well thought out” to prevent wrong programs from passing test runs. When wrong programs are falsely identified as correct the students who created the erroneous solutions may remain unaware of their failure, which may have a negative impact on their level of mastery of the material.

2.4.4 Good feedback

The importance of the feedback aspect of formative assessment is very important. (Carless et al, 2011) describe feedback as a key ingredient of the development of quality student learning. (Ahoniemi & Reinikainen, 2006) remark that novice students require profound and personalised feedback on their programming assignments to support them to improve their weaknesses. Feedback on assignments allows students to revise their submissions (Malmi et al, 2002). When students know what exactly went wrong with their submissions and where their programs failed, they can use the information to learn from their mistakes.

2.5 Related Works

Today there are many tools used in the process of assessing programming assessments, ranging from simple tools used only at universities and faculties at which they were developed to the commercial projects that are used in a number of institutions around the world.

The above mentioned tools can be divided into two main categories: Automatic assessment systems and online compilers

2.5.1 Automatic assessment systems

Computerized assessment offers speed, availability, consistency and objectivity of the assessment (Ala-Mutka, 2005). The field of automatic evaluation is huge and there are two different categorizations of existing systems. Carter et al. Divided the exercises into three basic categories: Multiple choice questions, Programming assignments and Visual answers.

2.5.1.1 Multiple choice questions

Multiple choice questions are the simplest form of automatic assessment in which the assessment procedure is frequently embedded into the questions themselves. The most common form of a multiple choice question has four or more alternatives, of which at least one is correct.

However, the number of correct, semi-correct and incorrect choices can vary, depending on the teachers' choice. Typically a student is rewarded with points for the correct and semi-correct choice, while an incorrect answer gives either zero or a negative number of points. The assessment procedure of multiple choice questions is very easy.

It compares the student's answer to the correct one and according to the grading formula gives points. The simplicity of multiple choice questions has made them a very popular feature in learning management systems, such as Moodle and WebCT.

(Whalley et al., 2006) showed that novice programmers were not yet able to work at fully “abstract level” (high cognitive level). So, students that cannot read a short piece of code and describe it are not capable intellectually to write code by themselves. Thus, it is better to assess novice programmers using MCQs. On the other hand, if the students are at an advanced level and the

course focus is on developing the students' programming skills then it is better to use Code Writing Assessment.

According to (Lister, 2005), assessment through MCQs can be effectively administered to beginner programmers who have acquired basic skills. If a student scores poorly or averagely on basic skills, s/he is bound to fail on final exams, which are comparatively more demanding and require more knowledge.

(Denenberg, 1981) stressed the need that evaluation results, questioning and structure must all be based on quality; otherwise the assessment results are of little value.

MCQs comprise a reliable evaluation method, not only in the theoretical field of Information Science but also in Programming. In addition, the test's complexity could be increased by increasing the number of suggested answers or by the addition of short-length answer questions.

More specifically, (Denenberg, 1981) suggests that students should be able to: Read a program (e.g. find the output of the program), Read a logical diagram (comprehension of its flows and operations), Convert a logical diagram to a code and write a program (e.g. find commands from missing code).

2.5.1.2 Programming assignments

The automatic assessment of programming assignments is the most usual example of Automatic assessments in the field of computer science. This category includes all systems that automatically assess some or all aspects of computer programs. The earliest assessment systems, often referred as grading programs were based on very simple output matching method: the output created by a teacher model.

Program was compared to the output of the student program. Today, assessment systems such as ASSYST have ability to evaluate student submissions in different areas: Complexity, Correctness, Efficiency, Style and Testing data adequacy.

A more sophisticated method for program assessment allows analysis of the internal structure of the student's submissions. The early work in this area was focused on estimating the execution time of each program block. More recent examples include the use of abstract syntax trees in order to determine whether a submission contains the required programming constructs. In addition, there are systems such as Ceilidh (later Course Master) that allow several different types of assessment: Complexity analysis, Dynamic correctness, Dynamic efficiency, Structural weakness and Typographical analysis.

The last aspect of the computer program that can be analysed refers to the style of programming which students use while solving a given problem. Programming style assessment is not concerned with the functionality of the program solution, but measures whether the student is capable to follow widely accepted coding conventions (e.g. use comments, code indentation, etc.) and write

understandable program code. Today, there are several automatic systems that include style assessment feature in software development process such as Style++ which allows measurement of 64 different styles during the development of C++ programs.

2.5.2 Online compilers

The online compilers are usually defined as tools that enable online development of the software products. There are several major advantages of this approach. For example, a student does not need to have a compiler installed on his personal computer and may work on the development from any other Internet and browser enabled device. The first and obvious precondition is of cause that the online compiler must support a programming language a student wishes to use in the development or programming process. In addition, another advantage is that this development environment allows students to test their solutions independently to the platform used during the original or online development.

JXXX that compiles java source files including applets, DJ's GNU Programming Platform (DJGPP) which supports C programming language, web 2.0 technology based solution called OLC that supports development of the software products in four different programming languages are some of the current Online compilers .These and other tools have known drawbacks.JXXX compiler could only be used to test already written code, DJ's GNU Programming Platform (DJGPP) provides a simple text editor for writing code that does not support basic coding conventions such as keyword highlighting, text indenting.

Almost every online compiler does not put emphasis on protection from plagiarism and does not provide the mechanisms by which the teacher could be sure that the student actually wrote submitted solution.

There are a few surveys of Automatic Assessments in the context of programming assignments. A Survey of Automated Assessment Approaches for Programming Assignments by Kirsti Ala- Mutka from 2005 concentrates on what features of programming assignments are automatically assessed whereas Douce et al. review the history of the field from 1960s to 2005. One of the main findings by Ala-Mutka is that dynamic analysis which is, assessment based on executing the program is often used to assess functionality, efficiency, and testing skills. Static checks that analyse the program without executing it are used to provide feedback from style, programming errors, software metrics, and even design. There are many features to assess, and Ala-Mutka concludes that the selected automatic assessment approach should always be pedagogically justified.

2.6 Features supported by recent programming assessment systems

2.6.1 Programming languages

A majority of the systems are either targeted only for Java or have support for Java. This fits well with the trend of Java being one of the most used introductory programming languages. Other popular languages supported by the systems include C/C++, Python, and Pascal. Some of the systems are language independent especially if the assessment is based on output comparison. Any language that can be executed on the same environment can be automatically assessed after the system is configured to compile and execute solutions in that language.

2.6.2 Methods of assessment

2.6.2.1 Sandboxing

The student's program is tested by executing the created executable in the sandbox with each of a number of specified test cases. The test suite used for the assessment is designed by the instructor. Detail of the test suite is specified in an XML file that is associated with the assignment.

In general, a sandbox is an isolated computing environment used by software developers to test new programming code. In a Java programming language and development environment, the sandbox is the program area and set of rules that programmers need to use when creating Java code (called an applet) that is sent as part of a page. Since a Java applet is sent automatically as part of the page and can be executed as soon as it arrives, the applet can easily do harm either accidentally or as the result of malicious intent if it is allowed unlimited access to memory and operating system services. The sandbox restrictions provide strict limitations on what system resources the applet can request or access. Essentially, the programmer must write code that "plays" only within the sandbox. Since the programming assignments are typically graded by running the students' solutions on the server side, securing the server against possibly malicious or just incorrect code is important.

The following approaches to secure execution of students' code have been mentioned in previous studies.

i. **Proper sandboxing**. Relying on existing solutions to securely run programs is a common approach. This can be done by using multiple tools like Systrace (used in EduComponents) Linux security module, Java security policy, ptrace, and chroot.

ii. **Static analysis**. Security can also be addressed by using custom solutions. For example, Algorithm Benchmark uses regular expressions to filter out malicious code.

iii. **Grading on the client**. Some systems deal with sandboxing by doing the grading on the client side in students' own computers.

Additional security feature implemented in some systems is to have a different server for running the student programs instead of doing it all on the same machine as the rest of the system. This is done, for example, by EduComponents. In addition to securing automatic assessment systems, sandboxing can help when assessing the performance of students' programs. Sandboxes can be configured to limit the available memory or CPU time to ensure assessed solutions are efficient enough.

2.6.2.2 Non-structural Similarity Analysis

The system is capable of comparing programming source code submitted by the student with the answer schemes provided by the instructor. The instructors need to provide more than one answer scheme in order to facilitate all the possible answer variation by the students. The student's answer will be compared to all of the answer schemes provided. The highest mark from the comparison will be the mark for the students answer.

2.6.2.3 Abstract syntax trees

More recent examples include the use of abstract syntax trees in order to determine whether a submission contains the required programming constructs.

2.6.2.4 Visual answers

A student manipulates visualization in order to develop a solution of a given programming task. In addition, visualization can be used for learning basic programming concepts, particularly data structures and algorithms. The main representative of this group of Automatic Assessment is TRAKLA system that, by using various heuristics, compares the model answer to the student's submission and can thus detect some simple errors in the final stage. On the other hand, its successor, TRAKLA2 is based on generalized assessment procedure which compares submitted solution of whole simulated algorithm to teacher model solution and tries to find identical states. Among the other systems which can be placed in this group, are Stratum which can help students understand logic, regular expressions etc., much easier, and Exorciser in which students can solve exercises or learn basics of theoretical computer science through graphical manipulation of the required entities (e.g. strings).

2.6.2.5 Test Runs

Most of the test runs are "fixed"; that is, expressions are given by the teacher and tested with every student. An example can be a Scheme expression (fact 5) returning 120, which is the factorial of 5, calculated using a fact procedure submitted by the student.

2.6.2.6 Keyword Analysis

This kind of assessment is done by searching the keywords used in the students' answers.Scheme-robo is one of the APASes that capable of searching the specific keywords used in the programming answer for problems written in one of a functional language, the Scheme(Saikkonen, 2001). Certain exercises need to be solved using a set of specific functions only determined by the instructor, while certain exercises are prohibited to use a certain function in order to promote more creative solutions from the students

2.6.2.7 Mutation testing

Mutation testing is a method to measure quality of test suites. In mutation testing, original tests are executed against multiple slightly modified programs. These modified programs, also known as mutants are typically generated from the original by applying small, well defined mutation operations to the code (e.g. change comparison operators from > to < or =>). Tests should catch these mutants.

2.6.2.8 Plagiarism Detection

Plagiarism is unacknowledged copying of documents or programs (Joy & Luck, 1999). In programming exercise context, a student is copying another program (from friends, books or internet sources) giving the impression that the work is his/hers. Such activities do happen for whatever reason, and it is a very time consuming task to check each and every single students' exercise in order to detect plagiarism. Luckily there are several projects to detect plagiarism through automated checking. Plague (Whale, 1986) is among the early systems developed to detect plagiarism in Pascal, Prolog, Bourne Shell and Llama. It uses a comparing algorithm (from the variant of Longest Common Subsequence - LCS) that compares the tokens from two sources of programming codes. However Plague did not survive long, because it is very complicated to modify the application to cater for different programming languages (Granville 2002). YAP (Yet another Plague) is another system developed to detect plagiarism (Wise, 1992). In the latest version, YAP3 uses Wise developed text matching algorithm known as RKR-GST the short form for Running-Karp-Rabin Greedy-String-Tiling (Wise, 1993). This algorithm is claimed to produce faster and better text matching result. The same algorithm in YAP3 was improved and implemented in JPlag (Prechelt et. al., 2000). JPlag was developed using Java and by the time of this writing, it supports a list of programming languages such as C, C++, Java, Scheme and etc. MOSS (Measure Of Software Similarity) is another plagiarism detector (Aiken, 2007) for C,C++, Java, C#, Python, Visual Basic, JavaScript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS assembly, a8086assembly, a8086 assembly, MIPS assembly and HCL2.

2.6.2.9 Diagram Analysis

There are special diagram analysers capable of checking the flowchart, object-oriented design and simulating electronic circuits. Such features are available in Course Master (Symeonidis, 1998). The flowcharts will be evaluated by converting the flowchart submitted by the students into a BASIC codes. The BASIC codes are sent to the dynamic tools and executed for dynamic evaluation. For the object oriented design evaluator, the student's design will be tested for completeness, correctness, accuracy, the use of correct relationships between the various components and the use of classes that are needed to complete the design. For the logical circuit simulator, students' logical circuit is sent to the Circuit Sim to be evaluated through a set of circuit simulation processes**.**

2.6.3 Resubmissions

Practice is important in learning programming and there should be room for mistakes and learning from them. Automatic assessment can help as it can give feedback despite the limited human re-sources. However, to prevent mindless trial-and-error problem solving, the number of resubmissions should be controlled.

Examples of how the problem of trial and error can be tackled.

i. Limiting the number of submissions, in addition to having deadlines, is the trivial approach supported by most of the current systems.
ii. Limiting the amount of feedback is another classical way to force students think after a failed submission. However, this can also create confusion among students. Especially, students not familiar with automatic assessment (who do not trust automatic assessment yet) may feel that the feedback provided by the system is erroneous if they are not able to understand why their submission was judged wrong. Compulsory time penalty after each submission can be used to direct students' behaviour. Moreover; length of this penalty can grow exponentially after each failed attempt.
iii. Making each exercise slightly different is an interesting concept that has been used in Quiz PACK by allowing parameterized, automatically assessed random assignments for C programming.

Programming contests provide a completely alternative approach where the assignment specification is visible only for a short period of time during which the assignment needs to be completed while competing against time (and others). This approach is adapted to education, for example, in Mooshak.

Various hybrid approaches and modifications are also possible. For example, Marmoset supports both unlimited and limited number of submissions.

First, there is a public test set to check the basic functionality. These tests can always be executed and repeated submissions are not penalized. Second, there are release tests that can only be asked N-times. Feedback from the release tests is also limited to force students to think before asking tests to be executed.

One central idea in most modern assessment systems that support resubmission is that they allow the student to learn from his mistakes and consequently also practice debugging skills. An adverse side-effect of allowing resubmissions is that students can misuse the assessment system itself using the system as a debugger. Another way of approaching the problem is to teach debugging and testing skills. ASSYST Jackson and Usher (1997) and Web-CAT (Edwards, 2004) both assess the quality of student-provided test cases by measuring test coverage. In Web-CAT it is also possible to withhold test results from instructor tests until a satisfactory coverage has been reached.

2.6.4 Possibility for Manual Assessment

It is often a good idea to combine both manual and automated assessment. Teaching assistants (TAs) can provide extra feedback by manually assessing a submission or they can override the grades, etc. previous studies identify two levels of manual intervention (no support for manual intervention being the third).

To enable the teacher to view the student submissions is the lightest way to support manual intervention. In this approach, the tool itself does not provide any features for the marking but at least makes it possible to manually assess the same submission. Often the same effect can be achieved by logging into the assessment system and fetching the submissions from the database or file system where they are stored. However, supporting this through the automatic assessment system makes it possible to separate the roles of teaching assistants (TAs) from administrators of the automatic assessments system.

Combining manual and automatic feedback means Teaching assistants (TAs) feedback and automated assessment can both exist at the same time and support each other. This is supported in Web-Cat for example. None of the systems clearly describes that they would allow teaching assistants (TAs) to completely override the automatic feedback but some systems are still expected to support this. However, this can easily create confusion among both teachers and learners if the origins of the grade are not transparent.

2.6.5 Distribution and Availability

It is surprising, and quite disappointing, to see how few systems are open-source, or even otherwise (freely) available. In many papers, it is stated that a prototype was developed but it could not be found. In some cases, a system might be mentioned to be open source but you need to contact the authors to get it.

There seems to be a steady interest in developing new automatic assessment tools. Sometimes the need to implement yet another system can be challenged and one should ask whether the new feature could be added directly into an existing open source system as in Web-CAT for example. To increase the adoption of existing systems, Automatic Assessment System developers are encouraged to make their systems open source making it easier for others to contribute.

2.6.6 Specialty

Quite often the driving force for the development of a completely new tool is a revolutionary idea of something that has not yet been done, or at least this is the case with tools that get researched and published.Specialities of automatic assessment systems identified during previous surveys include automatic assessment of GUIs has been identified already in the survey of Douce and is still of interest. New systems are still being developed and the existing ones extended to meet the special requirements of Graphical User Interface exercises and SQL tutoring systems have existed since the late 90's. New systems for this specialty have been recognized in previous surveys.

2.6.7 Security for online programming assessment systems practices.

(Luck and Joy, 1999) analysed security issues on Programming assessment systems covering robust environments, privacy, and data integrity. Security can be handled from ad-hoc solutions to solutions based on Virtual Machines (VM) in order to execute the programs on a safe and controlled environment. Another concern is the increase of plagiarism (Engels, 2007) and (Cheang, 2003). Luck and (Joy, 1999) and (Blumenstein et al, 2004) analyse the integration of plagiarism services in the assessment workflow.

As a major component in online learning, online assessments are important, both to ascertain students' progress and because they can be carried out flexibly in different locations and at different times (Reeves, 2000; Meyer & Zhu, 2013). According to a study carried out by (King, Guyette, and Piotrowski, 2009), 73.6% of students think that it is easier to cheat in an online environment than in a conventional one. Methods of cheating on online assessments include online communication, telecommunication, internet surfing (Rogers, 2006), copying and pasting from online sources (Underwood& Szabo, 2003), obtaining answer keys in an illegitimate way, taking the same assessment several times, and getting unauthorized help (Rowe, 2004).

Other means of cheating on online tests include someone other than the actual student taking the online test and the copying of answers from elsewhere (Sasikumar, 2013).

(Ndume, Tilya, and Twautomatic assessmentkyondo,2008) argue that preventing cheating in online course assessments is much harder than in traditional classrooms and that secure assessment of online courses requires the improvement of system security, the registration of learners with unique identification, and the overall administration of the online assessment. Therefore, improving the

security of online learning will improve the security of online assessments, and this should not be neglected.

Online systems need to authenticate the individual undertaking the assessment some systems have gone as far as taking photographs at random intervals to assure this (Rönnberg, 2001).

Security can also be improved by adopting a proctored or supervised testing environment, where all students are watched by an examiner as they undertake the assessment.

2.7 Review of Existing systems

2.7.1 BOSS programming assessment system

Boss is a tool for the assessment of programming assignments, which supports a variety of assessment styles and strategies, and provides 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.

The "BOSS" Online Submission System has been developed over a number of years, as a tool to facilitate the online submission and subsequent processing of programming assignments (Luck and Joy 1999).

2.7.1.1 Automatic Testing

BOSS evaluates correctness by the application of automatic tests, and two paradigms are currently employed, although the software is structured to allow the incorporation of further testing paradigms in the future.

The first paradigm defines input and output as data files, and a test is constructed by specifying the content of the expected output file (or files) for given input data files. A script (such as a UNIX shell script) may be incorporated to post-process the output generated after a test. Although this is a simple "black box" technique which is common for test harnesses, and is used in other systems such as Course Marker (Higgins et al. 2003), it has the advantage that (almost) any automatic test can be specified in this manner. Furthermore, if the data files are assumed to be text, then each test can be described clearly and concisely, and hence made accessible to students.

The second approach (which only applies when Java is the language used) uses JUnit tests (Lane 2004). In this case, input and output are specified as Java objects, and a test is constructed by specifying a method to be run taking the input object as argument, and returning the expected output object. This paradigm has the advantage of compatibility with development environments which support JUnit testing, and consistency with both classical and agile development methodologies.

Since the automatic tests will run code written by students, there is a danger that a student's program may (accidentally or otherwise) perform an unsafe operation, potentially damaging the

system on which the test is run. The test harness used by BOSS protects the system against unsafe or malicious code, using a variety of security techniques. BOSS delivers these paradigms of testing in two modes. The first mode is available to students at submission time to enable them to gain immediate feedback (and allow them to re-submit in the light of this feedback if they wish). The second mode is post-submission and allows the course manager to run a batch job of tests after the final submission deadline. This second mode is typically linked to marking categories and creates the starting point for the marking process.

The availability of automatic tests both to students, and securely to staff, allows their use either as a formative resource, or for summative evaluation purposes, or as a combination of both.

2.7.1.2 Automatic measurement

BOSS provides a set of simple program metrics, such as number of comments, percentage of methods declared abstract, etc., which can support the marker in assessing the subjective attributes of a program. The software is extensible, and inclusion of other metrics is straightforward.

2.7.1.3 Submission and authentication

A primary administrative function of BOSS is the online collection and storage of work submitted by students. This part of the process requires security features, including; verification of the student using the software, assuring Integrity of files submitted by a student, Transmission of data between student and the system, and the data protection from unauthorised access. Appropriate audit trails are in place so that all parts of the process can be checked.

Source code plagiarism has become a serious problem. Assessed assignments in larger modules may consist of hundreds of submissions which make manual comparison for evidence of plagiarism of all possible assignment combinations impractical. BOSS incorporates novel plagiarism detection software (Joy and Luck 1999; White and Joy 2005) which compares submissions automatically to seek for evidence of plagiarism and if evidence is found presents the offending submissions to the teacher for manual comparison.

2.7.1.4 Platform independence

Java was chosen to form the basis of a complete re-write of the system, not only because it would run on all major operating systems, but because its object-oriented paradigm together with a wide selection of supported Application Programming Interfaces (APIs) were seen to be supportive of rapid and accurate coding.

Two possible solutions to platform independence were considered. The first would involve Java clients and servers (so that BOSS would become an application which would run on student/staff computers), and the second a web-based solution accessible through web browsers. Since there are

compelling arguments in favour of each solution, both have been implemented and are currently supported.

2.7.1.5 Architecture of BOSS

The software uses client-server architecture with separate clients for students and for authorized staff (for security reasons). Each client is provided both as a secure web client and as a stand-alone application, so maximizing the flexibility of the system in terms of a user's working environment. There are consequently two distinct views of the software, according to whether the user is a student or a member of staff.

Component model

Assessments take a wide variety of forms, including single tasks (such as essays) or potentially complex entities (such as examinations). It is not uncommon to encounter a rubric such as, "Attempt question 1, question 2, any three questions from section B, and one question from section C".

The data model used by BOSS, i.e. the component model, is intended to support arbitrarily complex assessment structures. The model is simple, straightforward to store in a relational database, and able to cope with any rubric.

A complex assignment (in terms of choices and paths through the tasks to be completed) may be desirable as a component of an adaptive and individualised learning and teaching strategy. The purpose of introducing the component model is to free the teacher from restrictions on the structure of an assessment, allowing a complex assessment model to be deployed.

The component model is a description of the structure of an assessed piece of work. It is intended to cover all possible assignments given to students, including continuously assessed work, examinations, and tests.

The component model is based on the following four fundamental notions.

(i) A problem is a single task (such as a multiple-choice question, or an essay) which is not divisible into sub-problems, and has a maximum mark as an attribute.

(ii) A multi-component which is a triple (C, AC, MC), where C is a non-empty set {c1, c2, ..., cn} of components, AC is an integer in the range 0..|C|, and MC is an integer in the range 0..100. AC represents the number of components a student is expected to attempt. MC is the maximum mark for the whole multi-component. If AC=0, then a student is expected to attempt all sub-components. The maximum marks for the sub-components are used to determine the relative weightings of those components.

(iii) A component is either a problem or a multi-component.

(iv) An assessment is a multi-component.

2.7.1.6 Student's view

The BOSS software permits students to perform two principle tasks which are to submit their (programming) assignments online and being able to run automatic tests on their programs prior to submission (and afterwards if they wish to re-submit within the prescribed deadline).

2.7.1.7 Staff view

The BOSS software permits staff to perform five principle tasks.

i. Automatic tests can be run on the set of student submissions, and as part of the marking process. These tests may be a superset of those that a student can run, or they may be separate. For example, students may be given a (small) set of automatic tests to run on their programs prior to submission, for the purposes of ensuring that they have met minimum requirements for the assignments.

   Further tests available to staff alone might then be used to assess how well each student has met the full set of requirements.

ii. Plagiarism detection software assists in the identification of potential intracorpal source-code plagiarism.

iii. Submissions can be marked online by viewing the results of the automatic tests, running the submitted program, and viewing the submitted source code.

iv. Staff authorised by the module organiser can moderate the marks given to students' work by other markers.

v. Feedback can be given on each submission, and BOSS compares the feedback from the set of markers of a given submission and provides a mechanism for communicating this back to the student. In order to deliver these tasks in a manner which ensures data privacy (staff can only perform appropriate tasks) and allows for multiple marking of an item of work to be performed "blind", there are four staff roles, as follows.

Administrator. The administrator may create modules and allocate managers to individual modules. This role is not a "super user" one, and the administrator's view of the data is strictly limited.

Manager. Each module is allocated one (or more) managers, who can edit all properties of that module, including assignment details, marking criteria, and allocation of markers and moderators

Marker. Each problem contained within an assignment is allocated one or more markers by the module manager. Each marker is allocated submissions to mark, and will mark online according to the marking criteria authored by the manager. Weightings of individual marking categories, and the identity of the student, are hidden from the marker in order to ensure fairness and transparency of

the marking process. The markers have the opportunity to write feedback on the work marked, and it is expected that the manager will issue the markers with guidance as to what type of feedback is appropriate for that particular problem.

Moderator. Once a problem has been marked by all markers allocated to it, a moderator is required to review the marks awarded and the feedback given. If multiple markers have been allocated to each student, the moderator's view will contain all the marks awarded, and a "suggested" moderated mark for each marking category, which the moderator is free to alter. The weightings for the individual marking criteria are available to the moderator, but the student's identity is not.

Only when a student's work has been moderated are the final results available to the manager.

The ideal model, if resources are available, is for each piece of work to be double marked, moderated by a third person, who may or may not be the module manager.

However single marking is permitted by BOSS, in which case the role of moderator becomes one of checking the consistency and accuracy of the marker.

The BOSS system consists of three servers: student server, staff server and the testing (or slave) server. These are actually three separate Java processes which are usually run on the same machine but can, if so desired, execute on three physically separate machines. The primary function of the student server is to authenticate students and receive their coursework submission for appropriate storage and logging.

The student server is also capable of communication with the testing server if the automatic code tests are available to the student before they make their final submission. The staff server, to which access is only permitted to fully authenticated members of staff, provides testing, marking and moderation functionality to the client software. The testing server is not directly accessible, the staff and student servers communicate with it to request the automatic testing of student submissions. Each server executes as a process without administrative (super user) privileges to prevent the compromise of the entire machine should one server be maliciously attacked and compromised, an event which has not yet happened in the lifetime of the project. File system and database privileges are carefully allocated for each server. The test server is used to run submissions through a series of fully automatic tests. The testing system is functionally separate from the core BOSS system allowing some flexibility in the deployment of a testing system which may, depending on the scale of automatic testing required by the institution, involve separate computing hardware. Transfer of submissions from the student and staff servers to the testing server is encrypted to prevent malicious modification or theft of a submission.

BOSS offers automatic testing functionality in two modes: submission-based and batch-based. The first is available to students at submission time. The course manager can make available tests that give immediate feedback to students. These tests can be used as a simple check of the submission

and can help prevent erroneous submissions. Furthermore, based on the feedback given to them students can revise their submissions if they discover that they have not met the requirements (assuming that the final deadline has not passed). The majority of the automatic testing is performed in the second mode and cannot be seen or executed by a student. These post-submission tests are typically executed by the course manager as a batch job after the final submission deadline. The results of the post-submission tests can be linked to marking categories which assess the correctness of a submission freeing the marker to spend a greater amount of effort assessing the style of the submission.

2.7.1.8 Databases

Central to a data-bound application such as BOSS is the storage and management of the data. In addition to storage of submitted assignments as archives on secure backed-up file systems, an SQL database is used for other data, such as times of submissions, basic student identity information, and marks awarded. The initial deployment of a proprietary database was found to be unsuccessful (due to the repeated requirement of systems staff to manage the database), and Open Source databases such as MySQL, MSQL and PostgreSQL have since been used. Differences between the dialects of SQL used are a continual source of frustration, though the latest versions of MySQL and PostgreSQL allow interchange ability with minimal intervention, assisted by the use of Java Database Connectivity (JDBC) to connect with the Java servers.

2.7.1.9 Pedagogic Evaluation

BOSS is intended to be "pedagogically neutral". The use of BOSS over a period of years has demonstrated the effectiveness of a focused tool which addresses the requirements of assessing students' programming skills. The inclusion of a generic database schema and plagiarism detection software, together with a platform-independent client-server architecture, provide a foundation which is adaptable to changes both in technologies and in pedagogic requirements.

2.7.2 WeBWorK assessment system for Programming Fundamentals

WeBWorK is an assessment tool for use in the core courses of the computer science Curriculum, referred to in "Computer Science Curriculum 2001" as Programming Fundamentals .These core courses revolve around the following computer science essentials: fundamental programming constructs, algorithms and problem-solving, elementary data structures, recursion and event-driven programming.

WeBWorK was developed to solve problems written in the Problem Generating language (PG) for the Java, Python and SML (Standard Meta Language) programming languages. These problems were designed to test students on terminology, basic programming concepts, and the syntax and semantics of Java, Python and Standard Meta Language (SML).

Permission was also granted by the authors of a leading textbook to use sample student questions (Java Software Solutions: Foundations of Program Design (4th Edition), John Lewis and William Loftus, 2004). Whenever possible, the Problem Generating language PG was written so that problems would be randomized and individualized on a per-student basis.

2.7.3 PAT (Programming Adaptive Testing) system

PAT is a Web-based adaptive testing system for assessing students' programming knowledge. PAT was used in two high school programming classes by 73 students. The question bank of PAT consists of 443 questions. It was developed with the use of a Flash Mx tool. The programming was conducted in Action Script and the final files were extracted in html format. PAT can be used in the school's computer laboratory or via Web from anywhere. It was tailored to the course of "Application Development in a Programming Environment" (Bakali et al., 2004).PAT is not only a software tool to assess novice students in Programming but it can also predict their classification in the Programming course in National Exams.

2.7.3.1 Questions in PAT

In PAT, each question is classified to a difficulty level: A = easy question, B = moderate question, C = difficult question. In addition, the question's content was developed according to the low levels of Bloom's Taxonomy (Bloom, 1956).

The following Categories of questions were developed

i. **Recall of data**: Knowledge questions on the Theory of the course, the Syntax and Function of Frameworks of Structured Programming and of Sub-Programs in True/False and MCQ format (difficulty level A, B or C). Such questions examine student's memorization capability.

ii. **Comprehension**: A piece of code and a question involving the behaviour of the code (finding the output after the execution of a program). Such questions have been found efficient (Lister, 2001) as far as student's assessment on their ability to read and comprehend the code's Semantic (difficulty level B or C).

iii. **Application**: Exercises to examine students' skills to apply prior knowledge to new problems. Three types of exercises were used: 1) a piece of code, which can be realized through a Structure of Process or Choice or Repetition, where a student is called to choose an equivalent command for the execution of the above functions (difficulty level B); 2) also a Logical Diagram is given, where the student is called upon to find the equivalent command to express one or more functions (difficulty level C); 3) gap filling in a piece of code or program according to some expressions (Lister & Leaney, 2003a). Program gap filling difficulty (level B and mostly level C) is the most difficult activity and needs much more consideration and

capabilities, also it helps students in increasing their power of solving sub-problems (Hwang et al., 2008).

2.7.3.2 Model Structure

PAT presents to a student 30 questions from various Chapters of the exam material, depending on the students' level. Each student is tested on different questions at different levels. This ensures the quality of the exams as far as cheating is concerned, since students sit in close proximity in computer laboratories.

The student moves from one difficulty level to another according to his/her answer. If s/he answers an "A" question correctly then the next question is "B" otherwise it is "A". If s/he answers a "B" question correctly then the next question is "C" otherwise it is "A". If s/he answers a "C" question correctly then the next question is "C" otherwise it is "B". At the end of the test, PAT shows the student's total number of correct and wrong answers per chapter and level. Also, it shows the student's total number of correct answers out of 30, his/her final score and classification.

2.7.3.3 Grading

Significant effort was placed on Feedback. PAT seeks to serve both the teacher and the student. As far as the student is concerned, PAT not only serves as a means of practice on the exam material, but also as a means of feedback on student's shortcomings per chapter. As far as the teacher is concerned, PAT functions as a means of assessing the students' programming levels which indicates how well they are prepared for (National) exams. Then the teacher could try to help students overcome their weaknesses.

2.7.3.4 Analysis of the results

The student correctly answers all 30 questions (from 0 to 29), s/he will obtain the following best performance sequence of question levels:

A, B, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C.

On the contrary, if the student answers all 30 questions incorrectly, the worst performance sequence of question levels will be as follows:

A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A,A.

In the Results printout, the answers given by the student are characterized by the letter of the difficulty level and the corresponding question number, LQn, where L is the difficulty level (A, B or C) and Qn is the number of the question at the corresponding Level (Qn= 0..185 for level A, Qn=0..147 for level B, and Qn=0..108 for level C). For example, the following questions sequence appeared at a student's Results printout:

A5, B7, C3, B33, A12, B1, C77, C4, C100, B18, C5, C7, B22, A23, A27, A34, A47, A61, B75, C62, C55, C59, B81, C80, C19, B9, C0, C41, B29, A30.

This questions' sequence helps the teacher to immediately recognize which questions the student failed.

Regarding the example's questions sequence, the student answered wrongly the following questions:

C3: because a level B question follows B33: because a level A question follows 14 Also C100, C7, B22, A23, A27, A34, A47, C59, C19, C41 and B29.

At the end of the test, the following results are presented for each student:

(a) *Total Result (x)*: Number of the correct answers out of 30,

(b) Number of the correct answers per level in relation to the total number of questions per Level,

(c) *Final Score* (y) given by the following formula:

Final Score = 1* Number of Correct Answers at level A+

2* Number of Correct Answers at level B+

3* Number of Correct Answers at level C

(d) Classification of student which depends both on the Total Result and on the Final Score.

More specifically the classification is calculated as follows:

If (0<=x<=17) and (0<=y<=33)
TRY HARDER - LOW PROGRAMMING SKILLS!

If (16<=x<=20) and (34<=y<=51)
GOOD – MEDIUM PROGRAMMING SKILLS!!

If (21<=x<=30) and (52<=y<=87)
VERY GOOD – HIGH PROGRAMMING SKILLS!!!

2.7.3.5 Strengths and Weaknesses of PAT

Strengths of PAT include: Successful classification of the students, Prediction of students' performance in Greek National Exams, automated assessment process, speed in Results production, Large library of questions - possibility of test repetition with renewed interest, Pleasant and usable Graphic Work Environment (it was developed using FlashMx).The execution of PAT software requires only the installation of a browser and one can run PAT from any hard disk device even without Internet connection.

Shortcomings

PAT contains items to test only beginners in programming. Also, it was developed to test student's programming skills on "Glossa", a pseudo-language for Greek students.

2.8 Assessment Approaches

There are two common approaches in assessing programming exercises automatically; Ala-Mutka lists features of programs that have been assessed automatically. She divides features between the ones that need execution of a program (i.e. dynamic analysis) and the ones derived from a program code without executing it (i.e. static analysis). According to Ala-Mutka, functionality, efficiency, and testing skills are typically assessed through dynamic analysis. Static analysis, on the other hand, is used to give feedback on programming errors (e.g. dead code), various software metrics, and design.

In addition, both static and dynamic analyses are used to give feedback from various special features (e.g. GUI testing).

Features that are assessed using code assessment systems include:

2.8.1 Functionality

Functionality evaluates the correctness of programs' behaviour. It is the most common automatically assessed feature of programs and nearly all systems are able to give feedback from it.

2.8.2 Efficiency

Efficiency of computer programs is typically related to the usage time and space (i.e. memory) but the usage resources like disk space, network, or even power consumption of a portable device, can be relevant. Ala-Mutka focuses only on the time efficiency, perhaps because it still seems to be the dominant or only efficiency metric supported by automated assessment platforms. There are many options of presenting feedback in this category. For example, CPU time can be plotted as a function of input size as in the algorithm benchmark extension of OpenCPS.

2.8.3 Test adequacy.

Students write tests to their own or some other programs and feedback is then given based on various test adequacy metrics .Automated assessment of testing skills was supported already in 1997 by a tool called Assyst. Today, Web-CAT is a widely used tool designed around the principle that students test their own programs.

2.8.4 Style

Style is perhaps the most obvious feature to be assessed through static analysis. In most programming languages, there are (de-facto) standards defining indentation, variable naming conventions and other typesetting features. Although modern Integrated Development Environments (IDEs) e.g. Eclipse3 make it easier to write well-formatted code, giving feedback from style is still relevant. Quality of comments and related documentation are also part of programming style.

2.8.5 Programming errors

Programming errors includes use of un-initialized variables, dead code (i.e. code that will never be executed), and other errors detectable by static analysis. Many automated assessment tools incorporate static analysis tools such as Lint family (e.g. JSLint4) and Valgrind5.

2.8.6 Software metrics

Software metrics are easy to generate but the educational motivation needs to be carefully considered each time. For example, statement count, branch count, cyclomatic complexity, lines of code, lines of comments, percentage of lines containing comments, and code depth are useless for students unless accompanied by a desired value or range for the measure.

2.8.7 Design

Design might not be the most obvious automatically assessed feature. Feedback is typically about the details of the design, not really about the high level design. Examples of recent work that could be used in automated assessment of the higher level design is work by Dong et al. to recognize design patterns from programs, and work by Taherkhani to recognize different sorting algorithms through static analysis. Lower level design issues that are actually used in the existing automated assessment tools check if the structure of the solution matches the pool of allowed structures (e.g. there is a loop or a recursion present).

2.9 Chapter summary and Conclusion

This chapter has highlighted the following areas: Key factors that can contribute to the successful application of automatic assessment of programming assignments, related works, review of features supported by recent systems for automatic assessment of programming examinations, review of existing systems and features that are assessed using code assessment systems.

Despite the availability of other automatic assessment tools that could be used for programming assessments, this research has made a contribution through the development of a programming assessment system which can assess programs written in the vb.net programming language. Most of the available systems support assessment of java and C++ and there is still a need to assess visual basic.net programs as well.

An assessment system like PAT can be extended to support the assessment of other programming languages e.g. Java, Visual Basic as well as code writing exercises and then used by students.

The Webwork system can also be improved so that it is able to assess advanced programming concepts.

Static analysis approach has been adopted in this project. The reason it was chosen over dynamic analysis approach is that it is much cheaper and easier to develop and deploy. Static analysis does not require a complicated algorithm or code to develop thus the development cost is reduced.

Another advantage is that it does not require code compilation and execution and this reduces the server workload (Trung et.al.)

Under static analysis approach, test runs and keyword analysis methods have been used in assessing the student code. During a comparison, the system will search for specific keywords which have been defined by the instructor. With test runs, expressions are given by the teacher and tested with every student. The two methods were chosen over other methods because they were simple to implement under static analysis approach.

CHAPTER THREE: METHODOLOGY

This chapter presents the methodology used in this research. Research methodology is a system of explicit rules and procedure upon which research is based and against which claims for knowledge are evaluated (Nachmias and Nachmias, 1996). The Methodology was focused on the sources of data and their collection techniques, tools for data presentation analysis and interpretation.

3.1 Research design

Research design forms the conceptual structure within which the research is conducted, the plan for collection, measurement and analysis of data.

The research design used was Case study design.

This research was guided by a case study of a learning institution in Kenya.

The following procedure was followed in conducting the case study:

3.1.1 Planning

This involved identifying stake holders (those involved) in the data collection process and identifying a case study topic. The stake holders involved were the programming instructors and the students at the institution.

3.1.2 Developing Instruments for data collection-During this research, questionnaires and an interview schedule were prepared and later used to collect data.

3.1.3 Collecting data –During this phase the case study topic was introduced to the stake holders. Data collection was conducted in two phases. During phase one the oral interviews were conducted in order to gather user requirements and understand what challenges the students and instructors were facing. During phase two, questionnaires were administered to evaluate the system's usability and effectiveness.

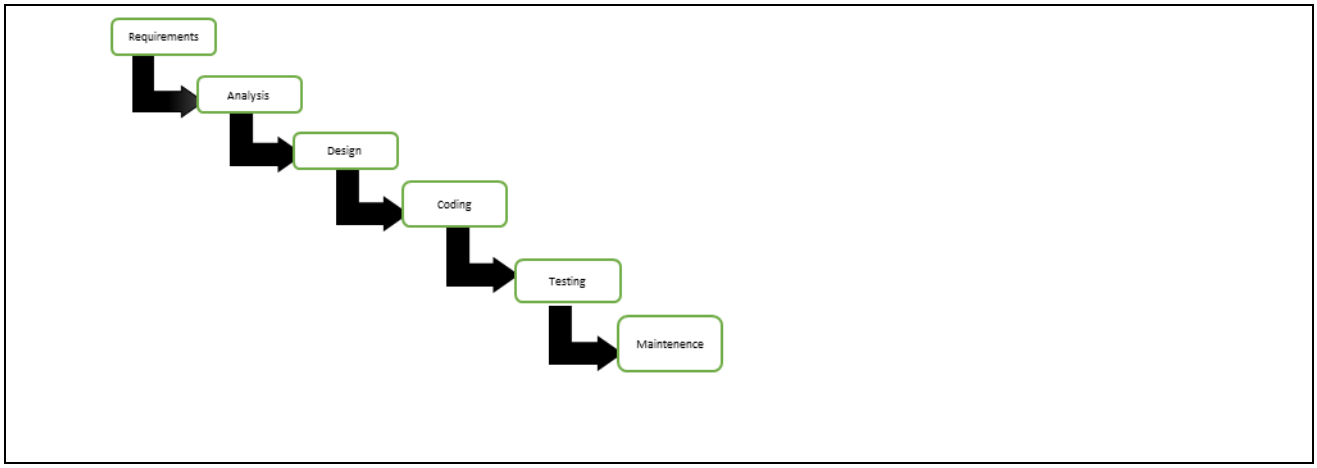
3.1.4 Analysing the data-This was done by analysing the questionnaire data using pivot tables in Ms excel. The data was keyed into excel worksheets and the COUNTIF formula was used to check for frequency. The SUS questionnaire adjective rating scale was used to check for system acceptability levels of the participants in usability testing. The analysed data was presented using column charts.

3.1.5 Revising the findings-This was done by documenting the results based on the elements of the case study which included the problem and how the problem was identified, whether the process for identifying the problem was effective and the results obtained.

3.2 System design

In system design, the waterfall methodology was used. The model views the process of software development in five stages. The activities in one stage are completed before moving to the next.

Figure 1: Stages of the waterfall model



3.2.1.1 Requirement gathering and analysis*:*

Findings from the case study were used to determine requirements. Data obtained from the literature review was also considered.

3.2.1.2 System Design*:*

The requirement specifications from the Requirement Analysis was studied and a system design prepared to help in specifying hardware and software requirements and also help in defining the overall system architecture.

3.2.1.3 Coding*:*

This stage involved the actual development of the system by developing the graphical user interface, implementing the model using Asp.net and creating the system database using SQL server.

3.2.1.4 Testing:

This is the stage after coding where every unit of the program was tested and integrated as a complete system to ensure the system was working according to required specification.

3.2.1.5 Maintenance*:*

This is the final stage of development in which all necessary maintenance activities are carried out in order to see that the software continues to work even when there is a new development in the future.

3.3 System technologies

In the design of the system, the following technologies were used:

3.3.1 Software

i. Operating system (windows 7)
ii. SQL server 2008 and higher versions.
iii. Visual studio 2010 software
iv. Web browser
v. Web server (IIS 7) -IIS has been used in order to handle server side processing .This Web server will allow a computer to host Web pages.
vi. Crystal reports version 11 for generating system reports.

3.3.2 Hardware

i. Random access memory-4GB
ii. Hard disk-250 GB
iii. Processor- Pentium Dual-core—CPU 2.10 GHz

3.4 System description

The online code assessment system for visual basic.net programs works with three roles: Administrator's role, Instructor's role and Student's role

3.4.1 Administrator Role

The administrator can**:** Create/delete accounts (add a list of instructor names and list of students names),create default passwords for instructors and students, register and book the student for the exam, view a list of all the system users he has added to the system and create and manage sessions in the system.

3.4.2 Instructor Role

The Instructor can**:** view the students score sheet, add exam question categories, add exam questions from the exam setup page,view the list of questions that he has added to the system, view individual student performance reports as well as reports for the entire list of students who have sat for an examination.

3.4.3 Student role

The student can: Log in to the system using their assigned username and password, sit for the exam and submit the exam for marking, obtain a score sheet from the system which displays the student total score and grade.

3.4.4 The system

The system picks questions and answers from the question and answer banks. Questions are picked in a random format.

Other features in the system include:

(i) Alerts –a reminder is generated to alert the candidate at least ten minutes before time expiry when they are sitting for the test.

(ii) Password authentication for the candidates, instructors and the administrator. The administrator schedules the student for the exam and the student is prompted to enter a user name and a password before sitting for the exam.

(iii) The system produces an immediate detailed report of the test highlighting the Students score according to various topics.

Apart from the individual students score sheet, the system generates other reports such as the performance of the entire group of students who have sat for an exam and performance reports on a semester basis.

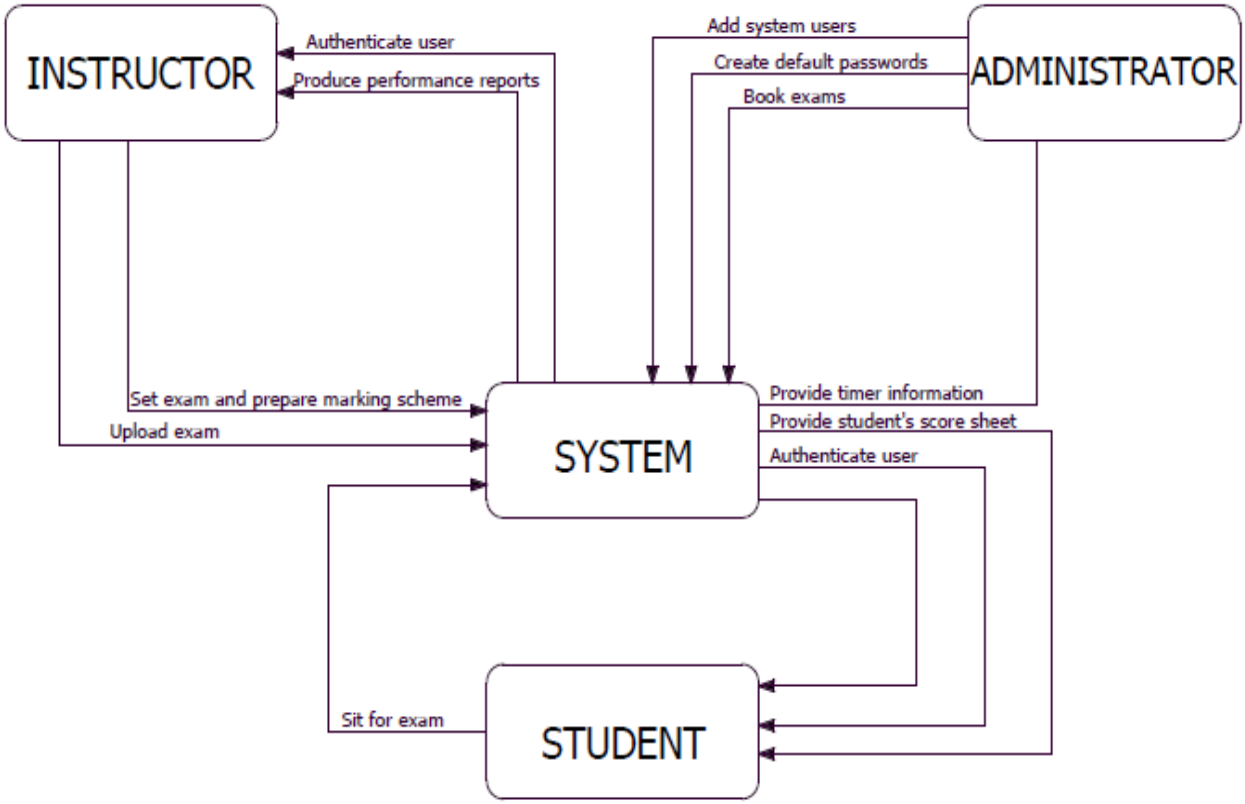
(iv) Marking of the questions is done by comparing the student's code with what has been provided as the solution by the instructor. Keywords entered by the instructor in the system are also used to analyse the student's answer.

The student is awarded a grade of a pass or fail.50% and above is categorized as a pass and 0-49% a fail. Questions carry different marks and that is determined by the complexity of the question. Each exam has a maximum of ten questions.

(v) The system contains a help feature which when clicked by a user provides step by step instructions of performing a task.

3.5 System Architecture

Figure 2 : System architecture

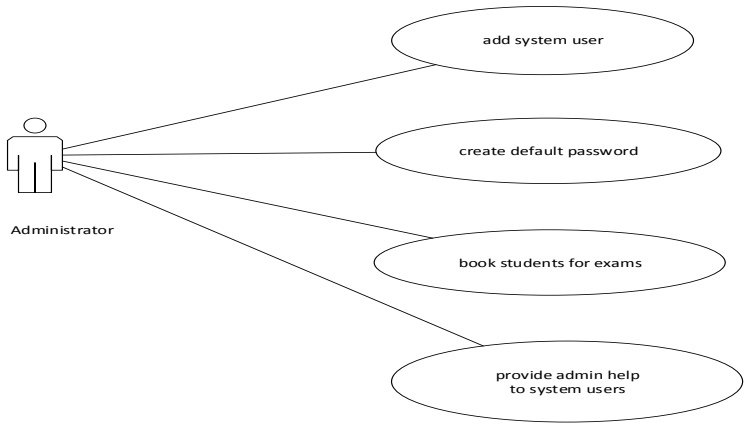


The system architecture diagram depicted in the figure above shows the relationship between the entities in the system. The entity STUDENT can take examination after he or she gains access to the system. The entity INSTRUCTOR can upload questions to be answered by student into the database and create a marking scheme for the questions. The entity ADMINISTRATOR has the responsibility of adding system users and setting the default passwords for the users of the system. The entity SYSTEM is responsible for authenticating the users of the system and also providing the timing functionality for the examination. The system logs off a student upon expiration of duration for the examination

3.6 Use case diagrams for the online code assessment system

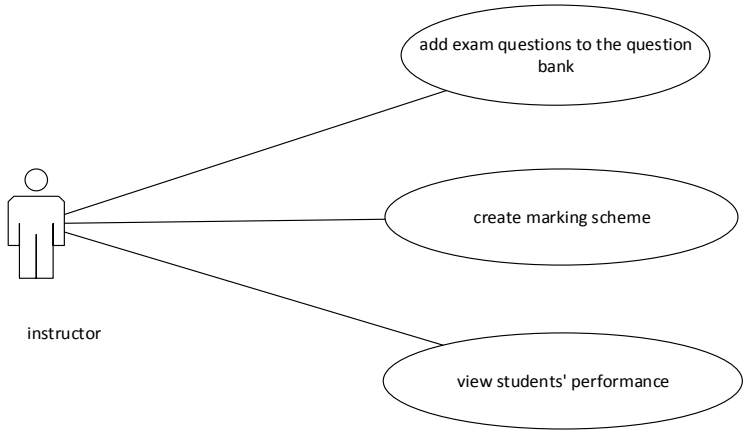
Use case diagrams describe the functionality of a system and the users of the system. The use case diagrams consist of actors and use cases. Use cases are the services provided by the system to the actors (users). Use case diagrams for each entity present in the system is presented below. These include use case diagrams for the administrator, lecturer, server and student.

Figure 3 : Use Case diagram for the administrator role



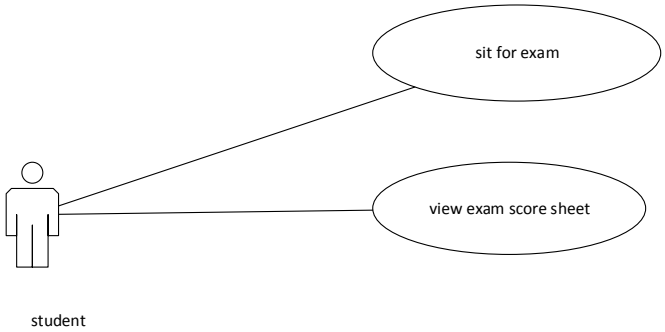
The above use case diagram shows the activities that are performed by the administrator including adding system users, creating default passwords, booking students for exams and providing admin help to the system users.

Figure 4: Use Case diagram for the instructor role



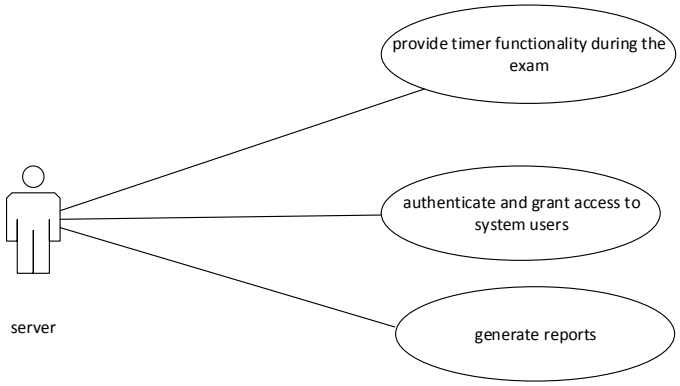
The above use case diagram shows the activities performed by the instructor including the setting of examinations and creating marking schemes, viewing students' performance and uploading set exams.

Figure 5 : Use Case diagram for the student role



The above Use case diagram shows the activities performed by the student including sitting for examinations, submitting the exam for marking and viewing the exam score sheet.

Figure 6: Use Case diagram for the server



Server use case diagram represents the responsibility of providing timer function during the exam, authenticating and granting access to system users, generating reports and picking questions randomly from the question bank.

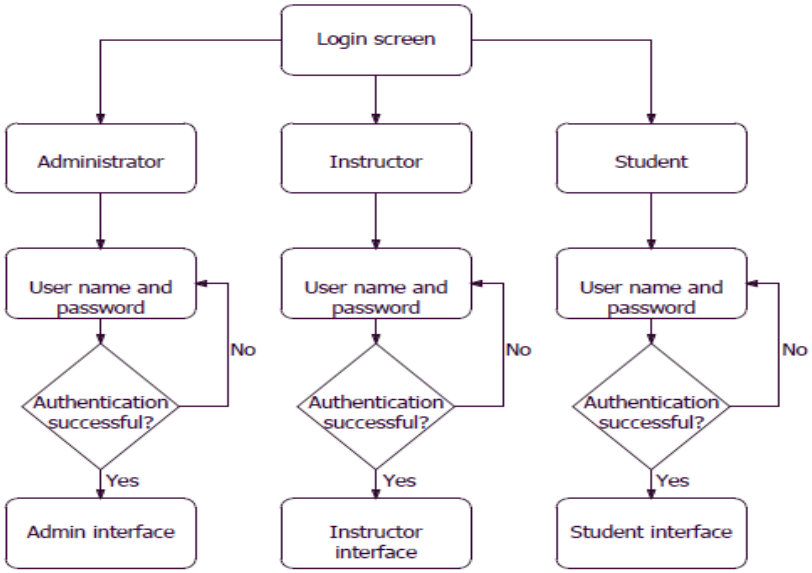
3.7 Database design

The organization of data in the database aimed to achieve three major objectives: -

i. Data integration.
ii. Data integrity.
iii. Data independence.

The system stores the information relevant for processing in the MS SQL SERVER database. This database contains tables, where each table corresponds to one particular type of information.

Figure 7: System process flow diagram



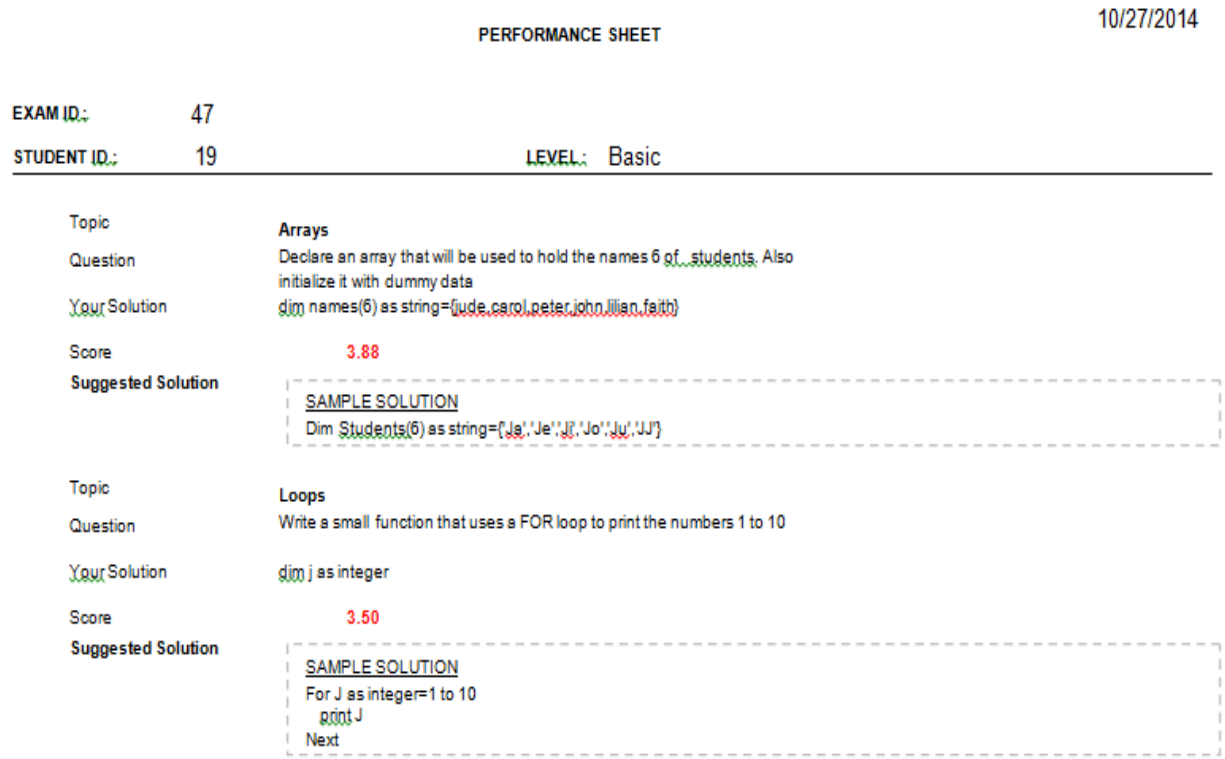
The diagram above illustrates the flow of processes from login screen to user's respective interfaces. Every user of the system is assigned a username and password .The system checks whether the username and password are correct and grants access to the user otherwise the user is prompted to enter their username and password again.

3.8 Interface layouts

Figure 8 : Login page

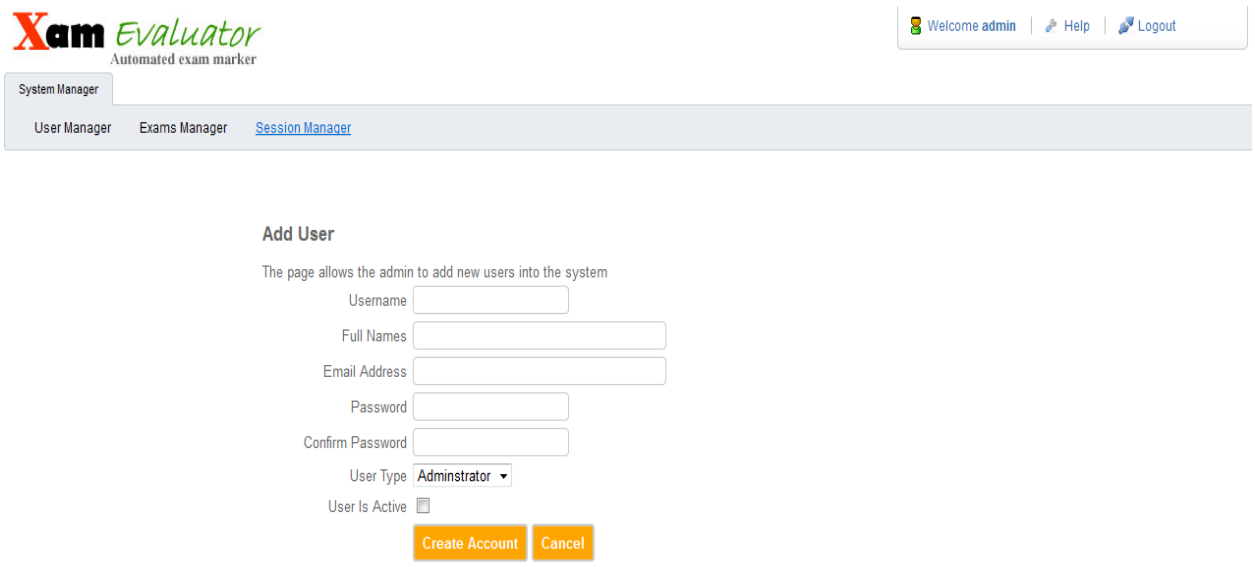
The figure above is a screen shot of the login page. This is the default page of the system. It is also known as the homepage of the system that automatically loads after the URL has been requested for by a web browser on the client system.

Figure 9: Student's performance sheet



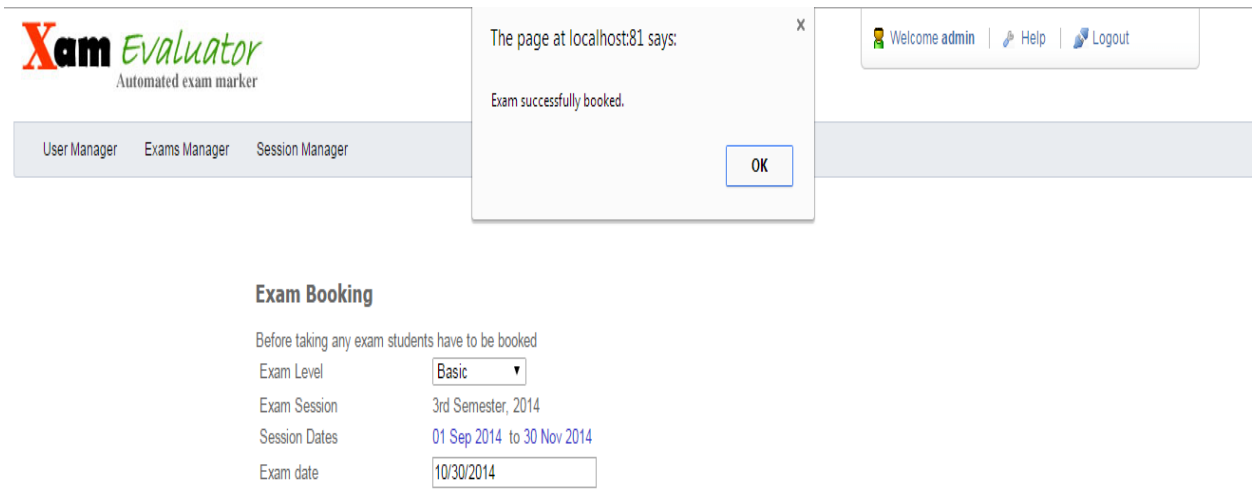
The figure above is a screen shot of the student's scoresheet.This is generated once the student submits a finished exam for marking. The performance sheet shows the student's solution and the expected solution.

Figure 10: Add system users' page



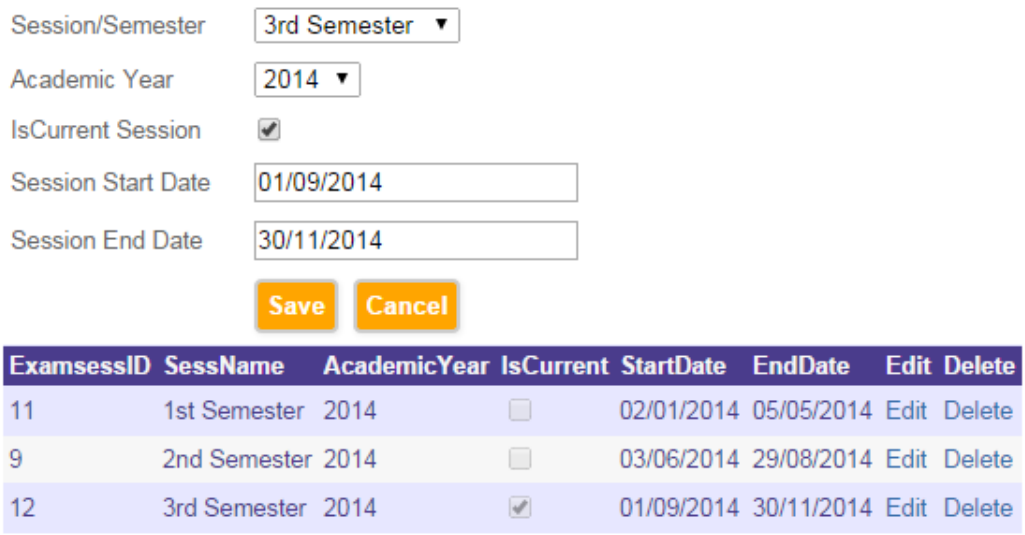
The figure above is a screen shot of the add user page. The page allows the administrator to add new system users by entering their usernames, email addresses, creating default passwords and creating user accounts for respective users.

Figure 11: Exam booking page



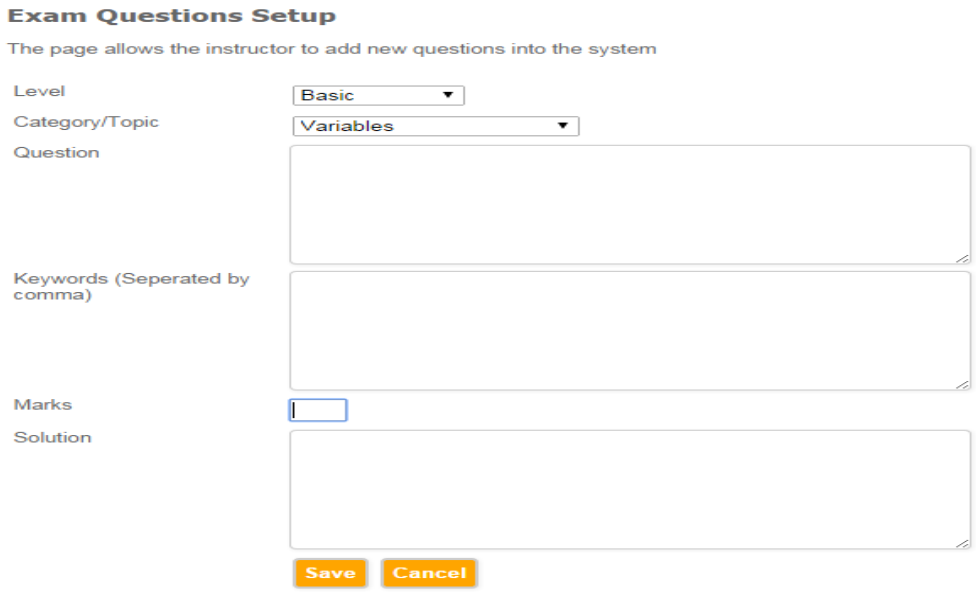
The figure above is a screen shot of the exam booking page. This page shows details that have to be filled so as to book an exam. This is followed by a message box indicating the exam has been booked successfully.

Figure 12: Manage session's page



The figure above is a screen shot of the manage sessions page. Sessions are categorized as 1st semester, 2nd semester and 3rd semester. The administrator can create session start dates and end dates from this page and specify whether a session is the current session.

Figure 13: Exam setup page



The figure above is a screen shot of the Exam setup page. The instructor can add questions by topic, add solutions, allocate marks to a question, add keywords which are expected in the student's input and finally save the question in the question and answer bank.

3.9 System functionality

The system is divided into the following functionalities

3.9.1 User Registration

The administrator registers students and instructors. The instructor's details include: Full Names, User name, Email address, Password, User type and whether instructor is an active user or not.

The Student's details include: Full Names, User name, Email address, Password, User type, whether instructor is an active user or not.

3.9.2 Booking exams

This is carried out by the administrator. He captures the exam level, exam session, session date, exam date, and Selects student's user id.

3.9.3 Uploading questions

This is done by the instructor. The instructor sets questions and prepares a marking scheme for each exam uploaded.

3.9.4 Marking the exam

This is done by the system. The answer that is typed by the student is compared to what the instructor has provided as the solution in the system by implementation of character matching and keyword analysis strategies to assess the code correctness.

3.9.5 Producing results

The results are generated by the system and a score sheet is produced for the student to view their scores.

CHAPTER FOUR: RESULTS AND DISCUSSION

This chapter provides an overview of the system testing strategies carried out on the developed code assessment system.

4.1 Testing

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets specified requirements.

Five categories of testing carried out were: System testing, Validation testing, Usability testing, Code assessment testing and System Effectiveness testing

4.1.1 System Testing

This involved performing a variety of tests on the system to evaluate its behaviour as defined by the scope of the project. The main reason for conducting system testing was to verify the system against specified requirements. The system was checked to determine whether it was behaving as per expectations.

4.1.1.1 System Testing Results

A test case is usually a single step, or occasionally a sequence of steps, to test the correct behaviour/functionality and features of an application. An expected result or expected outcome is usually given.

Table 1: System Testing Results

The table below shows a list of test cases that were used to conduct system testing.

Test Case No	Task	Expected Results	Actual Results	Status
TC1	Installation of the system	The System should install and display a login screen.	The system is installed and display a login screen	Pass
TC2	Admin/instructor/student log in using username and password	The system directs the user to their respective interface	The system user logs in by typing a user name and password	pass
TC3	Admin adds a new user to the system	The system generates a message saying the user has been successfully saved.	New user is added to the system	Pass
TC4	Admin books student for an exam	System generates a message saying the exam has been booked successfully.	Student is booked for the exam	Pass
TC5	Admin adds a new session start date and end date	System displays a message indicating that the session has been	Session is added to the system	Pass

		created successfully.		
TC6	Admin viewing of all system users	System displays a list of all system users.	All added system users are displayed	Pass
TC7	Instructor adding questions to the system	The system produces a message box saying the question has been successfully saved.	The question is added to the system	Pass
TC 8	System user changing password	The system directs the user to the login screen so that they can login using their new password	The system user enters the current password, new password and confirms the new password	Pass
TC9	Instructor viewing of reports	Instructor can view individual student performance and for the entire class.	Instructor can view individual student performance and for the entire class.	pass
TC 10	Instructing adding question categories	System displays a message indicating the category has been successfully saved.	The new category is added to the system	Pass
TC11	Student selecting next and previous question	The system directs the student to the previous or next question page.	The previous or next question link is selected/clicked.	Pass
TC12	Student clicking on submit button to submit the exam for marking	System displays a score sheet for the student showing their name, level of study, total score and grade.	Exam is submitted for marking	Pass
TC 13	Instructor viewing the list of questions added	The system displays a list of all added questions in the system.	A list of added questions is displayed.	Pass
TC14	Timer is started once the student clicks on take exam button	System displays the running timer on the screen.	Timer is running	Pass

The table above shows a list of use cases used to conduct system testing, tasks carried out, expected and actual results. From the system testing results, all tasks carried out during the test passed the test.

4.1.2 Validation testing

Software validation is the process of testing software to check whether it satisfies the customer needs or not. This testing is done during and/or at the end of the process of software development.
The following tasks were validated during validation testing: Password authentication, correct computation of results, Email validation and Exam booking.

4.1.2.1 Validation testing results

The following screen shots have been used to show validation testing results

Figure 14: Password authentication

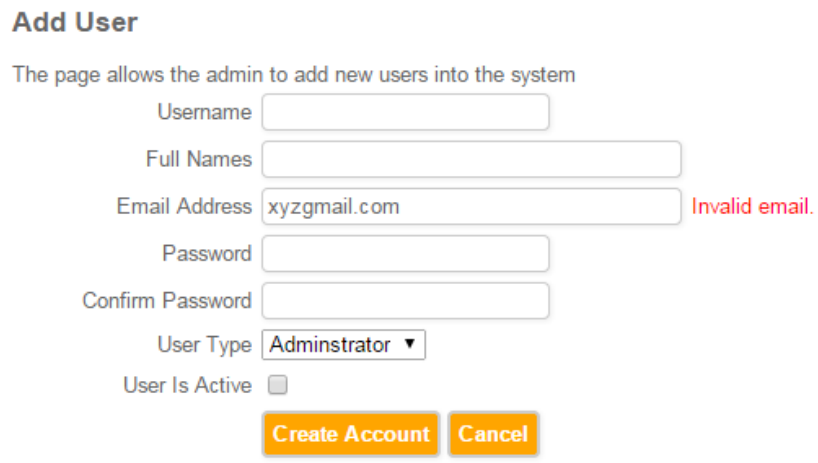
The figure above is a screen shot of the login screen where a user upon logging into the system is prompted to enter a username and a password. Whenever a user enters the wrong password, the system generates an invalid username or password error message.

Table 2: Correct Computation of Results

STUDENT SUMMARY PERFORMANCE SHEET			
STUDENT ID	**TOTAL SCORE**	**MARKS/ GRADE**	**LEVEL**
18	52.38/70	74.83% pass	BASIC
19	50.88/70	72.69% pass	BASIC
15	47.21/70	67.43% pass	BASIC
16	54.96/70	78.51% pass	BASIC
9	46.45/70	66.36% pass	BASIC
8	30.59/70	43.70% fail	BASIC
11	20.83/70	29.76% fail	BASIC
20	24.37/70	34.81% fail	BASIC
21	16.21/70	23.16% fail	BASIC
22	45.83/70	65.47% pass	BASIC

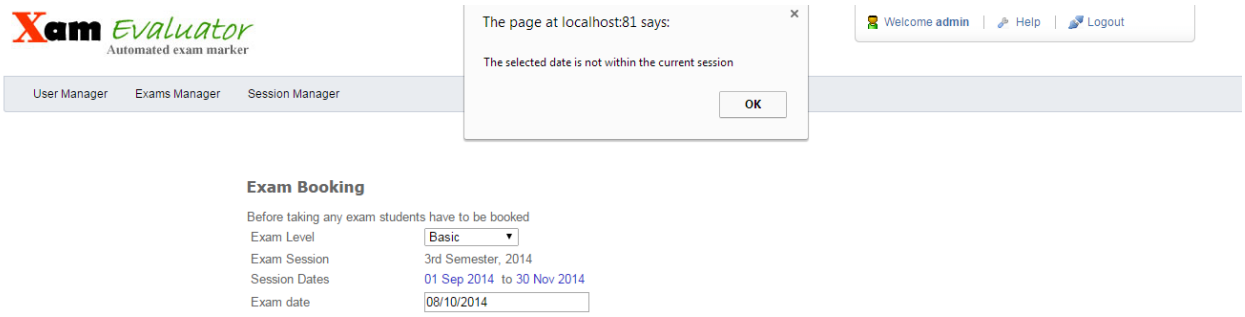
The table above shows student summary performance. Students with a score of 0-49 marks are graded as “fail” and those with scores of 50-100 marks are graded as “pass”.

Figure 15: Email Verification



The figure above is a screen shot of the add user page. Invalid email formats entered are detected by the system and an invalid email error message is generated prompting the admin to enter the correct email.

Figure 16: Exam Booking



The figure above is a screen shot of the exam booking page. The system is able to validate the exam booking activity according to date and session.

In the above screenshot a student is booked for an exam on 10th august but the system rejects the booking because the date is not within the current session.

4.1.3 Usability Testing

Usability testing refers to evaluating a product or service by testing it with representative users. Questionnaires were used to achieve usability testing.45 participants took part in the survey.29 students, 10 instructors and 6 administrators.

The usability test was done to find out whether participants were able to complete specified tasks successfully, identify how long it took to complete specified tasks and find out how satisfied participants were with the system's performance.

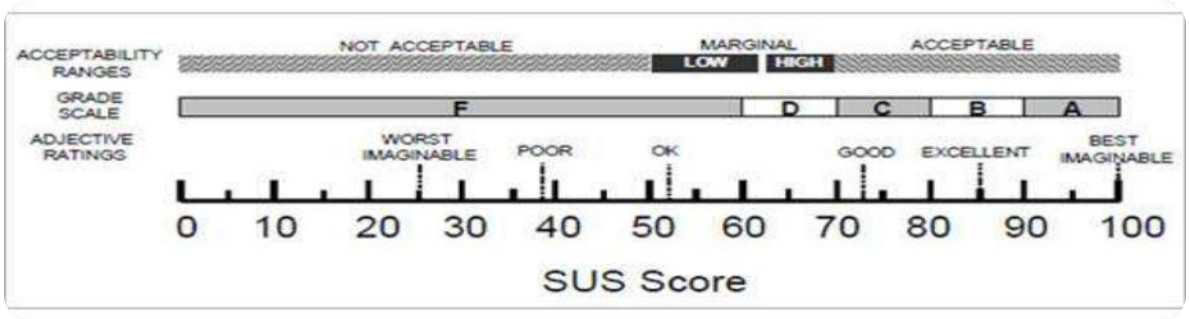
The System Usability Scale (SUS) was used for measuring the usability. SUS is a 10 item questionnaire with five response options for respondents from strongly agree to strongly disagree.

The System Usability Scale (SUS) was chosen because it is a very easy scale to administer to participants and can be used on small sample sizes with reliable results.

The following steps are used in Scoring SUS questionnaires

i. For odd items: subtract one from the user response.
ii. For even-numbered items: subtract the user responses from 5
iii. This scales all values from 0 to 4 (with four being the most positive response).
iv. Add up the converted responses for each user and multiply that total by 2.5. This converts the range of possible values from 0 to 100 instead of from 0 to 40.

Figure 17: SUS Questionnaire Adjective Scale



The SUS adjective scale is used to grade the SUS scores using alphabets(A,B,C,D,F),give adjective ratings(worst imaginable,poor,ok,good,excellent and best imaginable) and provide acceptability ranges.

4.1.3.1 SUS Questionnaire Results

In this research, the focus was on the users' satisfaction in using the online assessment system.

SUS questionnaire options used were; 1.Strongly disagree, 2.Disagree, 3.Neutral, 4.Agree, 5.Strongly agree.

Table 3: Overall summary of responses in percentage from all participants

Question no.	Code	Value	Frequency	Percentage (out of 100%)
Question 1	1	strongly disagree	2	4.444444444
	2	disagree	0	0
	3	neutral	3	6.666666667
	4	agree	5	11.11111111
	5	strongly agree	35	77.77777778
Question 2	1	strongly disagree	3	6.666666667
	2	disagree	0	0
	3	neutral	0	0
	4	agree	6	13.33333333
	5	strongly agree	36	80
Question 3	1	strongly disagree	4	8.888888889
	2	disagree	0	0
	3	neutral	4	8.888888889
	4	agree	2	4.444444444
	5	strongly agree	35	77.77777778
Question 4	1	strongly disagree	31	68.88888889
	2	disagree	6	13.33333333
	3	neutral	5	11.11111111
	4	agree	2	4.444444444
	5	strongly agree	1	2.222222222
Question 5	1	strongly disagree	0	0
	2	disagree	1	2.222222222
	3	neutral	4	8.888888889
	4	agree	4	8.888888889
	5	strongly agree	36	80
Question 6	1	strongly disagree	35	77.77777778
	2	disagree	6	13.33333333
	3	neutral	4	8.888888889
	4	agree	0	0
	5	strongly agree	0	0
Question 7	1	strongly disagree	1	2.222222222
	2	disagree	2	4.444444444
	3	neutral	2	4.444444444
	4	agree	4	8.888888889
	5	strongly agree	36	80
Question 8	1	strongly disagree	40	88.88888889
	2	disagree	4	8.888888889
	3	neutral	1	2.222222222
	4	agree	0	0
	5	strongly agree	0	0
Question 9	1	strongly disagree	0	0
	2	disagree	2	4.444444444

	3	neutral	2	4.444444444
	4	agree	3	6.666666667
	5	strongly agree	38	84.44444444
Question 10	1	strongly disagree	37	82.22222222
	2	disagree	3	6.666666667
	3	neutral	3	6.666666667
	4	agree	2	4.444444444
	5	strongly agree	0	0

The table above shows an overall summary of responses in percentage from 45 participants who comprised of 29 students, 10 teachers and 6 administrators.

Table 4: Summary of Students Responses

Category	**Student**									
Participant id	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**
p1	5	5	5	1	5	1	5	1	5	1
p10	4	5	5	3	5	3	5	1	5	1
p11	5	5	4	1	5	1	5	2	5	1
p12	5	5	1	1	5	1	4	1	5	1
p15	5	5	5	1	5	1	5	1	5	1
p16	4	4	5	1	5	1	5	1	5	1
p18	5	5	5	3	5	1	5	1	2	1
p20	5	5	3	5	4	1	5	1	5	1
p21	5	5	5	4	5	1	5	1	5	1
p23	5	5	5	3	3	1	5	1	5	1
p24	5	4	5	2	5	2	5	1	5	1
p25	4	5	5	1	5	1	4	1	4	1
p29	5	5	5	1	5	1	5	1	5	1
p30	5	5	3	1	5	1	5	1	5	1
p31	5	5	5	2	5	1	5	3	5	4
p32	5	5	5	2	5	1	2	1	5	1
p34	5	5	5	1	4	1	3	1	5	1
p35	5	5	5	1	5	3	5	1	2	1
p36	5	4	5	1	5	1	2	1	3	1
p37	5	5	5	1	5	2	5	1	5	2
p38	5	5	1	2	3	1	5	1	5	3
p39	4	5	5	2	5	1	5	2	5	1
p4	5	5	5	1	5	3	5	1	5	2
p41	5	5	5	1	5	2	5	1	5	1
p44	1	5	1	1	5	1	5	1	5	1
p45	4	5	5	1	5	3	3	1	3	4
p6	5	5	5	3	5	1	4	1	4	1
p7	3	5	5	1	4	1	5	1	5	1
p9	5	4	5	1	5	1	5	1	5	1

The table above shows a summary of 29 students' responses .The students have been allocated participant ids and the table shows how they answered the 10 questions on the questionnaire.

Values 1-5 have been used to represent SUS questionnaire format scale; 1. Strongly disagree, 2-Disagree,-Neutral, 4-Agree, 5-Strongly agree.

Table 5: Table showing satisfaction Level of students obtained from SUS scores

Satisfaction Level	No of Students	%
Satisfied	27	93.10345
Not satisfied	2	6.896552

The table above shows that there is a 93.1% satisfaction range. This means the students accepted to use the system.

Table 6: Summary of Instructor's responses

Category	Instructor									
Participant id	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
p13	3	1	5	1	3	1	5	1	4	1
p2	5	5	5	1	5	1	5	1	5	1
p26	5	5	1	1	5	1	5	1	5	3
p27	5	5	5	1	5	1	5	1	5	1
p28	5	1	5	1	4	1	5	1	5	1
p33	5	5	5	2	5	1	5	1	5	1
p40	5	4	5	1	5	1	4	1	5	1
p42	5	5	5	1	2	1	5	1	5	3
p43	5	1	5	1	5	2	5	1	5	1
p8	5	5	3	1	5	2	5	1	5	1

The table above shows a summary of 10 instructors' responses .The instructors have been allocated participant IDs and the table shows how they answered the 10 questions on the questionnaire.

Values 1-5 have been used to represent SUS questionnaire format scale; 1.Strongly disagree, 2.Disagree, 3.Neutral, 4.Agree, 5.Strongly agree.

Table 7: Table showing satisfaction Level of Instructors obtained from SUS scores

Satisfaction Level	No of Instructors	%
Satisfied	10	100
Not Satisfied	0	0

The table above shows that there is a 100% satisfaction range. This means the instructors were satisfied with the system's usability.

Table 8: Summary of Administrators responses

Category	Administrator									
participant id	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
p14	5	5	3	1	5	2	5	1	5	2
p17	5	5	5	1	3	1	5	1	5	1
p19	5	5	5	4	5	1	5	2	5	1
p22	5	5	5	3	5	1	1	1	5	1
p3	3	4	4	1	5	1	5	1	5	1
p5	1	5	5	1	5	1	5	2	5	1

The table above shows a summary of 6 administrators' responses .The administrators have been allocated participant ids and the table shows how they answered the 10 questions on the questionnaire.

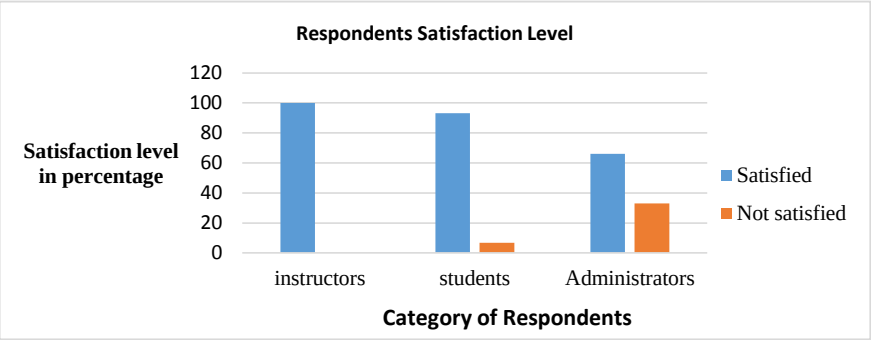
Values 1-5 have been used to represent SUS questionnaire format scale; 1.Strongly disagree, 2.Disagree3.Neutral, 4.Agree, 5.Strongly agree

Table 9: Table showing satisfaction level of Administrators obtained from SUS scores

Satisfaction Level	No of Administrators	%
Satisfied	4	66
Not Satisfied	2	33

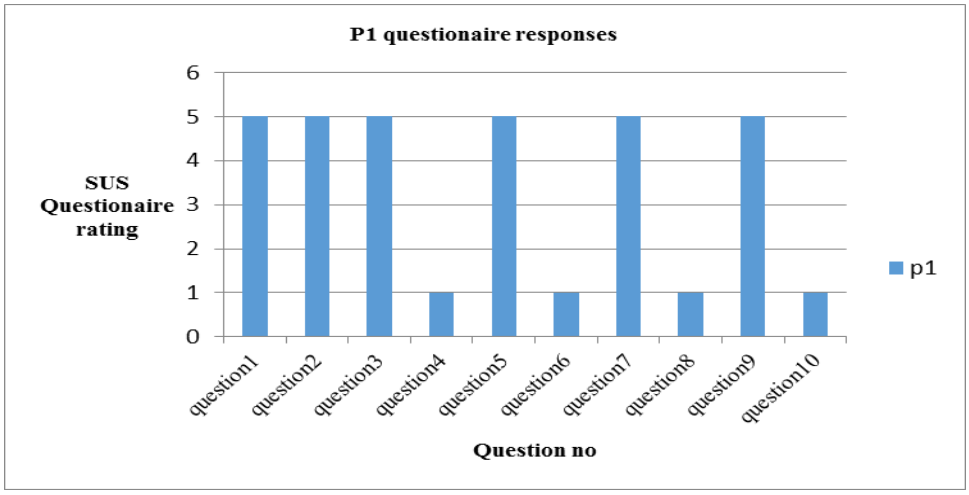
The table above shows that there is a 66% satisfaction range. This means 66%of the administrators were satisfied with the system's usability.

Figure 18: Chart showing system satisfaction level of all respondents



The above chart shows that 100% of instructors accepted the system,93% of students accepted the system while 6% did not.66% of administrators accepted the system while 33% did not.

Figure 19: Chart showing participant 1 responses to SUS Questionnaire questions



The above chart shows how participant (P1) responded to the ten questions in the SUS questionnaire with ratings from 1-5.

4.1.4 Code Assessment Testing

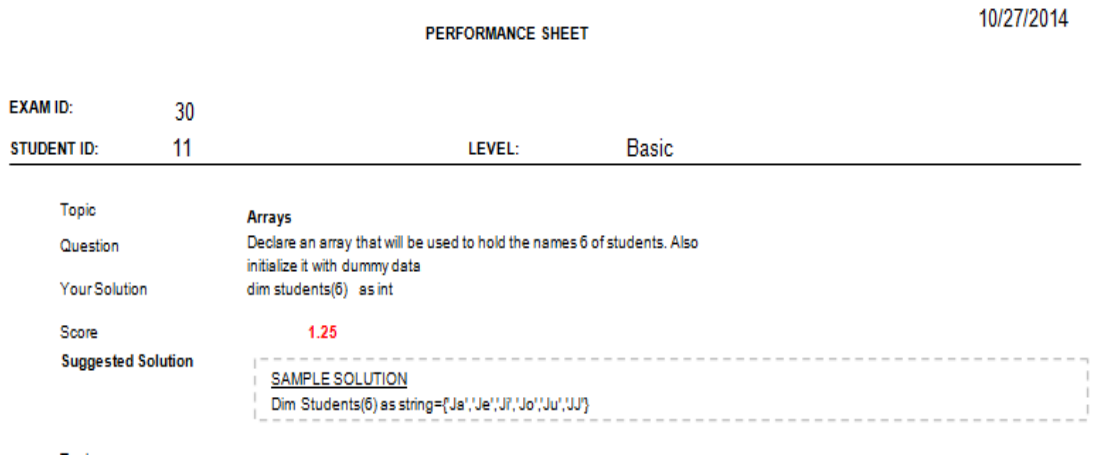
Every code assessment system requires the evaluation of the following features to determine whether it is working properly.

i. **Program correctness**- A program is said to be correct if it produces the correct output for every possible input.
ii. **Assessment adequacy**-This involves checking whether the questions obey pedagogical theories.
iii. **Program Style-** Checking for correct program style involves evaluating indentation, variable naming conventions and quality of comments.
iv. **Design**-Involves checking whether the structure of the solution provided matches the allowed structure

4.1.4.1 Code assessment Testing Results

Code written by the students was assessed using character matching strategy. Character matching is one of the few effective static analysis methods for evaluating output correctness. The solution provided by the teacher and the input from the student were matched and compared for syntax errors.

Figure 20: Student 1 performance sheet



10/27/2014

PERFORMANCE SHEET

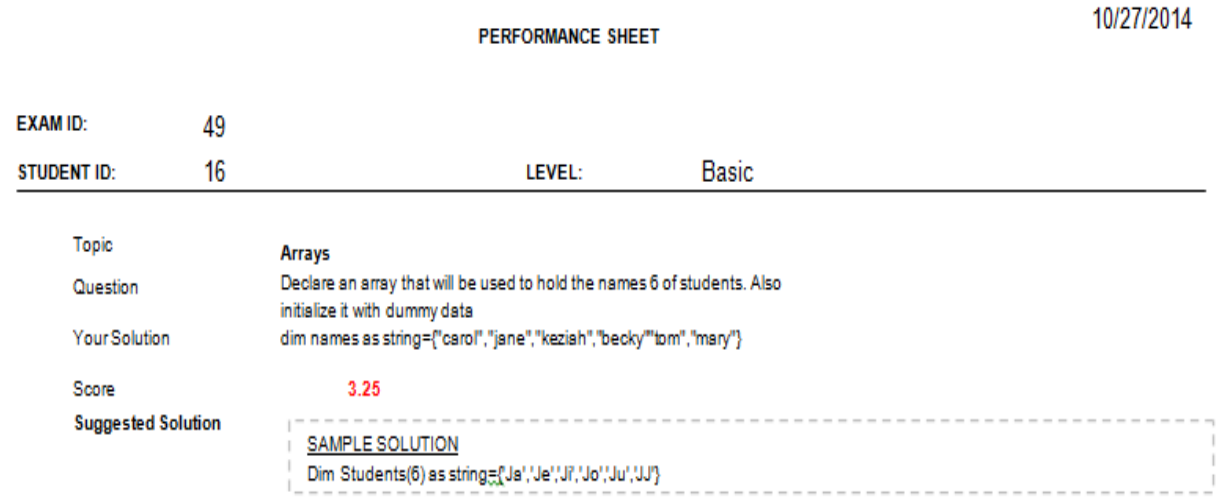
EXAM ID: 30

STUDENT ID: 11 LEVEL: Basic

Topic	Arrays
Question	Declare an array that will be used to hold the names 6 of students. Also initialize it with dummy data
Your Solution	dim students(6) as int
Score	1.25
Suggested Solution	SAMPLE SOLUTION Dim Students(6) as string={'Ja','Je','Ji','Jo','Ju','JJ'}

The figure above has been used to show student's 1 input solution comparison with the instructor's solution. Out of 5 marks, the student scored 1.25 marks for array declaration and for assigning the array with 6 elements as required but could not score more marks because the wrong data type was detected by the system and the answer was incomplete.

Figure 21: Student 2 performance sheet



10/27/2014

PERFORMANCE SHEET

EXAM ID: 49

STUDENT ID: 16 LEVEL: Basic

Topic	Arrays
Question	Declare an array that will be used to hold the names 6 of students. Also initialize it with dummy data
Your Solution	dim names as string={"carol","jane","kezish","becky""tom","mary"}
Score	3.25
Suggested Solution	SAMPLE SOLUTION Dim Students(6) as string={'Ja','Je','Ji','Jo','Ju','JJ'}

The figure above has been used to show student's 2 input solution comparison with the instructor's solution. Out of 5 marks, the student scored 3.25 marks better than student 1.The student could not score the 5 marks because of syntax errors detected on the assigned array elements.

4.1.4.2 System marking vs. Manual marking

A team of three instructors from the learning institution were requested to manually mark the submitted student's code so as to compare the scores with those of the system.

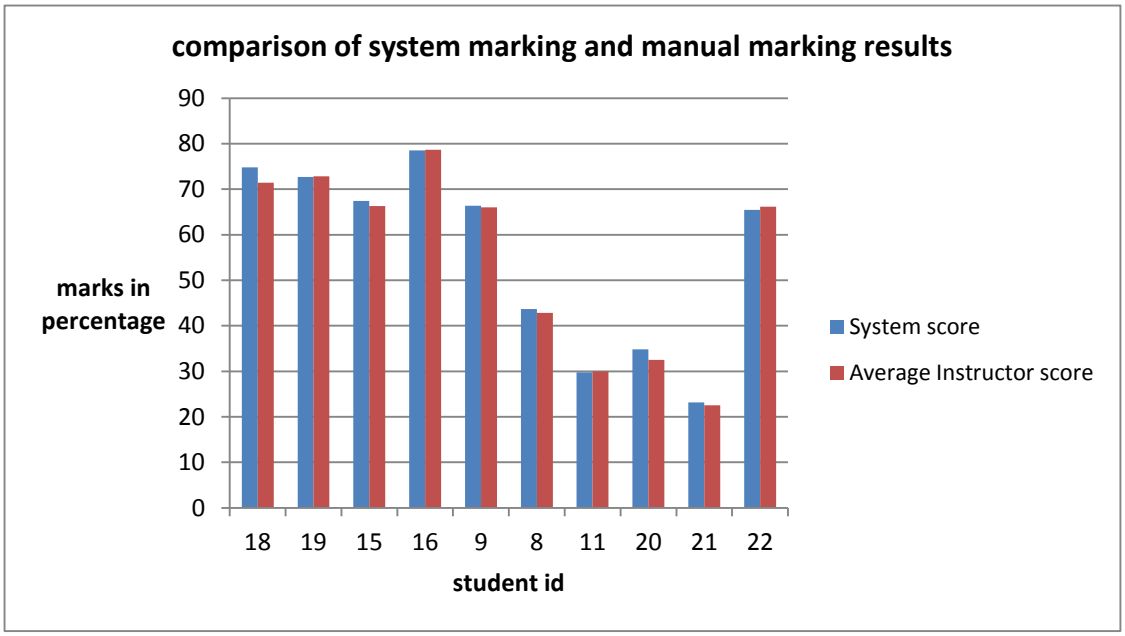
Table 10: Table showing manual results in percentage compared to system results

Student ID	System score	Instructor 1	Instructor2	Instructor 3	Average Instructor score	Difference
18	74.83	70.7	72.6	71	71.43333333	3.396666667
19	72.69	72	73	73.6	72.86666667	-0.176666667
15	67.43	66	68	65	66.33333333	1.096666667
16	78.51	79	78.2	78.8	78.66666667	-0.156666667
9	66.36	65	67	66	66	0.36
8	43.7	42	42.6	44	42.86666667	0.833333333
11	29.76	30.2	31	28.8	30	-0.24
20	34.81	33	32	32.6	32.53333333	2.276666667
21	23.16	21	24	22.6	22.53333333	0.626666667
22	65.47	66	67.2	65.2	66.13333333	-0.663333333

From the table above the difference in marks between the system and manual marking was very minimal with the highest difference being 3.39 marks. Differences between 0-5 marks were within the acceptable range. Differences above 5 marks can be addressed by remarking the same exam by a different team of instructors.

The above results expressed confidence in the system and showed that the system had an acceptable accuracy level.

Figure 22: Chart showing a comparison of system marking and manual marking results



The above chart shows results of instructor marking compared to the system's marking. From the results shown the differences between the two sets of results was very minor.

Table 11: Table showing Questions, Topics and Allocated marks for Basic level exam

Question	Marks	Topic
Declare an array that will be used to hold the names 6 of students. Also initialize it with dummy data	5	arrays
Write a small function that uses a FOR loop to print the numbers 1 to 10	5	loops
Write VB.NET code to declare a variable to store the age of a person.	5	variables
Create a function findmax that takes two integer values and returns the larger of the two.	10	functions
write a program using a for loop that will display the following time table in a list box	10	loops
1*2=2		
2*2=4		
3*2=6		
4*2=8		
5*2=10		
6*2=12		
7*2=14		
8*2=16		
9*2=18		
10*2=20		
Write a program to add two numbers and display the result	5	mathematical operators
Write a program to multiply two numbers and display the result	5	mathematical operators
Write a program to compare two values and indicate which number is bigger.	10	control structures
write a function named factorial that calculates factorial for a given number using a recursive function:	10	functions
Write a vb.net program to find the area of a circle	5	constants

The table above shows a list of 10 questions added to the system in a basic level exam. Marks were allocated depending on the complexity of the question. Questions were derived from the following topics; arrays, variables, functions, loops, mathematical operators, control structures and constants.

Table 12: Table showing Students' scores in each question

Question No.		q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	Total marks
Allocated Marks per question		**5**	**5**	**5**	**10**	**10**	**5**	**5**	**10**	**10**	**5**	**70**
Student id	18	3.88	3.5	2.5	8	8	4.5	4.5	9	8.5	0	52.38
	19	3.88	3.5	2.5	8	7	4.5	4.5	9	8	0	50.88
	15	1.25	2.5	3	7	6.33	3.25	4.5	9	8.5	1.88	47.21
	16	3.25	3.5	3	7.5	7.33	5	4	9	8	4.38	54.96
	9	2.12	3	5	8	5.33	3.5	3	8.5	8	0	46.45
	8	2.5	2.5	1.75	5	5.67	1.5	1.5	3.67	6.5	0	30.59
	11	2	1.5	5	1	0	1	0	3.33	7	0	20.83
	20	2.62	2	1.75	1.5	7.33	0.5	0.5	2.67	5.5	0	24.37
	21	0.62	2.5	1.25	2	4.17	0	0	1.67	4	0	16.21
	22	0	2.5	5	7	6.33	3.25	3.25	10	8.5	0	45.83

From the above scores on each question, students attained better scores on loops, variables, functions, and mathematical operators and scored poorly on arrays and constants.

4.1.5 System Effectiveness Testing

This was conducted with the help of a questionnaire which was administered to 29 students and 10 Instructors.

System Effectiveness Testing Results

Table 13: Table showing students responses on system effectiveness

Category: Students					
Evaluation Criteria	**Strongly agree**	**Agree**	**Neutral**	**Disagree**	**Strongly Disagree**
Satisfied with exam marking	86.2%	10.3%	3.4%	0%	0%
Satisfied with score computation	96.5%	3.4%	0%	0%	0%
Effective feedback	100%	0%	0%	0%	0%

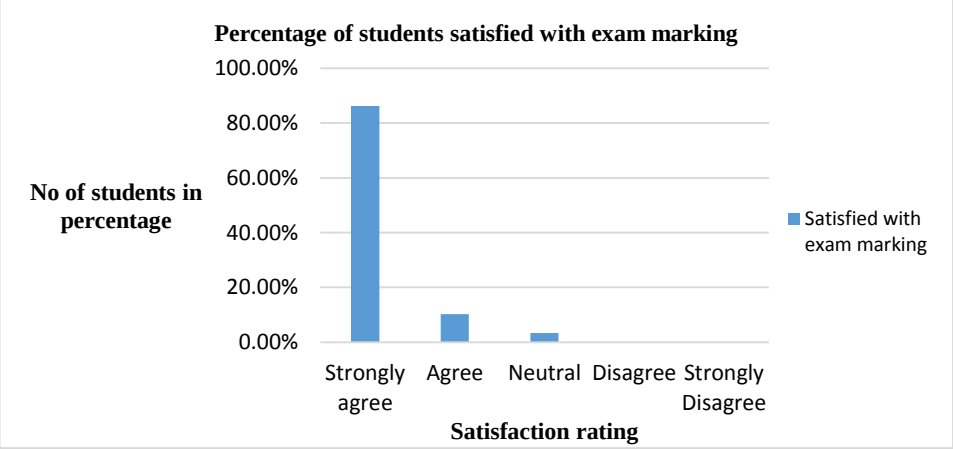
The table above shows that over 80% of students found the system to be effective on exam marking, score computation and feedback.

Table 14 :Table showing Instructors responses on system effectiveness

Category: Instructors						
Evaluation Criteria	**Strongly agree**	**Agree**	**Neutral**		**Disagree**	**Strongly Disagree**
Satisfied with exam marking	80%	10%	10%		0%	0%
Satisfied with score computation	100%	0%	0%		0%	0%
Reduction in workload	90%	10%	0%		0%	0%

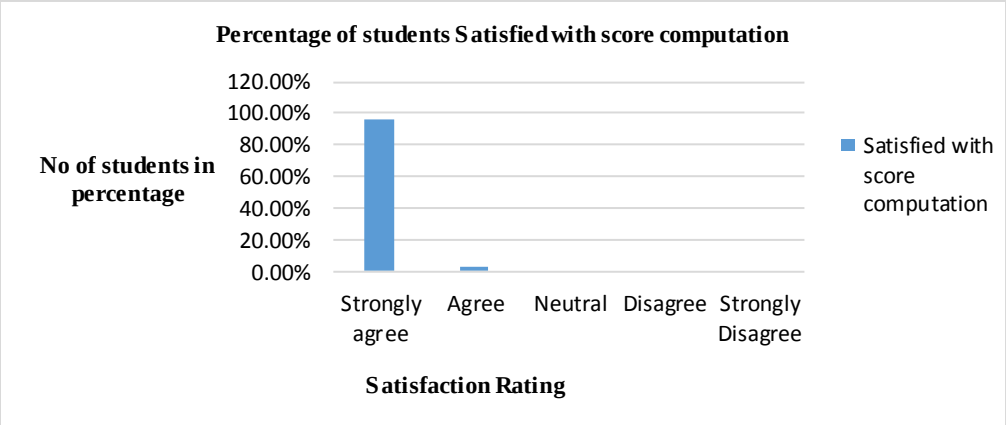
The table above shows that over 90% of instructors found the system to be effective on exam marking, score computation and workload reduction.

Figure 23: Chart showing percentage of students satisfied with system's marking



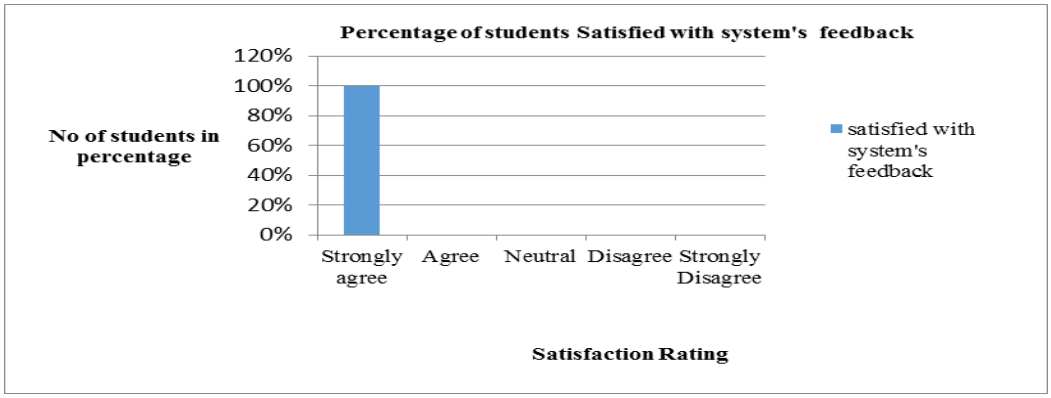
From the above chart, more than 80% are satisfied with the system's marking of exams.

Figure 24: Chart showing percentage of students satisfied with computation of test scores.



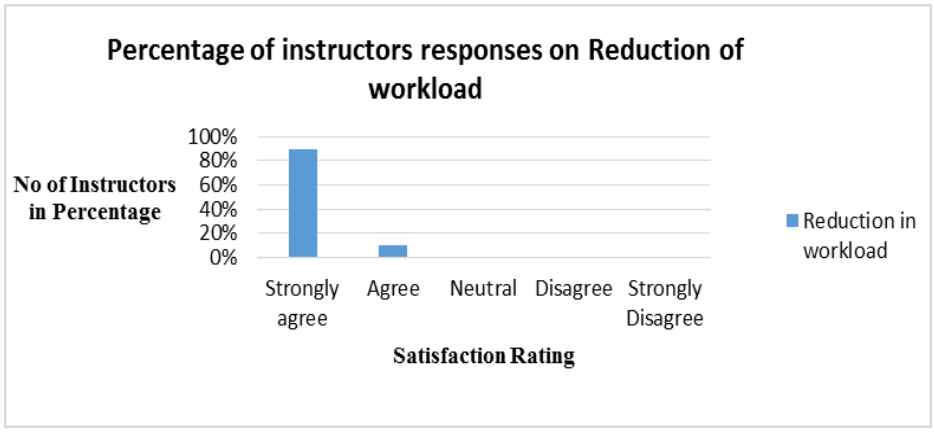
The chart above shows that 96.5% of the students are satisfied with score computation and 3% are not.

Figure 25: Chart showing percentage of students satisfied with the system's feedback



The above chart shows the percentage number of students who are satisfied with system's feedback.

Figure 26: Chart showing percentage of Instructors Responses on reduction of workload



The above chart shows 90% of the instructors strongly agree to a reduction in their workload.

4.2 Results summary

From the usability testing findings, 93.1% of the students, 100% of the instructors and 66% of the administrators accepted the system. Many respondents agreed to use the system frequently as they found it easy to use and were comfortable with the system's functionality. During System testing, the system was evaluated against some test data and the results from expected system response were acceptable. Student's response on exam marking, the system's feedback and score computation was quite impressive. Code assessment testing was conducted and programs were evaluated for correctness. The code assessment system performance results were evaluated against manual results submitted by three teachers. The difference in the results was very minimal and this was an indicator that the system was reliable and can be fully implemented in the learning institution.

CHAPTER 5: CONCLUSION

The developed system is an online code assessment system for visual basic.net programs that delivers questions set by the instructors to the student and generates the report of the results of students who take the examination as well as overall examination result summary based on the users query.

One of the objectives of this research project was to develop an online code assessment system that is able to assess correctness of visual basic.net programs and provide instant feedback. The online code assessment system was developed and was tested using various testing strategies to ensure that it was working correctly.

Results of usability testing carried out indicate that 93.1% of the students accepted to use the system, 100% of the instructors were satisfied with the system's usability. The last category was the administrators' category where 66% of the administrators who responded agreed to use the system.

System effectiveness tests were also carried out and over 80% of the students found the system to be effective on exam marking, score computation and effective feedback.

Exam marking was carried out using character matching strategy which is one of the assessment methods under static analysis. Further, student's answers were marked manually by three instructors from the learning institution and their results compared with those of the system. The results were analysed for differences and the conclusion was that the system was reliable since the difference in the manual results and system results was very minimal.90% of the instructors agreed to their workloads being reduced since the system was capable of marking the exam, generate results and provide instant feedback to the students.

5.1 Response to the research questions

The research was guided by the following research questions:

i.What are the existing types of online assessment systems for programming exams? This question was answered by a research on the different types of online assessment systems available today. From the study of literature review, the main types of online assessment systems are online compilers and online assessment systems. What was implemented in the learning institution is an online assessment system.

ii.What are the limitations of multiple choice format questions in programming examinations? Multiple choice is a form of assessment in which respondents are asked to select the best possible answer (or answers) out of the choices from a list. From the research conducted, some of the draw backs of implementing multiple choice format questions include: The tests can be time

consuming to formulate, the student's skills is not tested especially in programming examinations, there is a tendency to write items requiring only factual knowledge rather than higher-level skills and that multiple Choice items do not measure ability to organize and express ideas. These limitations have been addressed with the implementation of the developed code assessment system in the learning institution.

iii.What will be the effect of implementing a code assessment system in a learning institution that offers programming training? This question was answered by researching on institutions that have implemented such systems. The research conducted proved that implementation of such a system was advantageous to the institution in that there is a reduction in the instructor's workload and students obtain immediate feedback.

iv.Which methods of assessment exist in current code assessment systems? From the research conducted, existing methods of assessment include:Sandboxing,Non-structural Similarity Analysis, Abstract syntax trees, Visual answers, Test Runs, Keyword Analysis, Mutation testing, Plagiarism Detection and Diagram Analysis. The method of assessment to use depends on the assessment approach; whether it is static analysis or dynamic analysis approach.

5.2 Further work

The system's limitation is that it can assess only one programming language (vb.net).The system should accommodate other courses with time. Other areas to be worked on in future are Plagiarism detection, and checking of program style and design during code assessment. This will help in improving the effectiveness of the system.Recently; Tang et al. (2009a) have developed a token pattern approach. The idea is to propose decomposition of the output string into groups of successive characters, called tokens that represent meaningful pieces of information. A token pattern refers to a string of tokens automatically extracted from the expected output, each having a type, value and associated (default) matching rule(s). Matching rules are the criteria for determining correctness when the token is compared with the actual output. The approach is still under development, and further work is necessary to evaluate its potential. In practice, instructors usually use not just one strategy, but a combination of strategies. Currently, instructors still spend great effort in dealing with the output correctness determination problem. Hopefully, the undesirable pedagogical consequences due to the problem, as well as the effort spent by instructors, can be significantly reduced with the adoption of the new token pattern approach.

5.3 Appendix Section

Appendix A: References

1. Advances in Web Base Learning, Proceedings of the 6th International Conference on Web-based Learning (ICWL 2007), Springer, LNCS 4823, pp. 584-596.

2. Ala-Mutka, K., Uimonen, T., Järvinen, HM.: Supporting students in C++ programming courses with automatic program style assessment Journal of Information Technology Education, vol. 3, pp. 245-262.

3. Ala-Mutka, K., 2005. A survey of automated assessment approaches for programming assignments. Science Education, 15(2):83-102

4. Artal, CG., Suarez, M.D.A., Perez, I.S., Lopez, R.Q.: OLC, On-Line Compiler to Teach Programming Languages, International Journal of Computers, Communications & Control, vol. 3, no. 1, pp. 69-79.

5. Cheang, B., Kurnia, A., Lim, A., and Oon, W.-C. 2003. On automated grading of programming assignments in an academic institution. *Computers and Education 41*, 121–131.

6. Choy, M., Lam, S., Poon, C.K., Wang, F.L., Yu, Y.T. and Yuen, L. (2008). Design and implementation of an automated system for assessment of computer programming assignments.

7. Daly, C. and Waldron, J. (2004). Assessing the assessment of programming ability. ACM SIGCSE Bulletin, Vol. 36, No. 1, pp. 210-213.

8. Douce C., Livingstone D., Orwell J. test-based assessment of programming: a review. JERIC - Journal of Educa Computing, 5(3):4

9. Jackson, D. and Usher, M. (1997). Grading student programs using ASSYST. ACM SIGCSE Bulletin, Vol. 29, No. 1, pp.335-339.

10. Jackson, D. (2000). A semi-automated approach to online assessment. ACM SIGCSE Bulletin, Vol. 32, No. 3, pp. 164-168.

11. Joy, M., Griffiths, N. and Boyatt, R. (2005). The BOSS online submission and assessment system. Journal on Educational Resources in Computing, Vol.5, No 3, pp. 1- 28.

12. Mandal A.K., Mandal C. & Reade C.M.P. 2006. Architecture of an Automatic Program Evaluation System. In CSIE Proceeding.

13. Spacco, J., Hovemeyer, D., Pugh, W., Emad, F., Hollingsworth, J.K. and Padua-Perez, N. (2006). Experiences with Marmoset: Designing and using an advanced submission and testing system for programming courses. ACM SIGCSE Bulletin, Vol. 38, No. 3, pp. 13-17.

14. Tang C.M., Yu Y. T., Poon C.K. 2010. A Review of the Strategies for Output Correctness Determination in Automated Assessment of Student Programs. In Proceedings of Global Chinese Conference on Computers in Education.

15. Traynor, D. and Gibson, J.P. (2005). Synthesis and analysis of automatic assessment methods in CS1. ACM SIGCSE Bulletin, Vol. 37, No. 1, pp. 495-499.

16. Tremblay, G., Guerin, A., Pons, A. And Salah, A.(2008). Oto, a generic and extensible tool for marking programming assignments. Software: Practice and Experience, Vol. 38, No. 3, p.p. 307-333

17. Woit, D. and Mason, D. (2003). Effectiveness of online assessment. ACM SIGCSE Bulletin, Vol. 35, No. 1, pp. 137-141.

18. Yu, Y.T., Poon, C.K. and Choy, M. (2006). Experiences with PASS: Developing and using a programming assignment assessment system. Proceedings of the Sixth International Conference on Quality Software (QSIC'06), IEEE, pp. 360-368.

19. Karakaya, Z. (2001). Development and implementation of an on-line exam for a programming language course. Ankara: Metu.

20. Carter, J., English, J., Ala-Mutka, K., Dick, M., Fone, W., Fuller, U., & Sheard, J. 2003. How shall we assess this? ACM SIGCSE Bulletin, 35(4), 107 – 123.

21. Chen, J. Y. and Lu, J. F. 1993. A New Metric for Object-oriented Design. Information Software Technology Vol 35 (April 1993):232–240.

22. Chu, H. D., Dobson , J. E. and Liu I.C.. 2006. FAST-A Framework for Automating Statistic-based Testing 1997 [cited Jun 2006]

23. Saikkonen, R., Malmi , L., and Korhonen, A.. 2001. Fully Automatic Assessment of Programming Exercises. Paper read at ITiCSE2001.

24. Shafer, S. C. 2005. LUDWIG: An Online Programming Tutoring and Assessment System. Inroads – The SIGCSE Bulletin 37 (June 2005):56-60

Appendix B: Questionnaires

Questionnaire-Cover letter

Dear participant,

My name is Catherine Wambui Mukunga, a Master's student at The University of Nairobi taking Computer Science.

You are invited to participate in a research project under the title: Online code assessment system for visual basic.net programs.

The purpose of this survey is to get your views on the proposed online code assessment system for visual basic.net programs regarding the system usability.

This study has been approved by the Director of Education.

Please be assured that the information filled in this questionnaire will be treated with confidentiality. Filling this System Usability questionnaire will not take more than 15 minutes of your time.

The questionnaire contains ten questions. Summary results will be communicated to you two days from today through your email address.

Should you have any queries or comments regarding this survey, you are welcome to contact me through:

NAME: CATHERINE WAMBUI MUKUNGA

C/O UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFOMATICS

P.O BOX 30197, G.P.O

NAIROBI,KENYA
CELL PHONE: 0720-269144

EMAIL: mukunga@students.uonbi.ac.ke

Yours sincerely

Catherine Wambui Mukunga

SYSTEM USABILITY SCALE (SUS) QUESTIONAIRE

Instructions: For each of the following questions mark one box that best describes your answer.

Participant id	(leave this section blank)
Category(Student/Instructor/Administrator)	
Date	

1. I think that I would like to use this system frequently.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

1. I found the system unnecessarily complex.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

2. I thought the system was easy to use.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

3. I think that I would need the support of a technical person to be able to use this system.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

4. I found the various functions in this system were well integrated.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

5. I thought there was too much inconsistency in this system.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

6. I would imagine that most people would learn to use this system very quickly.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

7. I found the system very cumbersome to use.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

8. I felt very confident using the system.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

9. I needed to learn a lot of things before I could get going with this system.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

Additional Comments

SYSTEM EFFECTIVENESS EVALUATION QUESTIONNAIRE

Instructions: For each of the following questions mark one box that best describes your answer.

Participant id	(leave this section blank)
Category(Student/Instructor)	
Date	

1. The system has contributed to reducing my workload

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

2. I am satisfied with the system's marking

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

3. I am satisfied with the system's computation of my score

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

4. The assessment feedback has contributed to improved programming skills.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
☐	☐	☐	☐	☐

Additional Comments

Appendix C: Budget

Item	Amount
Stationery(printing paper)	2,000
Modem and internet connection	3,000
Air-time	2,000
Laptop	50,000
Printer	6,000
	Total amount 63,000

Appendix D: Project schedule

Milestone	Duration	Activity
Proposal	6th Jan-22nd 10th April 2014	Problem definition, literature review and methodology.
System design	11th April- 31st may 2014	Designing the system architecture and developing the system.
System testing	1st June- 22nd June	Testing the developed system using some test data and analysing the results.
Data collection, analysis and reporting)	23rd June -7th July 2014	Designing questionnaire, collecting and analysing the data.
Documentation and final report	8th july-30th July 2014	Documenting the system information and how it will work
Milestone 2 project presentation	31st July 2014	Mile stone 2 presentation
Milestone 3 project presentation	October 2014	Milestone 3 presentation

Appendix E: Source code

Marking an exam

```
Imports ExamSys.clsMain
Imports System.Web
'Imports ExamSys.VBNetCompiler
Imports ExamSys.cVBEvalProvider
Public Class markexam
    Inherits System.Web.UI.Page
    Dim OverallScore As Decimal = 0
    Private Sub markexam_Init(sender As Object, e As System.EventArgs) Handles Me.Init
        Response.Cache.SetCacheability(HttpCacheability.NoCache)
        'Response.Cache.SetExpires(DateTime.Now.AddDays(-1))
        'Response.Cache.SetCacheability(HttpCacheability.NoCache)
        Response.Cache.SetExpires(DateTime.Now.AddSeconds(-1))
        Response.Cache.SetNoStore()
    End Sub
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        'If IsCallback Then Response.Redirect("~/login.aspx")
        'If DoneExam Then Response.Redirect("~/login.aspx") Else
        "Compile()
        Dim ExamID As Integer = Request.QueryString.Item("bkid")
        UpdateExam(ExamID)
        DoneExam = True
        Response.Redirect("~/rptViewer.aspx?rptname=studentPerformance&params=bkid=" & ExamID)
    End Sub
    Sub UpdateExam(BookingID)
      Dim ExamSess As Integer = FN_ReturnValue("select ExamsessID from [Exam.ExamSessions] where IsCurrent
=1")
        Dim dsQuestions As DataSet
        Dim dtQuestions As DataTable
        dsQuestions = FN_ReturnDs("select * from [Exam.Answers] where Booking_id=" & BookingID)
        dtQuestions = dsQuestions.Tables(0)
        '//Calculate marks --A real algorithm required to check on the answers provided instead of random numbers
        Dim TotalMarks As Decimal
        For x = 0 To dtQuestions.Rows.Count - 1
            Dim score As Decimal = 0
```

```
        Dim Marks As Decimal = FN_ReturnValueDecimal("select marks from [Exam.Questions] where QUESTION_ID =" & dtQuestions.Rows(x).Item("Question"))
        TotalMarks += Marks
        If IsDBNull(dtQuestions.Rows(x).Item("Q_Answer")) Then
            score = 0
        Else
            If Trim(dtQuestions.Rows(x).Item("Q_Answer").ToString).Length = 0 Then
                score = 0
            Else
/                '//get errors
                Dim qid As Decimal = 0
                qid = CompileAndRunCode(dtQuestions.Rows(x).Item("Q_Answer"))
                If qid > Marks Then qid = Marks
                'score = qid / Marks
                '//find key words
                score = ((Marks - qid) / Marks) * Marks
                Dim KeyWords As String
                Dim dsKeywords As New DataSet
                Dim dtKeywords As New DataTable
            KeyWords = FN_ReturnString("select KEYWORDS from [Exam.Questions] where QUESTION_ID =" & dtQuestions.Rows(x).Item("Question"))
                dsKeywords = FN_ReturnDs("select * from [dbo].[split]('" & KeyWords & "',',')")
                dtKeywords = dsKeywords.Tables(0)
                Dim Index As Integer
                Dim FoundKeyword As Decimal = 0
                For q = 0 To dtKeywords.Rows.Count - 1
                Index = dtQuestions.Rows(x).Item("Q_Answer").indexOf("" & dtKeywords.Rows(q).Item("Data").ToString & "")
                    If Index <> -1 Then
                        '//keyword found
                        FoundKeyword += 1
                    Else
                    End If
                Next
                FoundKeyword = (FoundKeyword / dtKeywords.Rows.Count) * Marks
                score = (score + FoundKeyword) / 2
                ' Initialize the random-number generator.
                'Randomize(6)
```

```
                ' Generate random value between 1 and 6.
                'Dim value As Integer = CInt(Int((10 * Rnd()) + 1))
            End If
        End If
        FN_ExecuteQuery("update [Exam.Answers] set Marks_Scored=" & Math.Round(score, 2) & " where trx_id=" & dtQuestions.Rows(x).Item("trx_id"))
        OverallScore += score
        OverallScore = Math.Round(OverallScore, 2)
    Next
    '//Update master table
    FN_ExecuteQuery("Update [Exam.Booking] set [Status]='Done',[Session_ID]=" & ExamSess & ",[OverallScore]='" & OverallScore & "',TotalMarks='" & TotalMarks & "' where [Booking_ID]=" & BookingID)
    DoneExam = True
End Sub
'Function gET(vbcode As String)
'    ' Generate the Code Framework
'    Dim sb As StringBuilder = New StringBuilder("")
'    sb.Append("Imports System" & vbCrLf)
'    sb.Append("Imports System.Xml" & vbCrLf)
'    sb.Append("Imports System.Data" & vbCrLf)
'    ' Build a little wrapper code, with our passed in code in the middle
'    sb.Append("Namespace dValuate" & vbCrLf)
'    sb.Append("Class EvalRunTime " & vbCrLf)
'    sb.Append("Public Function EvaluateIt() As Object " & vbCrLf)
'    ' Insert our dynamic code
'    sb.Append(vbcode & vbCrLf)
'    sb.Append("End Function " & vbCrLf)
'    sb.Append("End Class " & vbCrLf)
'    sb.Append("End Namespace" & vbCrLf)
'    Return sb.ToString
'    Dim c As VBCodeProvider = New VBCodeProvider
'    Dim oRetVal As Object = CompileAndRunCode(Me.RichTextBox1.Text)
'    MsgBox(oRetVal)
'End Function
'Function CompileAndRunCode()
'    ' Instance our CodeDom wrapper
'    Dim ep As New cVBEvalProvider
'End Function
```

```
Public Function CompileAndRunCode(ByVal VBCodeToExecute As String) As Object
    Dim sReturn_DataType As String
    Dim sReturn_Value As String = ""
    ' Instance our CodeDom wrapper
    Dim ep As New cVBEvalProvider
    Try
        ' Compile and run
        Dim objResult As Object = ep.Eval(VBCodeToExecute)
        If ep.CompilerErrors.Count <> 0 Then
            'Diagnostics.Debug.WriteLine("CompileAndRunCode: Compile Error Count = " & ep.CompilerErrors.Count)
            'Diagnostics.Debug.WriteLine(ep.CompilerErrors.Item(0))
            'Return "ERROR" ' Forget it
        End If
        If ep.CompilerErrors.HasWarnings Then
            'ep.CompilerErrors.
        End If
        "Dim t As Type = objResult.GetType()
        "If t.ToString() = "System.String" Then
        "    sReturn_DataType = t.ToString
        "    sReturn_Value = Convert.ToString(objResult)
        "Else
        "    ' Some other type of data - not really handled at
        "    ' this point. rwd
        "    'ToDo: Add handlers for other data return types, if needed
        "    ' Here is an example to handle a dataset...
        "    'Dim ds As DataSet = DirectCast(objResult, DataSet)
        "    'DataGrid1.Visible = True
        "    'TextBox2.Visible = False
        "    'DataGrid1.DataSource = ds.Tables(0)
        "End If
        Return ep.CompilerErrors.Count
    Catch ex As Exception
        Dim sErrMsg As String
        sErrMsg = String.Format("{0}", ex.Message)
        ' Do Nothing - This is just a negative case
        ' This outcome is expected in late interpreting
        ' I suppose what I am saying is: Don't stop my program because the script writer can't write
```

```
            ' script very well.  To be fair, we could log this somewhere and notify somebody.
            Return ep.CompilerErrors.Count
        End Try
        ' Return sReturn_Value
    End Function
End Class
```

Viewing the student's score sheet

```
Public Class viewPerfomance
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    End Sub
End Class
```